



版本 2 的用户指南

# AWS Command Line Interface



# AWS Command Line Interface: 版本 2 的用户指南

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。



# Table of Contents

.....	ix
关于 AWS CLI .....	1
关于 AWS CLI 版本 2 .....	1
SDK 主要版本的维护和支持 .....	2
关于 Amazon Web Services .....	2
关于示例 .....	2
其他文档和资源 .....	3
AWS CLI 文档和资源 .....	3
其他 AWS 开发工具包 .....	3
开始使用 .....	5
先决条件 .....	6
创建 IAM 或 IAM Identity Center 管理账户 .....	6
后续步骤 .....	7
安装/更新 .....	7
AWS CLI 安装和更新说明 .....	7
AWS CLI 安装和卸载错误疑难解答 .....	20
后续步骤 .....	20
过去发行版 .....	20
AWS CLI 安装和卸载错误疑难解答 .....	36
后续步骤 .....	37
从源代码构建和安装 .....	37
为什么要从源代码构建？ .....	37
快速步骤 .....	38
步骤 1：设置所有要求 .....	40
步骤 2：配置 AWS CLI 源代码安装 .....	44
步骤 3：构建 AWS CLI .....	50
步骤 4：安装 AWS CLI .....	51
步骤 5：验证 AWS CLI 安装 .....	52
工作流程示例 .....	52
AWS CLI 安装和卸载错误疑难解答 .....	55
后续步骤 .....	55
Amazon ECR Public/Docker .....	56
先决条件 .....	56
在 Amazon ECR Public 和 Docker Hub 之间作出选择 .....	56

运行官方映像 .....	57
有关官方映像的接口和向后兼容性的说明 .....	58
使用特定版本和标签 .....	58
更新到最新的官方映像 .....	59
共享主机文件、凭据、环境变量和配置 .....	59
缩短 docker run 命令 .....	65
设置 .....	68
收集您的凭证信息以进行编程访问 .....	69
设置新的配置和凭证 .....	70
使用现有配置和凭证文件 .....	77
配置 AWS CLI .....	79
配置和凭证优先顺序 .....	79
此部分中的其他主题 .....	80
配置和凭证文件设置 .....	80
配置和凭证文件的格式 .....	81
配置设置存储在何处？ .....	88
使用命名配置文件 .....	89
使用命令设置和查看配置设置 .....	90
设置新的配置和凭证命令示例 .....	93
支持的 config 文件设置 .....	95
环境变量 .....	110
如何设置环境变量 .....	111
AWS CLI 支持的环境变量 .....	112
命令行选项 .....	120
如何使用命令行选项 .....	120
AWS CLI 支持的全局命令行选项 .....	121
命令行选项的常见用法 .....	125
命令完成 .....	126
工作原理 .....	126
在 Linux 或 macOS 上配置命令完成 .....	127
在 Windows 上配置命令完成 .....	130
重试 .....	131
可用重试模式 .....	132
配置重试模式 .....	134
查看重试日志 .....	135
使用 HTTP 代理 .....	136

使用示例 .....	136
向代理进行身份验证 .....	137
对 Amazon EC2 实例使用代理 .....	138
故障排除 .....	138
端点 .....	139
为单个命令设置端点 .....	139
为所有人设置全局终端节点 AWS 服务 .....	139
设置成为所有 AWS 服务使用 FIPS 端点 .....	141
设置成为所有 AWS 服务使用双堆栈端点 .....	142
设置特定于服务的端点 .....	143
端点配置和设置优先级 .....	146
身份验证和访问凭证 .....	147
配置和凭证优先顺序 .....	147
此部分中的其他主题 .....	148
IAM Identity Center 身份验证 .....	149
配置自动令牌刷新 .....	149
配置旧版不可刷新 .....	156
使用 IAM Identity Center 配置文件 .....	161
短期凭证 .....	163
IAM 角色 .....	164
先决条件 .....	165
IAM 角色的用法概览 .....	165
配置和使用角色 .....	166
使用 MFA .....	168
跨账户角色和外部 ID .....	169
指定角色会话名称以便于审核 .....	170
通过 Web 身份代入角色 .....	170
清除缓存的凭证 .....	172
IAM 用户 .....	172
步骤 1：创建您的 IAM 用户 .....	173
步骤 2：获取您的访问密钥 .....	173
配置 AWS CLI .....	173
将凭证用于 Amazon EC2 实例元数据 .....	175
先决条件 .....	176
配置 Amazon EC2 元数据的配置文件 .....	176
外部凭证 .....	177

使用 AWS CLI .....	180
获取帮助 .....	180
内置 AWS CLI 帮助命令 .....	181
AWS CLI 参考指南 .....	185
API 文档 .....	185
纠正错误 .....	186
其他帮助 .....	186
命令结构 .....	186
命令结构 .....	187
Wait 命令 .....	188
指定参数值 .....	189
通用参数类型 .....	190
含字符串的引号 .....	194
来自文件的参数 .....	198
生成 CLI 骨架模板 .....	201
速记语法 .....	213
自动提示 .....	215
工作原理 .....	215
自动提示功能 .....	215
自动提示模式 .....	218
配置自动提示 .....	219
控制命令输出 .....	219
灵敏输出 .....	219
服务器端与客户端输出选项 .....	220
输出格式 .....	220
分页 .....	229
筛选 输出 .....	234
返回代码 .....	257
向导 .....	258
工作方式 .....	258
别名 .....	260
先决条件 .....	260
步骤 1：创建别名文件 .....	260
步骤 2：创建别名 .....	261
步骤 3：调用别名 .....	264
别名存储库示例 .....	266

资源 .....	267
代码示例 .....	268
引导式命令示例 .....	268
DynamoDB .....	269
Amazon EC2 .....	272
S3 Glacier .....	290
IAM .....	296
Amazon S3 .....	301
Amazon SNS .....	318
命令示例 .....	320
操作和场景 .....	321
Bash 脚本示例 .....	6180
操作和场景 .....	6181
安全性 .....	6344
数据保护 .....	6344
数据加密 .....	6345
Identity and Access Management .....	6346
受众 .....	6346
使用身份进行身份验证 .....	6346
使用策略管理访问 .....	6349
如何 AWS 服务 使用 IAM .....	6351
对 AWS 身份和访问进行故障排除 .....	6351
合规性验证 .....	6353
韧性 .....	6354
基础设施安全性 .....	6354
强制实施最低 TLS 版本 .....	6355
排查错误 .....	6356
首先尝试的一般故障排除 .....	6356
检查您的 AWS CLI 命令格式 .....	6356
检查 AWS 区域 你的 AWS CLI 命令正在使用什么 .....	6357
确认您运行的是 AWS CLI 的最新版本 .....	6357
使用 --debug 选项 .....	6358
启用并查看 AWS CLI 命令历史记录日志 .....	6364
确认您的配置 AWS CLI 已完成 .....	6364
找不到命令错误 .....	6364
“aws --version”命令返回的版本与您安装的版本不同 .....	6367

卸载后，aws --version“”命令会返回一个版本 AWS CLI .....	6368
AWS CLI 处理了一个参数名称不完整的命令 .....	6369
访问被拒绝错误 .....	6370
凭证无效和密钥错误 .....	6371
签名与错误不匹配 .....	6372
SSL 证书错误 .....	6374
JSON 无效错误 .....	6375
其他 资源 .....	6377
迁移指南 .....	6378
新功能和变化 .....	6378
AWS CLI 版本 2 新功能 .....	6378
AWS CLI 版本 1 和 AWS CLI 版本 2 之间的突破性更改 .....	6380
迁移说明 .....	6386
将版本 1 替换为版本 2 .....	6386
并行安装 .....	6387
卸载 .....	6389
AWS CLI 安装和卸载错误故障排除 .....	6391
文档历史记录 .....	6392
AWS 词汇表 .....	6396

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。

# 什么是 AWS Command Line Interface ?

AWS Command Line Interface (AWS CLI) 是一个开源工具，可让您使用命令行 shell 中的命令与 AWS 服务进行交互。只需最少的配置，您就 AWS CLI 可以在终端程序的命令提示符下开始运行命令，这些命令实现 AWS Management Console 的功能与基于浏览器的功能相同：

- Linux Shell – 使用常见 Shell 程序（例如 [bash](#)、[zsh](#) 和 [tcsh](#)）在 Linux 或 macOS 中运行命令。
- Windows 命令行 — 在 Windows 上，在 Windows 命令提示符下或在 Windows 命令提示符下运行命令 PowerShell。
- 远程 – 通过远程终端程序（如 PuTTY 或 SSH）在 Amazon Elastic Compute Cloud (Amazon EC2) 实例上运行命令，或者使用 AWS Systems Manager 运行命令。

中的所有 IaaS（基础架构即服务）AWS 管理、管理和访问功能 AWS Management Console 均在 AWS API 和中提供。AWS CLI 新的 AWS IaaS AWS Management Console 功能和服务在发布时或发布后 180 天内通过 API 和 CLI 提供全部功能。

AWS CLI 提供对 AWS 服务的公共 API 的直接访问。您可以使用探索服务的功能 AWS CLI，并开发 shell 脚本来管理您的资源。除了低级别、等同于 API 的命令外，还有一些 AWS 服务为提供了自定义设置。AWS CLI 自定义项可能包括更高级别的命令，可简化具有复杂 API 的服务的使用。

## 关于 AWS CLI 版本 2

AWS CLI 版本 2 是最新主要版本 AWS CLI，支持所有最新功能。版本 2 中引入的某些功能无法向后兼容版本 1，您必须升级才能访问这些功能。版本 1 中有一些可能需要您更改脚本的“重大”更改。有关版本 2 中的重大更改的列表，请参阅 [从 AWS CLI 版本 1 迁移至版本 2](#)。

AWS CLI 版本 2 只能作为捆绑安装程序进行安装。虽然你可能会在软件包管理器中找到它，但这些都是不受支持的非官方软件包，不是由生产或管理的。AWS 我们建议您仅 AWS CLI 从官方 AWS 分发点安装，如本指南中所述。

要安装 AWS CLI 版本 2，请参阅 [the section called “安装/更新”](#)。

要检查当前安装的版本，请使用以下命令：

```
$ aws --version
aws-cli/2.15.30 Python/3.11.6 Linux/5.10.205-195.807.amzn2.x86_64 botocore/1.18.6
```

有关版本历史记录，请参阅 [AWS CLI 版本 2 变更日志](#)。GitHub



## SDK 主要版本的维护和支持

有关维护和支持开发工具包主要版本及其基础依赖关系的信息，请参阅[AWS 开发工具包和工具引用指南](#)中的以下内容：

- [AWS SDK 和工具维护政策](#)
- [AWS SDK 和工具版本支持矩阵](#)

## 关于 Amazon Web Services

Amazon Web Services (AWS) 是数字基础设施服务的集合，开发人员可在开发应用程序时加以利用。这些服务包括计算、存储、数据库和应用程序同步（消息和队列）。AWS 使用 pay-as-you-go 服务模式。您只需为您或您的应用程序使用的服务付费。此外，为了使原型设计和实验平台 AWS 更加平易近人，还提供免费使用 AWS 套餐。在此套餐中，低于某种使用水平的服务是免费的。有关 AWS 费用和免费套餐的更多信息，请参阅[AWS 免费套餐](#)。要获取 AWS 账户，请打开[AWS 主页](#)，然后选择创建 AWS 账户。

## 关于 AWS CLI 示例

本指南中 AWS Command Line Interface (AWS CLI) 示例的格式是使用下列约定进行设置的：

- 提示 – 命令提示符使用 Linux 提示符并显示为 (\$)。对于 Windows 特定的命令，C:\> 用作提示。请勿在键入命令时包含提示符。
- 目录 – 当必须从特定目录执行命令时，目录名称将显示在提示符符号之前。
- 用户输入 – 您在命令行处输入的命令文本采用 **user input** 格式。
- 可替换文本 – 变量文本（包括您选择的资源的名称，或您必须包含在命令中的由 AWS 服务生成的 ID）采用的格式为 **#####**。在多行命令中或需要特定键盘输入的命令中，键盘命令也可显示为可替换文本。
- 输出 – AWS 服务返回的输出显示在用户输入下方，采用 **computer output** 格式。

例如，以下 **aws configure** 命令示例显示了用户输入、可替换文本和输出：

1. 在命令行输入 **aws configure**，然后按 Enter 键。
2. AWS CLI 输出文本行，提示您输入其他信息。
3. 依次输入每个访问密钥，然后按 Enter。

- 然后，以显示的格式输入 AWS 区域名称，按 Enter，然后最后一次按 Enter 以跳过输出格式设置。
- 最终 Enter 命令将显示为可替换文本，因为这一行没有用户输入。

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: ENTER
```

以下示例显示带输出的简单命令。要使用此示例，请输入命令的完整文本（提示符后突出显示的文本），然后按 Enter。安全组的名称 `my-sg` 可替换为所需的安全组名称。JSON 文档（包括大括号）是输出。如果您将 CLI 配置为以文本或表格式进行输出，输出的格式将有差异。[JSON](#) 是默认输出格式。

```
$ aws ec2 create-security-group --group-name my-sg --description "My security group"
{
  "GroupId": "sg-903004f8"
}
```

## 其他文档和资源

### AWS CLI 文档和资源

除了本用户指南外，还有以下适用于 AWS CLI 的有价值的在线资源。

- [AWS CLI 版本 2 参考指南](#)
- [AWS CLI 代码示例存储库](#)
- [AWS CLI GitHub 存储库](#) 您可以在 GitHub 上查看和分流 AWS CLI 的源代码。加入 GitHub 上的用户社区，提供反馈、请求功能和提交自己的文章。
- [AWS CLI 别名示例存储库](#) 您可以在 GitHub 上查看和分流 AWS CLI 别名示例。
- [AWS CLI 版本 2 更新日志](#)

### 其他 AWS 开发工具包

根据您的用例，您可能希望使用 AWS 开发工具包或 AWS Tools for PowerShell 之一：

- [AWS Tools for PowerShell](#)
- [AWS SDK for Java](#)
- [AWS SDK for .NET](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for Ruby](#)
- [AWS SDK for Python \(Boto\)](#)
- [AWS SDK for PHP](#)
- [AWS SDK for Go](#)
- [AWS Mobile SDK for iOS](#)
- [AWS Mobile SDK for Android](#)

# 开始使用 AWS CLI

本章提供了开始使用 AWS Command Line Interface (AWS CLI) 版本 2 的步骤，并提供了相关说明的链接。

1. [完成所有先决条件](#) - 要使用 AWS CLI 访问 AWS 服务，您至少需要一个 AWS 账户和 IAM 凭证。为了提高 AWS 账户的安全性，建议您不要使用根账户凭证。您应创建一个具有最低权限的用户来为将在 AWS 中运行的任务提供访问凭证。
2. 使用以下方法之一安装或进入 AWS CLI：
  - (推荐) [the section called “安装/更新”](#)。
  - [the section called “过去发行版”](#)。安装特定版本的做法主要用于团队将其工具与特定版本保持一致的情况。
  - [the section called “从源代码构建和安装”](#)。从 GitHub 源代码构建 AWS CLI 是一种更深入的方法，主要供在我们预置安装程序不直接支持的平台上工作的客户使用。
  - [the section called “Amazon ECR Public/Docker”](#)。
  - 使用 AWS CloudShell 通过浏览器访问 AWS 控制台中的 AWS CLI 版本 2。有关更多信息，请参阅 [AWS CloudShell 用户指南](#)。
3. [进入 AWS CLI 后，第一次使用时需要在 AWS CLI 中配置您的 IAM 凭证](#)。

## 对安装程序或配置错误进行故障排除

如果您在安装、卸载或配置 AWS CLI 后遇到问题，请参阅[排查错误](#)以了解故障排除步骤。

## 主题

- [使用 AWS CLI 版本 2 的先决条件](#)
- [安装或更新到最新版本的 AWS CLI](#)
- [安装版本 2 的过去 AWS CLI 版本](#)
- [AWS CLI 从源代码构建并安装](#)
- [使用亚马逊 ECR 官方公共镜像或 Docker 镜像运行 AWS CLI](#)
- [设置 AWS CLI](#)

## 使用 AWS CLI 版本 2 的先决条件

要使用 AWS CLI 访问 AWS 服务，您需要 AWS 账户 和 IAM 凭证。运行 AWS CLI 命令时，AWS CLI 需要有权访问这些 AWS 凭证。为了提高 AWS 账户的安全性，建议您不要使用根账户凭证。您应创建一个具有最低权限的用户来为将在 AWS 中运行的任务提供访问凭证。

### 主题

- [创建 IAM 或 IAM Identity Center 管理账户](#)
- [后续步骤](#)

## 创建 IAM 或 IAM Identity Center 管理账户

### 在配置之前

要创建管理员用户，请选择以下选项之一。

选择一种方法来管理您的管理员	目的	方式	您也可以
在 IAM Identity Center 中 ( 建议 )	使用短期凭证访问 AWS。  这符合安全最佳实践。有关最佳实践的信息，请参阅《IAM 用户指南》中的 <a href="#">IAM 中的安全最佳实践</a> 。	有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 <a href="#">入门</a> 。	按照《AWS Command Line Interface 用户指南》中的 <a href="#">配置 AWS CLI 以使用 AWS IAM Identity Center</a> ，配置程式访问。
在 IAM 中 ( 不推荐使用 )	使用长期凭证访问 AWS。	按照《IAM 用户指南》中的 <a href="#">创建您的首个 IAM 管理员用户和组</a> 的说明操作。	按照《IAM 用户指南》中的 <a href="#">管理 IAM 用户的访问密钥</a> ，配置程式访问。

## 后续步骤

创建 AWS 账户和 IAM 凭证后，要使用 AWS CLI，您可以执行下列操作之一：

- 在电脑上[安装最新版本](#)的 AWS CLI 版本 2。
- 在电脑上[安装过去发行版](#)的 AWS CLI 版本 2。
- [使用 Docker 镜像](#)从电脑访问 AWS CLI 版本 2。
- 使用 AWS CloudShell 通过浏览器访问 AWS 控制台中的 AWS CLI 版本 2。有关更多信息，请参阅[AWS CloudShell 用户指南](#)。

## 安装或更新到最新版本的 AWS CLI

本主题介绍如何在支持的操作系统上安装或更新最新版本的 AWS Command Line Interface (AWS CLI)。有关最新版本的信息 AWS CLI，请参阅[AWS CLI 版本 2 的变更日 GitHub 志](#)。

要安装过去版本的 AWS CLI，请参阅[the section called “过去发行版”](#)。有关卸载说明，请参阅[卸载](#)。

### Important

AWS CLI 版本 1 和 2 使用相同的aws命令名称。如果您之前安装了 AWS CLI 版本 1，请参阅[从 AWS CLI 版本 1 迁移至版本 2](#)。

### 主题

- [AWS CLI 安装和更新说明](#)
- [AWS CLI 安装和卸载错误疑难解答](#)
- [后续步骤](#)

## AWS CLI 安装和更新说明

有关安装说明，请展开适用于操作系统的部分。

## Linux

### 安装和更新要求

- 您必须能够提取或“解压缩”下载的程序包。如果您的操作系统没有内置的 `unzip` 命令，请使用等效命令。
- 用 AWS CLI 途 `glibc`、`groff`、和 `less`。默认情况下，它们都包含在 Linux 的大多数主要发行版中。
- 我们支持 CentOS、Fedora、Ubuntu、亚马逊 Linux 1、亚马逊 Linux 2、亚马逊 Linux 2023 和 Linux ARM 的最新发行版的 64 位版本。AWS CLI
- 由于 AWS 不维护第三方存储库，因此我们无法保证它们包含最新版本的 AWS CLI。

### 安装或更新 AWS CLI

#### Warning

如果这是您首次在 Amazon Linux 上进行更新，则要安装最新版本的 AWS CLI，则必须使用以下命令卸载预安装的 yum 版本：

```
$ sudo yum remove awscli
```

删除 yum 安装后，AWS CLI 请按照以下 Linux 安装说明进行操作。

要更新当前安装的 AWS CLI，请在每次更新时下载一个新的安装程序以覆盖以前的版本。按照命令行中的以下步骤 AWS CLI 在 Linux 上安装。

以下是单个复制和粘贴组中的快速安装步骤，具体取决于您使用的是提供基本安装的 64 位 Linux 还是 Linux ARM。有关指导说明，请参阅随后的步骤。

### Linux x86 (64-bit)

#### Note

( 可选 ) 以下命令块无需先验证下载内容的完整性即可下载和安装 AWS CLI。要验证下载内容的完整性，请使用以下分步说明。

要安装 AWS CLI，请运行以下命令。

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

要更新 AWS CLI 的当前安装，请添加现有的符号链接和安装程序信息，以使用 `--bin-dir`、`--install-dir` 和 `--update` 参数构造 `install` 命令。以下命令块使用示例符号链接 `/usr/local/bin` 和示例安装位置 `/usr/local/aws-cli`。

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --
update
```

## Linux ARM

### Note

(可选) 以下命令块无需先验证下载内容的完整性即可下载和安装 AWS CLI。要验证下载内容的完整性，请使用以下分步说明。

要安装 AWS CLI，请运行以下命令。

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-aarch64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

要更新 AWS CLI 的当前安装，请添加现有的符号链接和安装程序信息，以使用 `--bin-dir`、`--install-dir` 和 `--update` 参数构造 `install` 命令。以下命令块使用示例符号链接 `/usr/local/bin` 和示例安装位置 `/usr/local/aws-cli`。

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-aarch64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --
update
```



## 引导式安装步骤

### 1. 通过以下方式之一下载安装文件：

#### Linux x86 (64-bit)

- 使用 **curl** 命令 `--o` 选项指定下载的程序包应写入的文件名。以下示例命令中使用的选项将下载的文件以本地名称 `awscliv2.zip` 写入到当前目录中。

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o  
"awscliv2.zip"
```

- 从 URL 下载 – 要使用浏览器下载安装程序，请使用以下 URL：[https://awscli.amazonaws.com/awscli-exe-linux-x86\\_64.zip](https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip)

#### Linux ARM

- 使用 **curl** 命令 `--o` 选项指定下载的程序包应写入的文件名。以下示例命令中使用的选项将下载的文件以本地名称 `awscliv2.zip` 写入到当前目录中。

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-aarch64.zip" -o  
"awscliv2.zip"
```

- 从 URL 下载 – 要使用浏览器下载安装程序，请使用以下 URL：<https://awscli.amazonaws.com/awscli-exe-linux-aarch64.zip>

### 2. ( 可选 ) 验证下载的 zip 文件的完整性

如果您 `.zip` 在上述步骤中选择手动下载 AWS CLI 安装程序包，则可以使用以下步骤使用该 GnuPG 工具来验证签名。

AWS CLI 安装程序包 `.zip` 文件使用 PGP 签名进行加密签名。如果文件有任何损坏或更改，则此验证失败，您不应继续安装。

- a. 使用您的程序包管理器下载并安装 `gpg` 命令。有关 GnuPG 的更多信息，请参阅 [GnuPG 网站](#)。
- b. 要创建公有密钥文件，请创建一个文本文件并粘贴到以下文本中。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
mQINBF2Cr7UBEADJZHcgus0Jl7ENSyumXh85z0TRV0xJorM2B/JL0kH0yigQluUG
```

```
ZMLhENaG0bYatdrKP+3H911vK050pXwn0/R7fB/FSTouki4ciIx50uLlnJZIxSzx
PqG10mkxImLnbGwoi6Lto0LYxqHN2iQtzlwTVmq9733zd3XfcXrZ3+Lb1HAgEt5G
TfNxEKJ8soPLYWmwDH6HWcnjZ/aIQRBTIQ05uVeEoYxSh6w0ai7ss/KveoSNBbYz
gbdzoqI2Y8cgH2nbfpg3DSasaLZEdCSsIsK1u05CinE7k2qZ7KgKAUIcT/cR/grk
C6VwsnDU00UCideXcQ8WeHutqvgZH1JgKDbznoIzeQHJD238GEu+eKhRHcz8/jeG
94zkcgJ0z3KbZGYMiTh277Fvj9zzvZsbMBCedV1BTg3TqgvDX4bdkhf5cH+7NtW0
lrFj6UwAsGukBTA0xC01/dnSmZhJ7Z1KmEWilro/g0rjt0xqRQut1IqG22TaqoPG
fYVN+en3Zwbt97kcgZDwqbuykNt64oZwc4XKCa3mprEGC3IbJTBFqglXmZ719ywG
EEUJY01b2XrSuPwm139beWdKM8kzr10jn10m6+lpTRCBfo0wa9F8YZRhHPAkWkKX
XDe0GpWRj4oh0x0d2GWkyV5xyN14p2tQ0Cd00Dmz80yUTgRpPVQUt0EhXQARAQAB
tCFBV1MgQ0xJIFR1YW0gPGF3cy1jbG1AYW1hem9uLmNvbT6JAlQEewEIAD4CGwMF
CwkIBwIGFQoJCAcCBByCAwECHgECF4AWIQT7Xbd/1cEYuAURraimMQrMRnJHXAUc
ZMKcEgUJCSEf3QAKCRCmMQrMRnJHXCIID/4vior9J5tB+icri5WbDudS3ak/ve4q
XS6ZLm5S81+CBxy5aLQUlyFhuuaEHDC11fG780duxatzeHENASYVo3mmKNwrCBza
NJaeaWKLGT0MKwBSP5aa3dva8P/4oUP9GsQn0uWoXwNDwfrMbNI8gn+jC/3MigW
vD3fu6zC0WWLITNv2SJoQ1wILmb/uGfha68o4iTB0vcftVRua06DyqF+CrHX/0j0
k1EDQFMY9M4tsYT7X8NwfI8Vmc89nzpVL9fwd44WwpKIw1FBZP8S0sgDx2xDsxv
L8kM2Gt0iH0cHqF0+V7xtTKZyloLiDbJKhu80Kc+YC/TmozD8oeGU2rEfxFLegwS
zT9N+jB38+dqaP9pRDSi45iGqyA8yavVBabPL0IQ9jU6eIV+kmcjIjcun/Uo8SjJ
0xQAsm41rxPaKV6vJUn10wVnuhSkKk8mzN01SZwu7Hua6rdcCaGeB8uJ44AP3QzW
BNnrjtoN6A1N0D2wFmFE/YL/rHPxU1XwPntubYB/t3rXFL7ENQ00QH0KVXgRC1ey
sHMglg46c+nQLRzVTshjDjmtzv9rcV9RKR0PetEggzCoD89veDA9jPR2Kw6RYkS
XzYm2fEv16/HRNYt7hJzneFqRIjHW5qAgSs/bcaRWpAU/QQzzJPVKCQNr4y0weyg
B8HctGjfod0p1A==
=gdMc
-----END PGP PUBLIC KEY BLOCK-----
```

以下是公有密钥的详细信息以供参考。

```
Key ID:          A6310ACC4672475C
Type:           RSA
Size:          4096/4096
Created:       2019-09-18
Expires:      2024-07-26
User ID:      AWS CLI Team <aws-cli@amazon.com>
Key fingerprint: FB5D B77F D5C1 18B8 0511 ADA8 A631 0ACC 4672 475C
```

- c. 使用以下命令导入 AWS CLI 公钥，*public-key-file-name* 替换为您创建的公钥的文件名。

```
$ gpg --import public-key-file-name
gpg: /home/username/.gnupg/trustdb.gpg: trustdb created
```

```
gpg: key A6310ACC4672475C: public key "AWS CLI Team <aws-cli@amazon.com>"
imported
gpg: Total number processed: 1
gpg:             imported: 1
```

- d. 下载您下载的软件包的 AWS CLI 签名文件。它具有与其对应的 .zip 文件相同的路径和名称，但扩展名为 .sig。在以下示例中，我们将其作为名为 awscliv2.sig 的文件保存到当前目录中。

#### Linux x86 (64-bit)

要获取最新版本的 AWS CLI，请使用以下命令块：

```
$ curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip.sig
```

对于的特定版本 AWS CLI，请在文件名后面附加连字符和版本号。在本示例中，版本 **2.0.30** 的文件名为 awscli-exe-linux-x86\_64-2.0.30.zip.sig，这会生成以下命令：

```
$ curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-x86_64-2.0.30.zip.sig
```

如需查看版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub

#### Linux ARM

要获取最新版本的 AWS CLI，请使用以下命令块：

```
$ curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-aarch64.zip.sig
```

对于的特定版本 AWS CLI，请在文件名后面附加连字符和版本号。在本示例中，版本 **2.0.30** 的文件名为 awscli-exe-linux-aarch64-2.0.30.zip.sig，这会生成以下命令：

```
$ curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-aarch64-2.0.30.zip.sig
```

如需查看版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub

- e. 验证签名，并将下载的 `.sig` 和 `.zip` 文件名作为参数传递给 `gpg` 命令。

```
$ gpg --verify awscliv2.sig awscliv2.zip
```

该输出值应该类似于以下内容。

```
gpg: Signature made Mon Nov  4 19:00:01 2019 PST
gpg:                using RSA key FB5D B77F D5C1 18B8 0511 ADA8 A631 0ACC 4672
 475C
gpg: Good signature from "AWS CLI Team <aws-cli@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:                There is no indication that the signature belongs to the owner.
Primary key fingerprint: FB5D B77F D5C1 18B8 0511 ADA8 A631 0ACC 4672 475C
```

#### Important

输出中的警告是预料中的，并不表示出现了问题。之所以发生这种情况，是因为您的个人 PGP 密钥（如果有的话）和 PGP 密钥之间没有信任链。AWS CLI 有关更多信息，请参阅[信任 Web](#)。

3. 解压缩安装程序。如果您的 Linux 发行版没有的内置 `unzip` 命令，请使用等效的命令对其解压缩。以下示例命令解压缩程序包并在当前目录下创建一个名为 `aws` 的目录。

```
$ unzip awscliv2.zip
```

#### Note

从以前的版本进行更新时，`unzip` 命令会提示覆盖现有文件。要跳过这些提示，例如使用脚本自动化，请为 `unzip` 使用 `-u` 更新。此标志会自动更新现有文件并根据需要创建新文件。

```
$ unzip -u awscliv2.zip
```

4. 运行安装程序。安装命令在新解压缩的 `install` 目录中使用一个名为 `aws` 的文件。默认情况下，文件全部安装到 `/usr/local/aws-cli` 并在 `/usr/local/bin` 中创建符号链接。该命令包括 `sudo` 以授予对这些目录的写入权限。

```
$ sudo ./aws/install
```

如果您指定您已经具有写入权限的目录，则可以不带 `sudo` 进行安装。对于 `install` 命令，请按以下说明指定安装位置：

- 确保您提供给 `-i` 和 `-b` 参数的路径不包含任何含空格字符或其他空白字符的卷名称或目录名称。如果有空格，安装将失败。
- `--install-dir` 或 `-i` – 此选项指定要将所有文件复制到的目录。

原定设置值为 `/usr/local/aws-cli`。

- `--bin-dir` 或 `-b` – 此选项指定安装目录中的主 `aws` 程序通过符号链接指向指定路径中的 `aws` 文件。您必须具有对指定目录的写入权限。创建指向路径中已经存在的目录的符号链接后，无需再将安装目录添加到用户的 `$PATH` 变量中。

原定设置值为 `/usr/local/bin`。

```
$ ./aws/install -i /usr/local/aws-cli -b /usr/local/bin
```

### Note

要更新当前安装的 AWS CLI，请添加现有的符号链接和安装程序信息，以使用 `--update` 参数构造 `install` 命令。

```
$ sudo ./aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --update
```

要查找现有的符号链接和安装目录，请执行以下步骤：

1. 使用 `which` 命令查找符号链接。这为您提供了与 `--bin-dir` 参数一起使用的路径。

```
$ which aws
/usr/local/bin/aws
```

2. 使用 `ls` 命令查找符号链接指向的目录。这为您提供了与 `--install-dir` 参数一起使用的路径。

```
$ ls -l /usr/local/bin/aws
```

```
lrwxrwxrwx 1 ec2-user ec2-user 49 Oct 22 09:49 /usr/local/bin/aws -> /usr/local/aws-cli/v2/current/bin/aws
```

5. 通过以下命令确认此安装。

```
$ aws --version  
aws-cli/2.15.30 Python/3.11.6 Linux/5.10.205-195.807.amzn2.x86_64 botocore/2.4.5
```

如果无法找到 `aws` 命令，您可能需要重新启动终端或按照[排查错误](#)中的说明排除故障。

## macOS

### 安装和更新要求

- 我们支持 mac AWS CLI OS 版本 10.9 及更高版本。如需了解更多信息，请参阅AWS 开发者工具博客上的[macOS AWS CLI v2 支持政策更新](#)。
- 由于 AWS 不维护第三方存储库，因此我们无法保证它们包含最新版本的 AWS CLI。

### 安装或更新 AWS CLI

如果要更新到最新版本，请使用与当前版本相同的安装方法。您可以通过以下方式 AWS CLI 在 macOS 上安装。

### GUI installer

以下步骤说明如何使用标准 macOS 用户界面和浏览器安装最新版本的 AWS CLI

1. 在浏览器中，下载 macOS pkg 文件：<https://awscli.amazonaws.com/AWSCLIV2.pkg>
2. 运行下载的文件，然后按照屏幕上的说明进行操作。您可以选择通过以下 AWS CLI 方式安装：
  - 针对电脑上的所有用户（需要 **sudo**）
    - 您可以安装到任何文件夹，或选择推荐的默认文件夹 `/usr/local/aws-cli`。
    - 安装程序会自动在 `/usr/local/bin/aws` 中创建一个符号链接，该链接指向您选择的安装文件夹中的主程序。
  - 仅针对当前用户（不需要 **sudo**）
    - 您可以安装到具有写入权限的任何文件夹。

- 由于具有标准用户权限，在安装程序完成后，您必须在命令提示符下使用以下命令在 \$PATH 中手动创建指向 aws 和 aws\_completer 程序的符号链接文件。如果 \$PATH 包含您可以写入的文件夹，并且您将该文件夹指定为目标路径，则可以在不使用 sudo 的情况下运行以下命令。如果 \$PATH 中没有可写入的文件夹，您必须在命令中使用 sudo，以获取写入指定目标文件夹的权限。符号链接的默认位置为 /usr/local/bin/。

```
$ sudo ln -s /folder/installed/aws-cli/aws /usr/local/bin/aws
$ sudo ln -s /folder/installed/aws-cli/aws_completer /usr/local/bin/
aws_completer
```

### Note

您可以在安装过程中的任意时刻按 Cmd+L 来查看安装的调试日志。这会打开一个日志窗格，让您能够筛选和保存日志。日志文件还会自动保存到 /var/log/install.log。

3. 要验证 Shell 是否可以在 \$PATH 中找到并运行 aws 命令，请使用以下命令。

```
$ which aws
/usr/local/bin/aws
$ aws --version
aws-cli/2.15.30 Python/3.11.6 Darwin/23.3.0 botocore/2.4.5
```

如果无法找到 aws 命令，您可能需要重新启动终端或按照[排查错误](#)中的说明排除故障。

## Command line installer - All users

如果您拥有 sudo 权限，则可以针对电脑上的所有用户安装 AWS CLI。我们在一个易于复制和粘贴的组中提供步骤。请参阅以下步骤中每一行的描述。

```
$ curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
$ sudo installer -pkg AWSCLIV2.pkg -target /
```

## 引导式安装说明

1. 使用 curl 命令下载文件。-o 选项指定下载的程序包应写入的文件名。在此示例中，文件写入当前文件夹中的 AWSCLIV2.pkg。

```
$ curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
```

2. 运行标准 macOS installer 程序，同时将下载的 .pkg 文件指定为源。使用 -pkg 参数指定要安装的程序包的名称以及要将程序包安装到的驱动器的 -target / 参数。系统会将文件安装到 /usr/local/aws-cli，并自动在 /usr/local/bin 中创建符号链接。您必须在命令中使用 sudo 来授予对这些文件夹的写入权限。

```
$ sudo installer -pkg ./AWSCLIV2.pkg -target /
```

安装完成后，调试日志将写入到 /var/log/install.log。

3. 要验证 Shell 是否可以在 \$PATH 中找到并运行 aws 命令，请使用以下命令。

```
$ which aws
/usr/local/bin/aws
$ aws --version
aws-cli/2.15.30 Python/3.11.6 Darwin/23.3.0 botocore/2.4.5
```

如果无法找到 aws 命令，您可能需要重新启动终端或按照[排查错误](#)中的说明排除故障。

## Command line - Current user

1. 要指定安装到哪个文件夹，必须使用任意文件名创建一个 XML 文件。AWS CLI 此文件是 XML 格式的文件，类似于以下示例。保持所显示的所有值不变，但必须将第 9 行中的 */Users/myusername* 路径替换为您要在其中安装 AWS CLI 的文件夹的路径。必须存在该文件夹，否则命令会失败。以下 XML 示例名为 *choices.xml*，它指定了要在文件夹 AWS CLI 中安装的安装程序/Users/myusername，并在该文件夹中创建了一个名为的文件夹aws-cli。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <array>
    <dict>
      <key>choiceAttribute</key>
      <string>customLocation</string>
      <key>attributeSetting</key>
      <string>/Users/myusername</string>
```



```
<key>choiceIdentifier</key>
  <string>default</string>
</dict>
</array>
</plist>
```

2. 使用 `curl` 命令下载 `pkg` 安装程序。`-o` 选项指定下载的程序包应写入的文件名。在此示例中，文件写入当前文件夹中的 `AWSCLIV2.pkg`。

```
$ curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
```

3. 使用以下选项运行标准 macOS installer 程序：

- 使用 `-pkg` 参数指定要安装的程序包的名称。
- 通过将 `-target` 参数设置为 `CurrentUserHomeDirectory`，指定仅安装到当前用户。
- 在 `-applyChoiceChangesXML` 参数中指定您创建的 XML 文件的路径（相对于当前文件夹）和名称。

以下示例将安装 AWS CLI 在文件夹中 `/Users/myusername/aws-cli`。

```
$ installer -pkg AWSCLIV2.pkg \
            -target CurrentUserHomeDirectory \
            -applyChoiceChangesXML choices.xml
```

4. 由于标准用户权限通常不允许写入 `$PATH` 中的文件夹，因此在此模式下，安装程序不会尝试将符号链接添加到 `aws` 和 `aws_completer` 程序中。AWS CLI 要使正常运行，您必须在安装程序完成后手动创建符号链接。如果 `$PATH` 包含您可以写入的文件夹，并且您将该文件夹指定为目标的路径，则可以在不使用 `sudo` 的情况下运行以下命令。如果 `$PATH` 中没有可写入的文件夹，您必须使用 `sudo`，以获取写入指定目标文件夹的权限。符号链接的默认位置为 `/usr/local/bin/`。将 `folder/installed` 替换为您的 AWS CLI 安装的路径。

```
$ sudo ln -s /folder/installed/aws-cli/aws /usr/local/bin/aws
$ sudo ln -s /folder/installed/aws-cli/aws_completer /usr/local/bin/
aws_completer
```

安装完成后，调试日志将写入到 `/var/log/install.log`。

5. 要验证 Shell 是否可以在 `$PATH` 中找到并运行 `aws` 命令，请使用以下命令。

```
$ which aws
```

```
/usr/local/bin/aws
$ aws --version
aws-cli/2.15.30 Python/3.11.6 Darwin/23.3.0 botocore/2.4.5
```

如果无法找到 `aws` 命令，您可能需要重新启动终端或按照[排查错误](#)中的说明排除故障。

## Windows

### 安装和更新要求

- 我们支持微软支持 AWS CLI 的 64 位 Windows 版本。
- 安装软件的管理员权限

### 安装或更新 AWS CLI

要 AWS CLI 在 Windows 上更新当前安装的，请在每次更新时下载一个新的安装程序以覆盖以前的版本。AWS CLI 会定期更新。要查看最新版本的发布时间，请查看[AWS CLI 版本 2 的变更日志](#)。

1. 下载并运行适用于 Windows 的 AWS CLI MSI 安装程序（64 位）：

<https://awscli.amazonaws.com/AWSCLIV2.msi>

或者，您可以运行 `msiexec` 命令来运行 MSI 安装程序。

```
C:\> msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
```

有关可与 `msiexec` 一起使用的各种参数，请参阅 Microsoft Docs 网站上的 [msiexec](#)。例如，您可以使用 `/qn` 标记进行静默安装。

```
C:\> msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi /qn
```

2. 要确认安装，请打开 Start（开始）菜单，搜索 `cmd` 以打开命令提示符窗口，然后在命令提示符处使用 `aws --version` 命令。

```
C:\> aws --version
aws-cli/2.15.30 Python/3.11.6 Windows/10 exe/AMD64 prompt/off
```

如果 Windows 无法找到该程序，您可能需要关闭并重新打开命令提示符窗口以刷新该路径，按照[排查错误](#)中的说明排除故障。

## AWS CLI 安装和卸载错误疑难解答

如果您在安装或卸载后遇到问题 AWS CLI，[排查错误](#)请参阅，了解故障排除步骤。有关相关性最高的故障排除步骤，请参阅[the section called “找不到命令错误”](#)、[the section called ““aws --version”命令返回的版本与您安装的版本不同”](#)和[the section called “卸载后，aws --version”命令会返回一个版本 AWS CLI”](#)。

### 后续步骤

成功安装后 AWS CLI，您可以安全地删除已下载的安装程序文件。完成中的步骤[the section called “先决条件”](#)并安装后 AWS CLI，应执行[the section called “设置”](#)。

## 安装版本 2 的过去 AWS CLI 版本

本主题介绍如何在支持的操作系统上安装 AWS Command Line Interface 版本 2 (AWS CLI) 的先前版本。有关 AWS CLI 版本 2 版本的信息，请参阅[AWS CLI 版本 2 变更日 GitHub 志](#)。

AWS CLI 版本 2 安装说明：

### Linux

#### 安装要求

- 您知道要安装第 2 AWS CLI 版的哪个版本。有关版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub
- 您必须能够提取或“解压缩”下载的程序包。如果您的操作系统没有内置的 unzip 命令，请使用等效命令。
- AWS CLI 版本 2 使用glibcgroff、和less。默认情况下，它们都包含在 Linux 的大多数主要发行版中。
- 我们在最近发行 AWS CLI 的 CentOS、Fedora、Ubuntu、亚马逊 Linux 1、亚马逊 Linux 2 和 Linux ARM 的 64 位版本上支持第 2 版。
- 由于 AWS 不维护第三方存储库，因此我们无法保证它们包含最新版本的 AWS CLI。

## 安装说明

按照命令行中的以下步骤 AWS CLI 在 Linux 上安装。

我们根据您是使用 64 位 Linux 还是 Linux ARM，在一个易于复制和粘贴的组中提供步骤。请参阅以下步骤中每一行的描述。

Linux x86 (64-bit)

### Note

( 可选 ) 以下命令块 AWS CLI 无需先验证下载内容的完整性即可下载和安装。要验证下载内容的完整性，请使用以下分步说明。

有关版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub

要安装 AWS CLI，请运行以下命令。

要指定一个版本，请在文件名后附加一个连字符和版本号。在本示例中，版本 **2.0.30** 的文件名为 `awscli-exe-linux-x86_64-2.0.30.zip`，这会生成以下命令：

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64-2.0.30.zip" -o
awscliv2.zip
unzip awscliv2.zip
sudo ./aws/install
```

要更新 AWS CLI 的当前安装，请添加现有的符号链接和安装程序信息，以使用 `--bin-dir`、`--install-dir` 和 `--update` 参数构造 `install` 命令。以下命令块使用示例符号链接 `/usr/local/bin` 和示例安装位置 `/usr/local/aws-cli`。

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64-2.0.30.zip" -o
awscliv2.zip
unzip awscliv2.zip
sudo ./aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --
update
```

## Linux ARM

### Note

( 可选 ) 以下命令块 AWS CLI 无需先验证下载内容的完整性即可下载和安装。要验证下载内容的完整性，请使用以下分步说明。

有关版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub

要安装 AWS CLI，请运行以下命令。

要指定一个版本，请在文件名后附加一个连字符和版本号。在本示例中，版本 **2.0.30** 的文件名为 `awscli-exe-linux-aarch64-2.0.30.zip`，这会生成以下命令：

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-aarch64-2.0.30.zip" -o
  "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

要更新 AWS CLI 的当前安装，请添加现有的符号链接和安装程序信息，以使用 `--bin-dir`、`--install-dir` 和 `--update` 参数构造 `install` 命令。以下命令块使用示例符号链接 `/usr/local/bin` 和示例安装位置 `/usr/local/aws-cli`。

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-aarch64-2.0.30.zip" -o
  "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --
update
```

### 1. 通过以下方式之一下载安装文件：

#### Linux x86 (64-bit)

- 使用 `curl` 命令 `--o` 选项指定下载的程序包应写入的文件名。以下示例命令中使用的选项将下载的文件以本地名称 `awscliv2.zip` 写入到当前目录中。

要指定一个版本，请在文件名后附加一个连字符和版本号。在本示例中，版本 **2.0.30** 的文件名为 `awscli-exe-linux-x86_64-2.0.30.zip`，这会生成以下命令：

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64-2.0.30.zip" -o  
"awscliv2.zip"
```

有关版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub

- 从 URL 下载 –

在您的浏览器中，AWS CLI 通过在文件名后面附加连字符和版本号来下载您的特定版本。

```
https://awscli.amazonaws.com/awscli-exe-linux-x86_64-version.number.zip
```

在此示例中，版本 *2.0.30* ##### *awscli-exe-linux-x86\_64-2.0.30*.zip，结果为以下链接：[https://awscli.amazonaws.com/awscli-exe-linux-x86\\_64-2.0.30.zip](https://awscli.amazonaws.com/awscli-exe-linux-x86_64-2.0.30.zip)

## Linux ARM

- 使用 **curl** 命令 – **-o** 选项指定下载的程序包应写入的文件名。以下示例命令中使用的选项将下载的文件以本地名称 `awscliv2.zip` 写入到当前目录中。

要指定一个版本，请在文件名后附加一个连字符和版本号。在本示例中，版本 *2.0.30* 的文件名为 `awscli-exe-linux-aarch64-2.0.30.zip`，这会生成以下命令：

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-aarch64-2.0.30.zip" -o  
"awscliv2.zip"  
unzip awscliv2.zip  
sudo ./aws/install
```

- 从 URL 下载 –

在您的浏览器中，AWS CLI 通过在文件名后面附加连字符和版本号来下载您的特定版本。

```
https://awscli.amazonaws.com/awscli-exe-linux-aarch64-version.number.zip
```

在本示例中，版本 *2.0.30* 的文件名为 `awscli-exe-linux-aarch64-2.0.30.zip`，这会生成以下链接：<https://awscli.amazonaws.com/awscli-exe-linux-aarch64-2.0.30.zip>

## 2. (可选) 验证下载的 zip 文件的完整性

如果您 `.zip` 在上述步骤中选择手动下载 AWS CLI 安装程序包，则可以使用以下步骤使用该 GnuPG 工具来验证签名。

AWS CLI 安装程序包 .zip 文件使用 PGP 签名进行加密签名。如果文件有任何损坏或更改，则此验证失败，您不应继续安装。

- a. 使用您的程序包管理器下载并安装 `gpg` 命令。有关 GnuPG 的更多信息，请参阅 [GnuPG 网站](#)。
- b. 要创建公有密钥文件，请创建一个文本文件并粘贴到以下文本中。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBF2Cr7UBEADJZHcgus0Jl7ENSyUmXh85z0TRV0xJorM2B/JL0kH0yigQ1uUG
ZMLhEnAG0bYatdrKP+3H911vK050pXwn0/R7fB/FSTouki4ciIx50uLlnJZIxSzx
PqG10mkxImLnbGwoi6Lto0LYxqHN2iQtzlwTVmq9733zd3XfcXrZ3+Lb1HAgEt5G
TfnxEKJ8soPlyWmWdH6HWcnjZ/aIQRBTIQ05uVeEoYxSh6w0ai7ss/KveoSNBbYz
gbdzoqI2Y8cgH2bnfgp3DSasaLZEdCSsIsK1u05CinE7k2qZ7KgKAUIcT/cR/grk
C6VwsnDU00UCideXcQ8WeHutqvgZH1JgKDbznoIzeQHJD238GEu+eKhRHcz8/jeG
94zkcqJ0z3KbZGYMiTh277Fvj9zzvZsbMBCedV1BTg3TqgvdX4bdkhf5cH+7NtW0
lrFj6UwAsGukBTA0xC0l/dnSmZhJ7Z1KmEWilro/g0rjt0xqRQut1IqG22TaqoPG
fYVN+en3ZwbT97kcgZDwqbuykNt64oZwC4XKCa3mprEGC3IbJTBFqg1XmZ719ywG
EEUJY01b2XrSuPwm139beWdKM8kzr10jnl0m6+lpTRCBfo0wa9F8YZRhHPAkWkKX
XDe0GpWrj4oh0x0d2GWkyV5xyN14p2tQ0Cd00Dmz80yUTgRpPVQUt0EHXQARAQAB
tCFBV1MgQ0xJIFRlYW0gPGF3cy1jbG1AYW1hem9uLmNvbT6JAlQEEwEiAD4CGwMF
CwkIBwIGFQoJCAwCBByCAwECHgECF4AWIQT7Xbd/1cEYuAURraimMQrMRnJHXAUC
ZMKcEgUJCSEf3QAKCRCmQrMRnJHXCiLD/4vior9J5tB+icri5WbDudS3ak/ve4q
XS6ZLm5S8l+CBxy5aLQUlyFhuuaEHDC11fG780duxatzeHENASYVo3mmKNwrCBza
NJaeaWKLGT0MKwBSP5aa3dva8P/4oUP9GsQn0uWoXwNDWfrMbNI8gn+jC/3MigW
vD3fu6zC0WwLITNv2SJoQ1wILmb/uGfha68o4iTB0vcftVRua06DyqF+CrHX/0j0
k1EDQFMY9M4tsYT7X8NwfI8Vmc89nzpVl9fwda44WwpKIw1FBZP8S0sgDx2xDsxv
L8kM2Gt0iH0cHqF0+V7xtTKZylo1iDbJKhu80Kc+YC/TmozD8oeGU2rEfxFLegwS
zT9N+jB38+dqaP9pRdsi45iGqyA8yavVBabpL0IQ9jU6eIV+kmcjIjcun/Uo8SjJ
0xQAsm41rxPaKV6vJU10wVNuhSkKk8mzN01SZwu7Hua6rdcCaGeB8uJ44AP3QzW
BNnrjtoN6A1N0D2wFmfE/YL/rHPxU1XwPntubYB/t3rXFL7ENQ00QH0KVXgRCley
sHMglg46c+nQLRzVTshjDjmtzv9rcV9RKR0PetEggzCoD89veDA9jPR2Kw6RYkS
XzYm2fEv16/HRNYt7hJzneFqRIjHW5qAgSs/bcaRWpAU/QQzzJpVKCQNr4y0weyg
B8HCtGjfod0p1A==
=gdMc
-----END PGP PUBLIC KEY BLOCK-----
```

以下是公有密钥的详细信息以供参考。

```
Key ID:          A6310ACC4672
Type:           RSA
```

```
Size:                4096/4096
Created:             2019-09-18
Expires:            2024-07-26
User ID:            AWS CLI Team <aws-cli@amazon.com>
Key fingerprint:   FB5D B77F D5C1 18B8 0511 ADA8 A631 0ACC 4672 475C
```

- c. 使用以下命令导入 AWS CLI 公钥，*public-key-file-name* 替换为您创建的公钥的文件名。

```
$ gpg --import public-key-file-name
gpg: /home/username/.gnupg/trustdb.gpg: trustdb created
gpg: key A6310ACC4672475C: public key "AWS CLI Team <aws-cli@amazon.com>"
imported
gpg: Total number processed: 1
gpg:                imported: 1
```

- d. 下载您下载的软件包的 AWS CLI 签名文件。它具有与其对应的 .zip 文件相同的路径和名称，但扩展名为 .sig。在以下示例中，我们将其作为名为 awscliv2.sig 的文件保存到当前目录中。

Linux x86 (64-bit)

要获取最新版本的 AWS CLI，请使用以下命令块：

```
$ curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip.sig
```

对于的特定版本 AWS CLI，请在文件名后面附加连字符和版本号。在本示例中，版本 **2.0.30** 的文件名为 awscli-exe-linux-x86\_64-2.0.30.zip.sig，这会生成以下命令：

```
$ curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-x86_64-2.0.30.zip.sig
```

有关版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub

Linux ARM

要获取最新版本的 AWS CLI，请使用以下命令块：



```
$ curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-aarch64.zip.sig
```

对于的特定版本 AWS CLI，请在文件名后面附加连字符和版本号。在本示例中，版本 **2.0.30** 的文件名为 `awscli-exe-linux-aarch64-2.0.30.zip.sig`，这会生成以下命令：

```
$ curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-aarch64-2.0.30.zip.sig
```

有关版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub

- e. 验证签名，并将下载的 `.sig` 和 `.zip` 文件名作为参数传递给 `gpg` 命令。

```
$ gpg --verify awscliv2.sig awscliv2.zip
```

该输出值应该类似于以下内容。

```
gpg: Signature made Mon Nov  4 19:00:01 2019 PST
gpg:                using RSA key FB5D B77F D5C1 18B8 0511 ADA8 A631 0ACC 4672
475C
gpg: Good signature from "AWS CLI Team <aws-cli@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:                There is no indication that the signature belongs to the owner.
Primary key fingerprint: FB5D B77F D5C1 18B8 0511 ADA8 A631 0ACC 4672 475C
```

#### Important

输出中的警告是预料中的，并不表示出现了问题。之所以出现此警告，是因为您的个人 PGP 密钥（如果您有）和 AWS CLI PGP 密钥之间没有信任链。有关更多信息，请参阅[信任 Web](#)。

3. 解压缩安装程序。如果您的 Linux 发行版没有的内置 `unzip` 命令，请使用等效的命令对其解压缩。以下示例命令解压缩程序包并在当前目录下创建一个名为 `aws` 的目录。

```
$ unzip awscliv2.zip
```

4. 运行安装程序。安装命令在新解压缩的 `install` 目录中使用一个名为 `aws` 的文件。默认情况下，文件全部安装到 `/usr/local/aws-cli` 并在 `/usr/local/bin` 中创建符号链接。该命令包括 `sudo` 以授予对这些目录的写入权限。

```
$ sudo ./aws/install
```

如果您指定您已经具有写入权限的目录，则可以不带 `sudo` 进行安装。对于 `install` 命令，请按以下说明指定安装位置：

- 确保您提供给 `-i` 和 `-b` 参数的路径不包含任何含空格字符或其他空白字符的卷名称或目录名称。如果有空格，安装将失败。
- `--install-dir` 或 `-i` – 此选项指定要将所有文件复制到的目录。

原定设置值为 `/usr/local/aws-cli`。

- `--bin-dir` 或 `-b` – 此选项指定安装目录中的主 `aws` 程序通过符号链接指向指定路径中的 `aws` 文件。您必须具有对指定目录的写入权限。创建指向路径中已经存在的目录的符号链接后，无需再将安装目录添加到用户的 `$PATH` 变量中。

原定设置值为 `/usr/local/bin`。

```
$ ./aws/install -i /usr/local/aws-cli -b /usr/local/bin
```

#### Note

要将当前安装的 AWS CLI 版本 2 更新到新版本，请添加现有的符号链接和安装程序信息，以使用 `--update` 参数构造 `install` 命令。

```
$ sudo ./aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --update
```

要查找现有的符号链接和安装目录，请执行以下步骤：

1. 使用 `which` 命令查找符号链接。这为您提供了与 `--bin-dir` 参数一起使用的路径。

```
$ which aws
/usr/local/bin/aws
```

2. 使用 `ls` 命令查找符号链接指向的目录。这为您提供了与 `--install-dir` 参数一起使用的路径。

```
$ ls -l /usr/local/bin/aws
lrwxrwxrwx 1 ec2-user ec2-user 49 Oct 22 09:49 /usr/local/bin/aws -> /usr/
local/aws-cli/v2/current/bin/aws
```

5. 通过以下命令确认此安装。

```
$ aws --version
aws-cli/2.15.30 Python/3.11.6 Linux/5.10.205-195.807.amzn2.x86_64 botocore/2.4.5
```

如果无法找到 `aws` 命令，您可能需要重新启动终端或按照[排查错误](#)中的说明排除故障。

( 可选 ) 验证下载的 zip 文件的完整性

如果您 .zip 在上述步骤中选择手动下载 AWS CLI 版本 2 安装程序包，则可以使用以下步骤使用该 GnuPG 工具来验证签名。

AWS CLI 版本 2 安装程序包 .zip 文件使用 PGP 签名进行加密签名。如果文件有任何损坏或更改，则此验证失败，您不应继续安装。

1. 使用您的程序包管理器下载并安装 `gpg` 命令。有关 GnuPG 的更多信息，请参阅 [GnuPG 网站](#)。
2. 要创建公有密钥文件，请创建一个文本文件并粘贴到以下文本中。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
mQINBF2Cr7UBEADJZHcgus0Jl7ENSyumXh85z0TRV0xJorM2B/JL0kH0yigQ1uUG
ZMLhENaG0bYatdrKP+3H911vK050pXwn0/R7fB/FSTouki4ciIx50uLlnJZIxSzx
PqG10mkxImLnbGwoi6Lto0LYxqHN2iQtzlwTVmq9733zd3XfcXrZ3+Lb1HAgEt5G
TfnxEKJ8soPLyWmWDH6HWCnjZ/aIQRBTIQ05uVeEoYxSh6w0ai7ss/KveoSNBbYz
gbdzoqI2Y8cgH2nbfgp3DSasaLZEdCSsIsK1u05CinE7k2qZ7KgKAUIcT/cR/grk
C6VwsnDU00UCideXcQ8WeHutqvgZH1JgKDbznoIzeQHJD238GEu+eKhRHcz8/jeG
94zkcgJ0z3KbZGYMiTh277Fvj9zzvZsbMBCedV1BTg3TqgvdX4bdkhf5cH+7NtW0
lrFj6UwAsGukBTA0xC01/dnSmZhJ7Z1KmEWilro/g0rjt0xqRQut1IqG22TaqoPG
fYVN+en3Zwbt97kcgZDwqbuykNt64oZwc4XKCa3mprEGC3IbJTBFqglXmZ719ywG
EEUJY01b2XrSuPwml39beWdKM8kzr10jnl0m6+lpTRCBfo0wa9F8YZRhHPAkWkKX
XDe0GpWRj4oh0x0d2GwkyV5xyN14p2tQ0Cd00Dmz80yUTgRpPVQUt0EhXQARAQAB
tCFBV1MgQ0xJIFRlYW0gPGF3cy1jbG1AYW1hem9uLmNvbT6JAlQEEwEIAAD4CGwMF
CwkIBwIGFQoJCAAsCBBYCAwECHgECF4AWIQT7Xbd/1cEYuAURraimMQrMRnJHXAUC
```

```
ZMKcEgUJCSEf3QAKCRCmMQrMRnJHXCiLD/4vior9J5tB+icri5WbDudS3ak/ve4q
XS6ZLm5S81+CBxy5aLQUlyFhuuaEHDC11fG780duxatzeHENASYVo3mmKNwrCBza
NJaeaWKLGT0MKwBSP5aa3dva8P/4oUP9GsQn0uWoXwNDWfrMbNI8gn+jC/3MigW
vD3fu6zC0WWLITNv2SJoQlWILmb/uGfha68o4iTB0vcftVRuao6DyqF+CrHX/0j0
k1EDQFMY9M4tsYT7X8NwfI8Vmc89nzpvL9fwda44WwpKIw1FBZP8S0sgDx2xDsxv
L8kM2Gt0iH0cHqF0+V7xtTKZyloIiDbJKhu80Kc+YC/TmozD8oeGU2rEFXfLegwS
zT9N+jB38+dqaP9pRDsi45iGqyA8yavVBabpL0IQ9jU6eIV+kmcjIjcun/Uo8SjJ
0xQAsm41rxPaKV6vJUn10wVnuhSkKk8mzN01SZwu7Hua6rdcCaGeB8uJ44AP3QzW
BNnrjtoN6A1N0D2wFmfE/YL/rHPxU1XwPntubYB/t3rXFL7ENQ00QH0KVXgRC1ey
sHMglg46c+nQLRzVTshjDjmtzvh9rcV9RKR0PetEggzCoD89veDA9jPR2Kw6RYkS
XzYm2fEv16/HRNYt7hJzneFqRIjHW5qAgSs/bcaRwPAU/QQzzJPVKCQNr4y0weyg
B8HCtGjfod0p1A==
=gdMc
-----END PGP PUBLIC KEY BLOCK-----
```

以下是公有密钥的详细信息以供参考。

```
Key ID:           A6310ACC4672
Type:             RSA
Size:            4096/4096
Created:         2019-09-18
Expires:        2024-07-26
User ID:        AWS CLI Team <aws-cli@amazon.com>
Key fingerprint: FB5D B77F D5C1 18B8 0511 ADA8 A631 0ACC 4672 475C
```

- 使用以下命令导入 AWS CLI 公钥，*public-key-file-name* 替换为您创建的公钥的文件名。

```
$ gpg --import public-key-file-name
gpg: /home/username/.gnupg/trustdb.gpg: trustdb created
gpg: key A6310ACC4672475C: public key "AWS CLI Team <aws-cli@amazon.com>" imported
gpg: Total number processed: 1
gpg:             imported: 1
```

- 下载您下载的软件包的 AWS CLI 签名文件。它具有与其对应的 .zip 文件相同的路径和名称，但扩展名为 .sig。在以下示例中，我们将其作为名为 *awscliv2.sig* 的文件保存到当前目录中。

Linux x86 (64-bit)

对于最新版本的 AWS CLI，请使用以下命令块：

```
$ curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip.sig
```

对于特定版本的 AWS CLI，在文件名后附加一个连字符和版本号。在本示例中，版本 **2.0.30** 的文件名为 `awscli-exe-linux-x86_64-2.0.30.zip.sig`，这会生成以下命令：

```
$ curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-x86_64-2.0.30.zip.sig
```

有关版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub

## Linux ARM

要获取最新版本的 AWS CLI，请使用以下命令块：

```
$ curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-aarch64.zip.sig
```

对于的特定版本 AWS CLI，请在文件名后面附加连字符和版本号。在本示例中，版本 **2.0.30** 的文件名为 `awscli-exe-linux-aarch64-2.0.30.zip.sig`，这会生成以下命令：

```
$ curl -o awscliv2.sig https://awscli.amazonaws.com/awscli-exe-linux-aarch64-2.0.30.zip.sig
```

有关版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub

5. 验证签名，并将下载的 `.sig` 和 `.zip` 文件名作为参数传递给 `gpg` 命令。

```
$ gpg --verify awscliv2.sig awscliv2.zip
```

该输出值应该类似于以下内容。

```
gpg: Signature made Mon Nov  4 19:00:01 2019 PST
gpg:                using RSA key FB5D B77F D5C1 18B8 0511 ADA8 A631 0ACC 4672 475C
gpg: Good signature from "AWS CLI Team <aws-cli@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:                There is no indication that the signature belongs to the owner.
Primary key fingerprint: FB5D B77F D5C1 18B8 0511 ADA8 A631 0ACC 4672 475C
```

**⚠ Important**

输出中的警告是预料中的，并不表示出现了问题。之所以发生这种情况，是因为您的个人 PGP 密钥（如果有的话）和 PGP 密钥之间没有信任链。AWS CLI 有关更多信息，请参阅[信任 Web](#)。

## macOS

### 安装要求

- 您知道要安装第 2 AWS CLI 版的哪个版本。有关版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub
- 在苹果支持 AWS CLI 的 64 位 macOS 版本上，我们支持第 2 版。
- 由于 AWS 不维护第三方存储库，因此我们无法保证它们包含最新版本的 AWS CLI。

### 安装说明

您可以通过以下方式在 macOS 上安装 AWS CLI 版本 2。

#### GUI installer

以下步骤说明如何使用标准 macOS 用户界面和浏览器安装或更新至最新 AWS CLI 版本的 2。如果要更新到最新版本，请使用与当前版本所用的相同安装方法。

1. 在浏览器中，通过在文件名后附加一个连字符和版本号来下载特定版本的 AWS CLI。

```
https://awscli.amazonaws.com/AWSCLIV2-version.number.pkg
```

在本示例中，版本 **2.0.30** 的文件名为 `AWSCLIV2-2.0.30.pkg`，这会生成以下链接：<https://awscli.amazonaws.com/AWSCLIV2-2.0.30.pkg>。

2. 运行下载的文件，然后按照屏幕上的说明进行操作。您可以选择通过以下方式安装 AWS CLI 版本 2：
  - 针对电脑上的所有用户（需要 **sudo**）
  - 您可以安装到任何文件夹，或选择推荐的默认文件夹 `/usr/local/aws-cli`。

- 安装程序会自动在 `/usr/local/bin/aws` 中创建一个符号链接，该链接指向您选择的安装文件夹中的主程序。
- 仅针对当前用户（不需要 **sudo**）
  - 您可以安装到具有写入权限的任何文件夹。
  - 由于具有标准用户权限，在安装程序完成后，您必须在命令提示符下使用以下命令在 `$PATH` 中手动创建指向 `aws` 和 `aws_completer` 程序的符号链接文件。如果 `$PATH` 包含您可以写入的文件夹，并且您将该文件夹指定为目标路径，则可以在不使用 `sudo` 的情况下运行以下命令。如果 `$PATH` 中没有可写入的文件夹，您必须在命令中使用 `sudo`，以获取写入指定目标文件夹的权限。符号链接的默认位置为 `/usr/local/bin/`。

```
$ sudo ln -s /folder/installed/aws-cli/aws /usr/local/bin/aws
$ sudo ln -s /folder/installed/aws-cli/aws_completer /usr/local/bin/
aws_completer
```

#### Note

您可以在安装过程中的任意时刻按 `Cmd+L` 来查看安装的调试日志。这会打开一个日志窗格，让您能够筛选和保存日志。日志文件还会自动保存到 `/var/log/install.log`。

3. 要验证 Shell 是否可以在 `$PATH` 中找到并运行 `aws` 命令，请使用以下命令。

```
$ which aws
/usr/local/bin/aws
$ aws --version
aws-cli/2.15.30 Python/3.11.6 Darwin/23.3.0 botocore/2.4.5
```

如果无法找到 `aws` 命令，您可能需要重新启动终端或按照[排查错误](#)中的说明排除故障。

## Command line installer - All users

如果您拥有 `sudo` 权限，则可以针对电脑上的所有用户安装 AWS CLI 版本 2。我们在一个易于复制和粘贴的组中提供步骤。请参阅以下步骤中每一行的描述。

对于的特定版本 AWS CLI，请在文件名后面附加连字符和版本号。在本示例中，版本 `2.0.30` 的文件名为 `AWSCLIv2-2.0.30.pkg`，这会生成以下命令：

```
$ curl "https://awscli.amazonaws.com/AWSCLIV2-2.0.30.pkg" -o "AWSCLIV2.pkg"
$ sudo installer -pkg AWSCLIV2.pkg -target /
```

1. 使用 `curl` 命令下载文件。`-o` 选项指定下载的程序包应写入的文件名。在此示例中，文件写入当前文件夹中的 `AWSCLIV2.pkg`。

对于的特定版本 AWS CLI，请在文件名后面附加连字符和版本号。在本示例中，版本 `2.0.30` 的文件名为 `AWSCLIV2-2.0.30.pkg`，这会生成以下命令：

```
$ curl "https://awscli.amazonaws.com/AWSCLIV2-2.0.30.pkg" -o "AWSCLIV2.pkg"
```

有关版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub

2. 运行标准 macOS `installer` 程序，同时将下载的 `.pkg` 文件指定为源。使用 `-pkg` 参数指定要安装的程序包的名称以及要将程序包安装到的驱动器的 `-target /` 参数。系统会将文件安装到 `/usr/local/aws-cli`，并自动在 `/usr/local/bin` 中创建符号链接。您必须在命令中使用 `sudo` 来授予对这些文件夹的写入权限。

```
$ sudo installer -pkg ./AWSCLIV2.pkg -target /
```

安装完成后，调试日志将写入到 `/var/log/install.log`。

3. 要验证 Shell 是否可以在 `$PATH` 中找到并运行 `aws` 命令，请使用以下命令。

```
$ which aws
/usr/local/bin/aws
$ aws --version
aws-cli/2.15.30 Python/3.11.6 Darwin/23.3.0 botocore/2.4.5
```

如果无法找到 `aws` 命令，您可能需要重新启动终端或按照[排查错误](#)中的说明排除故障。

## Command line - Current user

1. 要指定安装到哪个文件夹，必须创建一个 XML 文件。AWS CLI 此文件是 XML 格式的文件，类似于以下示例。保留所有值如图所示，但必须将第 9 行中的 `/users/myUsername` 路径替换为要安装 AWS CLI 版本 2 的文件夹的路径。必须存在该文件夹，否则命令会失败。此 XML 示例指定安装程序在文件夹 `AWS CLI` 中安装 `/Users/myusername`，并在其中创建一个名为 `aws-cli` 的文件夹。



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <array>
    <dict>
      <key>choiceAttribute</key>
      <string>customLocation</string>
      <key>attributeSetting</key>
      <string>/Users/myusername</string>
      <key>choiceIdentifier</key>
      <string>default</string>
    </dict>
  </array>
</plist>
```

2. 使用 `curl` 命令下载 `pkg` 安装程序。-o 选项指定下载的程序包应写入的文件名。在此示例中，文件写入当前文件夹中的 `AWSCLIV2.pkg`。

对于的特定版本 AWS CLI，请在文件名后面附加连字符和版本号。在本示例中，版本 `2.0.30` 的文件名为 `AWSCLIV2-2.0.30.pkg`，这会生成以下命令：

```
$ curl "https://awscli.amazonaws.com/AWSCLIV2-2.0.30.pkg" -o "AWSCLIV2.pkg"
```

有关版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub

3. 使用以下选项运行标准 macOS installer 程序：
  - 使用 `-pkg` 参数指定要安装的程序包的名称。
  - 通过将 `-target` 参数设置为 `CurrentUserHomeDirectory`，指定仅安装到当前用户。
  - 在 `-applyChoiceChangesXML` 参数中指定您创建的 XML 文件的路径（相对于当前文件夹）和名称。

以下示例将安装 AWS CLI 在文件夹中 `/Users/myusername/aws-cli`。

```
$ installer -pkg AWSCLIV2.pkg \
            -target CurrentUserHomeDirectory \
            -applyChoiceChangesXML choices.xml
```

4. 由于标准用户权限通常不允许写入 \$PATH 中的文件夹，因此在此模式下，安装程序不会尝试将符号链接添加到 aws 和 aws\_completer 程序中。AWS CLI 要使正常运行，必须在安装程序完成后手动创建符号链接。如果 \$PATH 包含您可以写入的文件夹，并且您将该文件夹指定为目标的路径，则可以在不使用 sudo 的情况下运行以下命令。如果 \$PATH 中没有可写入的文件夹，您必须使用 sudo，以获取写入指定目标文件夹的权限。符号链接的默认位置为 /usr/local/bin/。

```
$ sudo ln -s /folder/installed/aws-cli/aws /usr/local/bin/aws
$ sudo ln -s /folder/installed/aws-cli/aws_completer /usr/local/bin/
aws_completer
```

安装完成后，调试日志将写入到 /var/log/install.log。

5. 要验证 Shell 是否可以在 \$PATH 中找到并运行 aws 命令，请使用以下命令。

```
$ which aws
/usr/local/bin/aws
$ aws --version
aws-cli/2.15.30 Python/3.11.6 Darwin/23.3.0 botocore/2.4.5
```

如果无法找到 aws 命令，您可能需要重新启动终端或按照[排查错误](#)中的说明排除故障。

## Windows

### 安装要求

- 您知道要安装第 2 AWS CLI 版的哪个版本。有关版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub
- 我们支持微软支持 AWS CLI 的 64 位 Windows 版本。
- 安装软件的管理员权限

### 安装说明

要在 Windows 上更新当前安装的 AWS CLI 版本 2，请在每次更新时下载一个新的安装程序以覆盖以前的版本。AWS CLI 会定期更新。要查看最新版本的发布时间，请查看[AWS CLI 版本 2 的变更日志](#)。GitHub

1. 以下列方式之一下载并运行适用于 Windows (64 位) 的 AWS CLI MSI 安装程序：

- 正在@@ 下载并运行 MSI 安装程序：要为的特定版本创建下载链接 AWS CLI，请在文件名后面附加连字符和版本号。

```
https://awscli.amazonaws.com/AWSCLIV2-version.number.msi
```

在本示例中，版本 **2.0.30** 的文件名为 `AWSCLIV2-2.0.30.msi`，这会生成以下链接：<https://awscli.amazonaws.com/AWSCLIV2-2.0.30.msi>。

- 使用 `msiexec` 命令：您也可以通过将链接添加到 `msiexec` 命令来使用 MSI 安装程序。对于的特定版本 AWS CLI，请在文件名后面附加连字符和版本号。

```
C:\> msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2-version.number.msi
```

在本示例中，版本 **2.0.30** 的文件名为 `AWSCLIV2-2.0.30.msi`，这会生成以下链接：<https://awscli.amazonaws.com/AWSCLIV2-2.0.30.msi>。

```
C:\> msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2-2.0.30.msi
```

有关可与 `msiexec` 一起使用的各种参数，请参阅 Microsoft Docs 网站上的 [msiexec](#)。

有关版本列表，请参阅[AWS CLI 版本 2 变更日志](#)。GitHub

- 要确认安装，请打开 Start (开始) 菜单，搜索 `cmd` 以打开命令提示符窗口，然后在命令提示符处使用 `aws --version` 命令。

```
C:\> aws --version
aws-cli/2.15.30 Python/3.11.6 Windows/10 exe/AMD64 prompt/off
```

如果 Windows 无法找到该程序，您可能需要关闭并重新打开命令提示符窗口以刷新该路径，按照[排查错误](#)中的说明排除故障。

## AWS CLI 安装和卸载错误疑难解答

如果您在安装或卸载后遇到问题 AWS CLI，[排查错误](#)请参阅，了解故障排除步骤。有关相关性最高的故障排除步骤，请参阅[the section called “找不到命令错误”](#)、[the section called ““aws --version”命令返回的版本与您安装的版本不同”](#)和[the section called “卸载后，aws --version”命令会返回一个版本 AWS CLI”](#)。

## 后续步骤

完成中的步骤[the section called “先决条件”](#)并安装后 AWS CLI，应执行[the section called “设置”](#)。

## AWS CLI 从源代码构建并安装

本主题介绍如何在受支持的操作系统上从源代码安装或更新到最新版本的 AWS Command Line Interface (AWS CLI)。

有关最新版本的信息 AWS CLI，请参阅[AWS CLI 版本 2 变更日 GitHub 志](#)。

### Important

AWS CLI 版本 1 和 2 使用相同的aws命令名称。如果您之前安装了 AWS CLI 版本 1，请参阅[从 AWS CLI 版本 1 迁移至版本 2](#)。

### 主题

- [为什么要从源代码构建？](#)
- [快速步骤](#)
- [步骤 1：设置所有要求](#)
- [步骤 2：配置 AWS CLI 源代码安装](#)
- [步骤 3：构建 AWS CLI](#)
- [步骤 4：安装 AWS CLI](#)
- [步骤 5：验证 AWS CLI 安装](#)
- [工作流程示例](#)
- [AWS CLI 安装和卸载错误疑难解答](#)
- [后续步骤](#)

## 为什么要从源代码构建？

AWS CLI [可作为大多数平台和环境的预先构建的安装程序](#)以及 Docker 镜像提供。

通常，这些安装程序涵盖了大多数使用案例。从源代码安装的说明旨在帮助解决我们的安装程序未涵盖的使用案例。部分使用案例包括：

- 预构建的安装程序不支持您的环境。例如，预构建的安装程序不支持 ARM 32 位。
- 预构建的安装程序具有您的环境所缺少的依赖项。例如，Alpine Linux 使用 [musl](#)，但当前安装程序需要 `glibc`，而导致预构建的安装程序无法立即运行。
- 预构建的安装程序需要您的环境限制访问的资源。例如，安全强化型系统可能不会授予对共享内存的权限。这是冻结的 `aws` 安装程序所需要的。
- 因为最好能够完全控制代码和软件包的构建过程，而预构建的安装程序通常会成为软件包管理器中维护者的阻碍。从源代码构建使发行版维护者能够更简化流程来保持 AWS CLI 更新。通过第三方软件包管理器（例如 `brew`、`和`）进行安装 AWS CLI 时，启用维护者可以为客户提供更多 up-to-date 版本 `yum` 的 `apt`。
- 修补 AWS CLI 功能的客户需要 AWS CLI 从源代码构建和安装。对于想要在将更改贡献到 AWS CLI GitHub 存储库之前测试自己对源代码所做的更改的社区成员来说，这一点尤其重要。

## 快速步骤

### Note

假设所有代码示例都从源代码目录的根目录运行。

要 AWS CLI 从源代码构建和安装，请按照本节中的步骤操作。AWS CLI 利用 [GNU Autotools](#) 从源代码进行安装。在最简单的情况下，AWS CLI 可以通过从 AWS CLI GitHub 存储库的根目录运行默认示例命令从源代码安装。

1. [设置您的环境的所有要求](#)。这包括能够运行 [GNU Autotools](#) 生成的文件以及安装 Python 3.8 或更高版本。
2. 在终端中，导航到 AWS CLI 源文件夹的顶层并运行该 `./configure` 命令。此命令检查系统是否存在所有必需的依赖关系，并 AWS CLI 根据检测到的和指定的配置生成一个 `Makefile` 用于构建和安装的依赖关系。

### Linux and macOS

以下 `./configure` 命令示例 AWS CLI 使用默认设置来设置编译配置。

```
$ ./configure
```

## Windows PowerShell

在运行任何调用 MSYS2 的命令之前，必须保留当前工作目录：

```
PS C:\> $env:CHERE_INVOKING = 'yes'
```

然后使用以下 `./configure` 命令示例设置编译配置，以便 AWS CLI 使用 Python 可执行文件的本地路径、安装到 `C:\Program Files\AWSCLI` 以及下载所有依赖项。

```
PS C:\> C:\msys64\usr\bin\bash -lc " PYTHON='C:\path\to\python.exe' ./configure --prefix='C:\Program Files\AWSCLI' --with-download-deps "
```

有关详细信息、可用配置选项和默认设置信息，请参阅 [the section called “步骤 2：配置 AWS CLI 源代码安装”](#) 部分。

3. 运行 `make` 命令。此命令 AWS CLI 根据您的配置设置构建。

以下 `make` 命令示例使用您的现有 `./configure` 设置通过默认选项进行构建。

## Linux and macOS

```
$ make
```

## Windows PowerShell

```
PS C:\> C:\msys64\usr\bin\bash -lc "make"
```

有关详细信息和可用的构建选项，请参阅 [the section called “步骤 3：构建 AWS CLI”](#) 部分。

4. 运行 `make install` 命令。此命令将您构建的 AWS CLI 安装到系统上所配置的位置。

以下 `make install` 命令示例使用默认命令设置，在所配置的位置安装您构建的 AWS CLI 并创建符号链接。

## Linux and macOS

```
$ make install
```

## Windows PowerShell

```
PS C:\> C:\msys64\usr\bin\bash -lc "make install"
```

安装后，AWS CLI 使用以下命令将路径添加到中：

```
PS C:\> $Env: PATH +=";C:\Program Files\AWSCLI\bin\"
```

有关详细信息和可用的安装选项，请参阅 [the section called “步骤 4：安装 AWS CLI”](#) 部分。

### 5. 使用以下命令确认已 AWS CLI 成功安装：

```
$ aws --version  
aws-cli/2.15.30 Python/3.11.6 Windows/10 exe/AMD64 prompt/off
```

有关安装错误的故障排除步骤，请参阅 [the section called “AWS CLI 安装和卸载错误疑难解答”](#) 部分。

## 步骤 1：设置所有要求

要 AWS CLI 从源代码构建，你需要事先完成以下内容：

### Note

假设所有代码示例都从源代码目录的根目录运行。

### 1. 通过分叉 AWS CLI GitHub 存储库或下载 AWS CLI 源代码压缩包来下载源代码。有关说明，请参阅以下内容之一：

- 从中分叉并克隆 [AWS CLI 存储库 GitHub](#)。有关更多信息，请参阅 GitHub 文档中的 [Fork 存储库](#)。
- 通过 <https://awscli.amazonaws.com/awscli.tar.gz> 下载最新的源代码压缩包，使用以下命令解压缩内容：

```
$ curl -o awscli.tar.gz https://awscli.amazonaws.com/awscli.tar.gz  
$ tar -xzf awscli.tar.gz
```

**Note**

要下载特定版本，请使用以下链接格式：[https://awscli.amazonaws.com/  
awscli-versionnumber.tar.gz](https://awscli.amazonaws.com/awscli-versionnumber.tar.gz)

例如，对于版本 2.10.0，链接如下：[https://awscli.amazonaws.com/  
awscli-2.10.0.tar.gz](https://awscli.amazonaws.com/awscli-2.10.0.tar.gz)

源版本从 AWS CLI 的 2.10.0 版本开始可用。

( 可选 ) 通过完成以下步骤，验证下载的 zip 文件的完整性：

1. 您可以通过以下步骤使用 GnuPG 工具验证签名。

AWS CLI 安装程序包 .zip 文件使用 PGP 签名进行加密签名。如果文件有任何损坏或更改，则此验证失败，您不应继续安装。

2. 使用您的程序包管理器下载并安装 gpg 命令。有关 GnuPG 的更多信息，请参阅 [GnuPG 网站](#)。
3. 要创建公有密钥文件，请创建一个文本文件并粘贴到以下文本中。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
mQINBF2Cr7UBEADJZHcgusOJl7ENSyumXh85z0TRV0xJorM2B/JL0kH0yigQluUG
ZMLhENaG0bYatdrKP+3H911vK050pXwn0/R7fB/FSTouki4ciIx50uLlnJZIxSzx
PqG10mkxImLNBGwoi6Lto0LYxqHN2iQtzlwTVmq9733zd3XfcXrZ3+Lb1HAgEt5G
TfNxEKJ8soPLYWmwDH6HWCnjZ/aIQRBTIQ05uVeEoYxSh6w0ai7ss/KveoSNBbYz
gbdzoqI2Y8cgH2nbfpg3DSasaLZEdCSsIsK1u05CinE7k2qZ7KgKAUIcT/cR/grk
C6VwsnDU00UCideXcQ8WeHutqvgZH1JgKDbznoIzeQHJD238GEu+eKhRHcz8/jeG
94zkcqJ0z3KbZGYMiTh277Fvj9zzvZsbMBCedV1BTg3Tqgvdx4bdkhf5cH+7NtW0
lrFj6UwAsGukBTA0xC01/dnSmZhJ7Z1KmEWi1ro/g0rjt0xqRQut1IqG22TaqoPG
fYVN+en3Zwbt97kcgZDwqbuyknt64oZwc4XKCa3mprEGC3IbJTBfqq1XmZ719yWg
EEUJY01b2XrSuPwM139beWdKM8kzr10jn10m6+1pTRCBfo0wa9F8YZRhHPAkwKkX
XDe0GpWRj4oh0x0d2GWkyV5xyN14p2tQ0Cd00Dmz80yUTgRpPVQUt0EhXQARAQAB
tCFBV1MgQ0xJIFRlYW0gPGF3cy1jbG1AYW1hem9uLmNvbT6JAlQEEwEiAD4WIQT7
Xbd/1cEYUauraimMQrMRnJHXAUCXYkvTQIbAwUJB4TOAAULCQgHAgYVVCgkICwIE
FgIDAQIEAQIXgAAKRCrCmMQrMRnJHXJIXEACHLUIkg80uPUkGjE3jejvQSA1aWuAM
zy6fdpd1RUz6M6nmsUh0ExjVIvibEJpzk5mhuSZ41b0vJ2ZUPgCv4zs2nBd7BGJ
MxKiWgBREgVtdqZ0SzyYH4PYCJSE732x/Fw9hfnh1dMTXNcrQXzw0mmFNNegG00x
au+VnpcR5Kz3smiTrIwZbRudo1ijhCYPQ7t5Cmp9kjC6b0bvy1hSIg2xNbMAN/Do
ikebA136uA6Y/Uczjj3GxZW4ZWeFirMidKbtqvUz2y0UFszobjiBSqZZHCreC34B
hw9bFNpuWC/0SrXgohdsc6vK50pDGDv5kM2qo9tMQ/izsAwTh/d/GzZv8H41V9e0
```



```
tEis+EpR497PaxKKh9tJf0N6Q1YLRHof5xePZt0I1S3gfvsh5hXA3HJ9yIxb8T0H
QYmVr3aIUes20i6meI3fuV36VFupwfrTKaL7VXnsrK2fq5cRvyJLNzXucg0WAjPF
RrAGLzY7nP1xeg1a0aeP+pdsqjq1PJom80Cwc1+6DWbg0jsC74WoesAqgBIt0DMB
rsal1y/q+bPzpsnWjzHV8+1/EtZmSc8ZUGSJ0PkfC7h0bnfk118h+1QtKTjZme4d
H17gsBJr+opwJw/Zio2LMjQB0qlm3K1A4zFTh7wBC7He6KPQea1p2XAMgtvATtNe
YLZATHZKTJyiqA==
=vY0k
-----END PGP PUBLIC KEY BLOCK-----
```

以下是公有密钥的详细信息以供参考。

```
Key ID:          A6310ACC4672
Type:           RSA
Size:          4096/4096
Created:       2019-09-18
Expires:      2023-09-17
User ID:      AWS CLI Team <aws-cli@amazon.com>
Key fingerprint:  FB5D B77F D5C1 18B8 0511 ADA8 A631 0ACC 4672 475C
```

- 使用以下命令导入 AWS CLI 公钥，*public-key-file-name* 替换为您创建的公钥的文件名。

```
$ gpg --import public-key-file-name
gpg: /home/username/.gnupg/trustdb.gpg: trustdb created
gpg: key A6310ACC4672475C: public key "AWS CLI Team <aws-cli@amazon.com>"
imported
gpg: Total number processed: 1
gpg:          imported: 1
```

- 下载你下载的软件包的 AWS CLI 签名文件，[网址为 https://awscli.amazonaws.com/awscli.tar.gz.sig](https://awscli.amazonaws.com/awscli.tar.gz.sig)。它具有与其对应的压缩包文件相同的路径和名称，但扩展名为 `.sig`。将其保存在与压缩包文件相同的路径中。或者使用以下命令块：

```
$ curl -o awscliv2.sig https://awscli.amazonaws.com/awscli.tar.gz.sig
```

- 验证签名，并将下载的 `.sig` 和 `.zip` 文件名作为参数传递给 `gpg` 命令。

```
$ gpg --verify awscliv2.sig awscli.tar.gz
```

该输出值应该类似于以下内容。

```
gpg: Signature made Mon Nov  4 19:00:01 2019 PST
gpg:                using RSA key FB5D B77F D5C1 18B8 0511 ADA8 A631 0ACC 4672
 475C
gpg: Good signature from "AWS CLI Team <aws-cli@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:                There is no indication that the signature belongs to the owner.
Primary key fingerprint: FB5D B77F D5C1 18B8 0511  ADA8 A631 0ACC 4672 475C
```

### Important

输出中的警告是预料中的，并不表示出现了问题。之所以出现此警告，是因为您的个人 PGP 密钥（如果您有）和 AWS CLI PGP 密钥之间没有信任链。有关更多信息，请参阅[信任 Web](#)。

2. 您的环境可以运行 [GNU Autotools](#) 生成的文件，如 `configure` 和 `Makefile`。这些文件可跨 POSIX 平台广泛移植。

### Linux and macOS

如果您的环境中尚未安装 Autotools 或者您需要对其进行更新，请按照《GNU 文档》中的[如何安装 Autotools \(以用户身份\) ?](#) 或[基本安装](#)中的安装说明进行操作。

### Windows PowerShell

### Warning

我们建议，如果您在 Windows 环境中，请使用预构建的安装程序。有关预构建的安装程序的安装说明，请参阅 [the section called “安装/更新”](#)

由于 Windows 不附带兼容 POSIX 的外壳，因此您需要安装其他软件才能 AWS CLI 从源代码安装。[MSYS2](#) 提供了一系列工具和库来帮助构建和安装 Windows 软件，特别是对于 Autotools 使用的基于 POSIX 的脚本。

1. 安装 MSYS2。有关安装和使用 MSYS2 的信息，请参阅《MSYS2 文档》中的[安装和使用说明](#)。
2. 打开 MSYS2 终端并使用以下命令安装自动工具。

```
$ pacman -S autotools
```

### Note

使用本指南中适用于 Windows 的配置、构建和安装代码示例时，假定默认的 MSYS2 安装路径为 C:\msys64\usr\bin\bash。在内部调用 MSYS2 时，PowerShell 您将使用以下格式，bash 命令用引号表示：

```
PS C:\> C:\msys64\usr\bin\bash -lc "command example"
```

以下命令示例调用 ./configure 命令。

```
PS C:\> C:\msys64\usr\bin\bash -lc "./configure"
```

3. 安装了 Python 3.8 或更高版本的解释器。所需的最低 Python 版本遵循与 [AWS SDK 和工具的官方 Python 支持政策](#) 相同的时间表。口译员只能在口译结束 end-of-support 后的 6 个月内获得支持。
4. (可选) 安装 AWS CLI 的所有构建和运行时 Python 库依赖项。./configure 命令会通知您是否缺少任何依赖项以及如何安装它们。

您可以通过配置自动安装和使用这些依赖项。有关更多信息，请参阅 [the section called “下载依赖项”](#)。

## 步骤 2：配置 AWS CLI 源代码安装

构建和安装的配置 AWS CLI 是使用 configure 脚本指定的。要获取所有配置选项的文档，请使用 --help 选项运行 configure 脚本：

Linux and macOS

```
$ ./configure --help
```

Windows PowerShell

```
PS C:\> C:\msys64\usr\bin\bash -lc "./configure --help"
```

最重要的选项如下所示：

- [安装位置](#)
- [Python 解释器](#)
- [下载依赖项](#)
- [安装类型](#)

## 安装位置

的源代码安装 AWS CLI 使用两个可配置的目录来安装 AWS CLI：

- `libdir`-要安装 AWS CLI 的父目录。AWS CLI 安装路径是 `<libdir-value>/aws-cli`。Linux 和 macOS 的默认 `libdir` 值是 `/usr/local/lib`，它创建默认安装目录 `/usr/local/lib/aws-cli`
- `bindir`-安装 AWS CLI 可执行文件的目录。默认位置是 `/usr/local/bin`。

以下 `configure` 选项控制所使用的目录：

- `--prefix` - 设置要用于安装的目录前缀。Linux 和 macOS 的默认值为 `/usr/local`。
- `--libdir` - 设置要用于安装 AWS CLI 的 `libdir`。默认值为 `<prefix-value>/lib`。如果未指定 `--libdir` 和 `--prefix`，则 Linux 和 macOS 的默认设置为 `/usr/local/lib/`。
- `--bindir`-设置 `bindir` 用于安装 AWS CLI `aws` 和 `aws_completer` 可执行文件。默认值为 `<prefix-value>/bin`。如果未指定 `bindir` 和 `--prefix`，则 Linux 和 macOS 的默认设置为 `/usr/local/bin/`。

## Linux and macOS

以下命令示例使用 `--prefix` 选项执行 AWS CLI 的本地用户安装。此命令将 AWS CLI 在 `$HOME/.local/lib/aws-cli` 和可执行文件安装在 `$HOME/.local/bin` 以下位置：

```
$ ./configure --prefix=$HOME/.local
```

以下命令示例使用 `--libdir` 选项将 AWS CLI 作为附加应用程序安装到 `/opt` 目录中。此命令将 AWS CLI 在 `/opt/aws-cli` 和可执行文件安装到其默认位置 `/usr/local/bin`

```
$ ./configure --libdir=/opt
```

## Windows PowerShell

以下命令示例使用 `--prefix` 选项执行 AWS CLI 的本地用户安装。此命令将 AWS CLI 在 `$HOME/.local/lib/aws-cli` 和可执行文件安装在 `$HOME/.local/bin` 以下位置：

```
$ C:\msys64\usr\bin\bash -lc "./configure --prefix='C:\Program Files\AWSCLI'"
```

以下命令示例使用 `--libdir` 选项将 AWS CLI 作为附加应用程序安装到 `/opt` 目录中。此命令会安装 a AWS CLI 到 `C:\Program Files\AWSCLI\opt\aws-cli`。

## Python 解释器

### Note

强烈建议在针对 Windows 进行安装时指定 Python 解释器。

该 `./configure` 脚本会自动选择已安装的 Python 3.8 或更高版本的解释器，用于使用 [AM\\_PATH\\_PYTHON](#) Autoconf AWS CLI 宏构建和运行该解释器。

运行 `configure` 脚本时，可以使用 `PYTHON` 环境变量显式设置要使用的 Python 解释器：

### Linux and macOS

```
$ PYTHON=/path/to/python ./configure
```

### Windows PowerShell

```
PS C:\> C:\msys64\usr\bin\bash -lc "PYTHON='C:\path\to\python' ./configure"
```

## 下载依赖项

默认情况下，要求 AWS CLI 的所有构建和运行时依赖项都已安装在系统上。这包括任何 Python 库依赖项。运行 `configure` 脚本时会检查所有依赖项，如果系统缺少任何 Python 依赖项，则 `configure` 脚本会出错。

当您的系统缺少依赖项时，以下代码示例会出错：

## Linux and macOS

```
$ ./configure
checking for a Python interpreter with version >= 3.8... python
checking for python... /Users/username/.envs/env3.11/bin/python
checking for python version... 3.11
checking for python platform... darwin
checking for GNU default python prefix... ${prefix}
checking for GNU default python exec_prefix... ${exec_prefix}
checking for python script directory (pythondir)... ${PYTHON_PREFIX}/lib/python3.11/
site-packages
checking for python extension module directory (pyexecdir)... ${PYTHON_EXEC_PREFIX}/
lib/python3.11/site-packages
checking for --with-install-type... system-sandbox
checking for --with-download-deps... Traceback (most recent call last):
  File "<frozen runpy>", line 198, in _run_module_as_main
  File "<frozen runpy>", line 88, in _run_code
  File "/Users/username/aws-code/aws-cli/./backends/build_system/__main__.py", line
125, in <module>
    main()
  File "/Users/username/aws-code/aws-cli/./backends/build_system/__main__.py", line
121, in main
    parsed_args.func(parsed_args)
  File "/Users/username/aws-code/aws-cli/./backends/build_system/__main__.py", line
49, in validate
    validate_env(parsed_args.artifact)
  File "/Users/username/aws-code/aws-cli/./backends/build_system/validate_env.py",
line 68, in validate_env
    raise UnmetDependenciesException(unmet_deps, in_venv)
validate_env.UnmetDependenciesException: Environment requires following Python
dependencies:

awsCRT (required: ('>=0.12.4', '<0.17.0')) (version installed: None)

We recommend using --with-download-deps flag to automatically create a virtualenv
and download the dependencies.

If you want to manage the dependencies yourself instead, run the following pip
command:
/Users/username/.envs/env3.11/bin/python -m pip install --prefer-binary
'awsCRT>=0.12.4,<0.17.0'

configure: error: "Python dependencies not met."
```

## Windows PowerShell

```
PS C:\> C:\msys64\usr\bin\bash -lc "./configure"
checking for a Python interpreter with version >= 3.8... python
checking for python... /Users/username/.envs/env3.11/bin/python
checking for python version... 3.11
checking for python platform... darwin
checking for GNU default python prefix... ${prefix}
checking for GNU default python exec_prefix... ${exec_prefix}
checking for python script directory (pythondir)... ${PYTHON_PREFIX}/lib/python3.11/
site-packages
checking for python extension module directory (pyexecdir)... ${PYTHON_EXEC_PREFIX}/
lib/python3.11/site-packages
checking for --with-install-type... system-sandbox
checking for --with-download-deps... Traceback (most recent call last):
  File "<frozen runpy>", line 198, in _run_module_as_main
  File "<frozen runpy>", line 88, in _run_code
  File "/Users/username/aws-code/aws-cli/./backends/build_system/__main__.py", line
125, in <module>
    main()
  File "/Users/username/aws-code/aws-cli/./backends/build_system/__main__.py", line
121, in main
    parsed_args.func(parsed_args)
  File "/Users/username/aws-code/aws-cli/./backends/build_system/__main__.py", line
49, in validate
    validate_env(parsed_args.artifact)
  File "/Users/username/aws-code/aws-cli/./backends/build_system/validate_env.py",
line 68, in validate_env
    raise UnmetDependenciesException(unmet_deps, in_venv)
validate_env.UnmetDependenciesException: Environment requires following Python
dependencies:

awsCRT (required: ('>=0.12.4', '<0.17.0')) (version installed: None)

We recommend using --with-download-deps flag to automatically create a virtualenv
and download the dependencies.

If you want to manage the dependencies yourself instead, run the following pip
command:
/Users/username/.envs/env3.11/bin/python -m pip install --prefer-binary
'awsCRT>=0.12.4,<0.17.0'

configure: error: "Python dependencies not met."
```

要自动安装所需的 Python 依赖项，请使用 `--with-download-deps` 选项。使用此标志时，构建过程会执行以下操作：

- 跳过 Python 库依赖项检查。
- 将设置配置为下载所有必需的 Python 依赖项，并在构建 AWS CLI 期间仅使用下载的依赖项来 `make` 构建。

以下配置命令示例使用 `--with-download-deps` 选项下载和使用 Python 依赖项：

Linux and macOS

```
$ ./configure --with-download-deps
```

Windows PowerShell

```
PS C:\> C:\msys64\usr\bin\bash -lc " ./configure --with-download-deps"
```

## 安装类型

源代码安装过程支持以下安装类型：

- `system-sandbox-`（默认）创建隔离的 Python 虚拟环境，将其安装 AWS CLI 到虚拟环境中，并在虚拟环境中使用符号链接到 `aws` 和 `aws_completer` 的可执行文件。此安装的运行时直接 AWS CLI 取决于所选的 Python 解释器。

这是一种用于在系统上安装的轻量级 AWS CLI 安装机制，它遵循了 Python 的最佳实践，在虚拟环境中对安装进行沙箱。此安装适用于希望以尽可能顺畅的方式 AWS CLI 从源代码安装并将安装与 Python 安装相结合的客户。

- `portable-exe-` 将冻结 AWS CLI 为独立的可执行文件，该可执行文件可以分发到架构相似的环境。这与生成 AWS CLI 的官方预构建可执行文件的过程相同。`portable-exe` 冻结在 `configure` 步骤中选择的 Python 解释器的副本中，以用于 AWS CLI 的运行时。这允许将其移到其他可能没有 Python 解释器的计算机上。

这种类型的版本很有用，因为您可以确保您的 AWS CLI 安装不与环境中已安装的 Python 版本耦合，并且可以将版本分发到可能尚未安装 Python 的其他系统。这使您能够控制所使用的可 AWS CLI 执行文件的依赖关系和安全性。



要配置安装类型，请使用 `--with-install-type` 选项并指定 `portable-exe` 或 `system-sandbox` 的值。

以下 `./configure` 命令示例指定 `portable-exe` 的值：

Linux and macOS

```
$ ./configure --with-install-type=portable-exe
```

Windows PowerShell

```
PS C:\> C:\msys64\usr\bin\bash -lc "./configure --with-install-type=portable-exe"
```

### 步骤 3：构建 AWS CLI

使用以下 `make` 命令 AWS CLI 使用您的配置设置进行构建：

Linux and macOS

```
$ make
```

Windows PowerShell

```
PS C:\> C:\msys64\usr\bin\bash -lc "make"
```

#### Note

使用 `make` 命令时，将在后台完成以下步骤：

1. 使用 Python [venv](#) 模块在构建目录中创建虚拟环境。虚拟环境是使用 [Python 标准库中提供的 pip 版本](#) 引导的。
2. 复制 Python 库依赖项。根据是否在 `configure` 命令中指定了 `--with-download-deps` 标志，此步骤将执行以下操作之一：
  - 指定了 `--with-download-deps`。Python 依赖项是 `pip` 安装的。这包括 `wheel`、`setuptools` 和所有 AWS CLI 运行时依赖项。如果您正在构建 `portable-`

exe，则安装 `pyinstaller`。这些要求都是在通过 [pip-compile](#) 生成的锁定文件中指定的。

- 未指定 `--with-download-deps`。来自 Python 解释器的站点包中的 Python 库以及任何脚本（例如 `pyinstaller`）都被复制到用于构建的虚拟环境中。
3. `pip install` 直接在 AWS CLI 代码库上运行，进行离线、树内构建，然后将其安装到构建虚拟环境中。[此安装使用 pip 标志 `--no-build-isolation`、`--use-feature=in-tree-build`、`--` 和 `no-cache-dir --no-index`](#)
  4. （可选）如果在 `configure` 命令中将 `--install-type` 设置为 `portable-exe`，则使用 [pyinstaller](#) 生成独立的可执行文件。

## 步骤 4：安装 AWS CLI

该 `make install` 命令会将您构建的版本安装到系统上的配置位置。

### Linux and macOS

以下命令示例 AWS CLI 使用您的配置和编译设置进行安装：

```
$ make install
```

### Windows PowerShell

以下命令示例 AWS CLI 使用您的配置和编译设置安装，然后添加一个包含以下路径的环境变量 AWS CLI：

```
PS C:\> C:\msys64\usr\bin\bash -lc " make install "
```

```
PS C:\> $Env: PATH += ";C:\Program Files\AWSCLI\bin\"
```

`make install` 规则支持 [DESTDIR](#) 变量。指定后，此变量会在安装 AWS CLI 时将指定的路径作为已配置的安装路径的前缀。默认情况下，未为该变量设置任何值。

### Linux and macOS

以下代码示例使用 `--prefix=/usr/local` 标志来配置安装位置，然后对 `make install` 命令使用 `DESTDIR=/tmp/stage` 来更改该目的地。这些命令会导致 AWS CLI 安装在 `/tmp/stage/usr/local/lib/aws-cli` 中，而其可执行文件位于 `/tmp/stage/usr/local/bin` 中。

```
$ ./configure --prefix=/usr/local
$ make
$ make DESTDIR=/tmp/stage install
```

## Windows PowerShell

以下代码示例使用 `--prefix=\awscli` 标志来配置安装位置，然后对 `make install` 命令使用 `DESTDIR=C:\Program Files` 来更改该目的地。这些命令会导致 AWS CLI 安装在 `C:\Program Files\awscli` 中。

```
$ ./configure --prefix=\awscli
$ make
$ make DESTDIR='C:\Program Files' install
```

### Note

运行 `make install` 时，将在后台完成以下步骤

1. 将以下内容之一移至配置的安装目录：
  - 如果安装类型为 `system-sandbox`，则移动您构建的虚拟环境。
  - 如果安装类型为 `portable-exe`，则移动已构建的独立可执行文件。
2. 在配置的 `bin` 目录中为 `aws` 和 `aws_completer` 可执行文件创建符号链接。

## 步骤 5：验证 AWS CLI 安装

使用以下命令确认已 AWS CLI 成功安装：

```
$ aws --version
aws-cli/2.15.30 Python/3.11.6 Windows/10 exe/AMD64 prompt/off
```

如果无法识别 `aws` 命令，您可能需要重启终端以更新新的符号链接。如果您在安装或卸载后遇到其他问题 AWS CLI，请参阅，了解常见[排查错误](#)的疑难解答步骤

## 工作流程示例

本节提供了一些从源代码安装的基本工作流示例。

## 基本 Linux 和 macOS 安装

以下示例是一个基本的安装工作流程，AWS CLI 其中安装在的默认位置 `/usr/local/lib/aws-cli`。

```
$ cd path/to/cli/respository/
$ ./configure
$ make
$ make install
```

## 自动 Windows 安装

### Note

您必须以管理员 PowerShell 身份运行才能使用此工作流程。

MSYS2 可以在 CI 设置中自动使用，请参阅《MSYS2 文档》中的[在 CI 中使用 MSYS2](#)。

### Downloaded Tarball

下载 `awscli.tar.gz` 文件，解压缩，然后安装 AWS CLI。使用以下命令时，请替换以下路径：

- `C:\msys64\usr\bin\bash` ( 包括 MSYS2 路径的位置 )。
- `.\awscli-2.x.x\` ( 包括解压缩的 `awscli.tar.gz` 文件夹名称 )。
- `PYTHON='C:\path\to\python.exe'` ( 包括本地 Python 路径 )。

以下代码示例 PowerShell 使用 MSYS2 自动构建和安装，并指定要使用哪个 Python 的本地安装：  
AWS CLI

```
PS C:\> curl -o awscli.tar.gz https://awscli.amazonaws.com/awscli.tar.gz #
  Download the awscli.tar.gz file in the current working directory
PS C:\> tar -xvzf .\awscli.tar.gz # Extract awscli.tar.gz file
PS C:\> cd .\awscli-2.x.x\ # Navigate to the root of the extracted files
PS C:\> $env:CHERE_INVOKING = 'yes' # Preserve the current working directory
PS C:\> C:\msys64\usr\bin\bash -lc " PYTHON='C:\path\to\python.exe' ./configure --
prefix='C:\Program Files\AWSCLI' --with-download-deps "
PS C:\> C:\msys64\usr\bin\bash -lc "make"
PS C:\> C:\msys64\usr\bin\bash -lc "make install"
```

```
PS C:\> $Env:PATH +=";C:\Program Files\AWSCLI\bin\"
PS C:\> aws --version
aws-cli/2.15.30 Python/3.11.6 Windows/10 source-sandbox/AMD64 prompt/off
```

## GitHub Repository

下载 `awscli.tar.gz` 文件，解压缩，然后安装 AWS CLI。使用以下命令时，请替换以下路径：

- `C:\msys64\usr\bin\bash` ( 包括 MSYS2 路径的位置 )。
- `C:\path\to\cli\repository` \ 其中包含克隆 [AWS CLI 存储库](#) 的 GitHub 路径。如需了解更多信息，请参阅 [文档中的 Fork 存储库](#) GitHub
- `PYTHON='C:\path\to\python.exe'` ( 包括本地 Python 路径 )。

以下代码示例 PowerShell 使用 MSYS2 自动构建和安装，并指定要使用哪个 Python 的本地安装：  
AWS CLI

```
PS C:\> cd C:\path\to\cli\repository\
PS C:\> $env:CHERE_INVOKING = 'yes' # Preserve the current working directory
PS C:\> C:\msys64\usr\bin\bash -lc " PYTHON='C:\path\to\python.exe' ./configure --
prefix='C:\Program Files\AWSCLI' --with-download-deps "
PS C:\> C:\msys64\usr\bin\bash -lc "make"
PS C:\> C:\msys64\usr\bin\bash -lc "make install"
PS C:\> $Env:PATH +=";C:\Program Files\AWSCLI\bin\"
PS C:\> aws --version
```

## Alpine Linux 容器

以下是一个 Dockerfile 示例，可用于在 Alpine Linux 容器 AWS CLI 中进行有效安装，[以替代预先构建的 Alpine 二进制文件](#)。使用此示例时，请 `AWSCLI_VERSION` 用所需的 AWS CLI 版本号替换：

```
FROM python:3.8-alpine AS builder

ENV AWSCLI_VERSION=2.10.1

RUN apk add --no-cache \
    curl \
    make \
    cmake \
    gcc \
    g++ \
```

```
libc-dev \  
libffi-dev \  
openssl-dev \  
&& curl https://awscli.amazonaws.com/awscli-${AWSCLI_VERSION}.tar.gz | tar -xz \  
&& cd awscli-${AWSCLI_VERSION} \  
&& ./configure --prefix=/opt/aws-cli/ --with-download-deps \  
&& make \  
&& make install  
  
FROM python:3.8-alpine  
  
RUN apk --no-cache add groff  
  
COPY --from=builder /opt/aws-cli/ /opt/aws-cli/  
  
ENTRYPOINT ["/opt/aws-cli/bin/aws"]
```

此镜像是从类似于 Amazon Linux 2 上构建的容器中构建和 AWS CLI 调用的：

```
$ docker build --tag awscli-alpine .  
$ docker run --rm -it awscli-alpine --version  
aws-cli/2.2.1 Python/3.8.11 Linux/5.10.25-linuxkit source-sandbox/x86_64.alpine.3  
prompt/off
```

此镜像的最终大小小于 AWS CLI Docker 官方镜像的大小。有关官方 Docker 映像的信息，请参阅 [the section called “Amazon ECR Public/Docker”](#)。

## AWS CLI 安装和卸载错误疑难解答

有关安装错误的故障排除步骤，请参阅 [排查错误](#) 以了解常见故障排除步骤。有关相关性最高的故障排除步骤，请参阅 [the section called “找不到命令错误”](#)、[the section called ““aws --version”命令返回的版本与您安装的版本不同”](#) 和 [the section called “卸载后，aws --version”命令会返回一个版本 AWS CLI”](#)。

对于疑难解答指南中未涉及的任何问题，请使用 [AWS CLI 存储库](#) 中的 source-distribution 标签搜索问题 GitHub。如果没有现有问题可以掩盖您的错误，请 [创建一个新问题](#) 以获得 AWS CLI 维护者的帮助。

## 后续步骤

安装完成后 AWS CLI，您应该执行一个 [the section called “设置”](#)。

# 使用亚马逊 ECR 官方公共镜像或 Docker 镜像运行 AWS CLI

本主题介绍如何使用官方的亚马逊弹性容器注册表 ( Amazon ECR Public ) 或 Docker Hub 镜像在 Docker 上运行、AWS CLI 版本控制和配置版本 2。有关如何使用 Docker 的更多信息，请参阅 [Docker 的文档](#)。

官方镜像提供 AWS 直接支持和维护的隔离、便携性和安全性。这使您能够在基于容器的环境中使用 AWS CLI 版本 2，而不必自己管理安装。

## 主题

- [先决条件](#)
- [在 Amazon ECR Public 和 Docker Hub 之间作出选择](#)
- [运行官方 AWS CLI 版 2 张图片](#)
- [有关官方映像的接口和向后兼容性的说明](#)
- [使用特定版本和标签](#)
- [更新到最新的官方映像](#)
- [共享主机文件、凭据、环境变量和配置](#)
- [缩短 docker run 命令](#)

## 先决条件

您必须已经安装 Docker。有关安装说明，请参阅 [Docker 网站](#)。

要验证 Docker 的安装，请运行以下命令并确认是否有输出。

```
$ docker --version
Docker version 19.03.1
```

## 在 Amazon ECR Public 和 Docker Hub 之间作出选择

我们建议使用 Amazon ECR Public 而不是 Docker Hub 来处理映像 AWS CLI。Docker Hub 对公有使用者有更严格的速率限制，这会导致出现节流问题。此外，Amazon ECR Public 会在多个区域复制映像，以提供强大的可用性和处理区域中断问题。

有关 Docker Hub 速率限制的更多信息，请参阅 Docker 网站上的 [了解 Docker Hub 速率限制](#)。

## 运行官方 AWS CLI 版 2 张图片

首次使用 `docker run` 命令时，系统会将最新映像下载到计算机。`docker run` 命令的每次后续使用都会从本地副本运行。

要运行 AWS CLI 版本 2 的 Docker 镜像，请使用 `docker run` 命令。

### Amazon ECR Public

[正式 AWS CLI 版本 2 Amazon ECR 公共镜像托管在 Amazon ECR Public 的存储库中。aws-cli/aws-cli](#)

```
$ docker run --rm -it public.ecr.aws/aws-cli/aws-cli command
```

### Docker Hub

正式 AWS CLI 版本 2 的 Docker 镜像托管在 Docker Hub 的存储库中。amazon/aws-cli

```
$ docker run --rm -it amazon/aws-cli command
```

下面是此命令运行方式：

- `docker run --rm -it repository/name -` 等效于 `aws` 可执行文件。每次运行此命令时，Docker 都会启动已下载映像的容器，并执行您的 `aws` 命令。默认情况下，镜像使用最新版本的 AWS CLI 版本 2。

例如，要在 Docker 中调用 `aws --version` 命令，请运行以下命令。

### Amazon ECR Public

```
$ docker run --rm -it public.ecr.aws/aws-cli/aws-cli --version  
aws-cli/2.15.30 Python/3.7.3 Linux/4.9.184-linuxkit botocore/2.4.5dev10
```

### Docker Hub

```
$ docker run --rm -it amazon/aws-cli --version  
aws-cli/2.15.30 Python/3.7.3 Linux/4.9.184-linuxkit botocore/2.4.5dev10
```

- `--rm` – 指定在命令退出后清理容器。



- `-it` – 指定要使用 打开伪 TTY。stdin 这使您能够在 AWS CLI 版本 2 在容器中运行时为其提供输入，例如，使用 `aws configure` 和 `aws help` 命令。当选择是否省略 `-it` 时，请考虑以下事项：
  - 如果您正在运行脚本，则不需要 `-it`。
  - 如果您的脚本遇到错误，则从 Docker 调用中省略 `-it` 也许能解决此问题。
  - 如果您正在尝试通过管道发送输出，`-it` 可能会导致错误，从 Docker 调用中省略 `-it` 也许能解决此问题。如果希望保留 `-it` 标志，但仍想通过管道发送输出，请禁用 AWS CLI 默认使用的[客户端分页程序](#)，应该能解决此问题。

有关 `docker run` 命令的更多信息，请参阅 [Docker 参考指南](#)。

## 有关官方映像的接口和向后兼容性的说明

- 映像上支持的唯一工具是 AWS CLI。仅应直接运行 `aws` 可执行文件。例如，尽管 `less` 和 `groff` 已明确安装在映像上，但不应在 AWS CLI 命令之外直接执行它们。
- `/aws` 工作目录由用户控制。除非用户在运行 AWS CLI 命令时发出指示，否则图像不会写入此目录。
- 依赖最新标签并不能保证向后兼容。为了确保向后兼容，您必须固定使用特定的 `<major.minor.patch>` 标签，因为这些标签是不可改变的；它们只会被推送一次。

## 使用特定版本和标签

官方 AWS CLI 版本 2 镜像有多个版本可供您使用，首先是版本 2.0.6。要运行版本 2 的特定 AWS CLI 版本，请在 `docker run` 命令中附加相应的标签。首次使用带有标签的 `docker run` 命令时，该标签的最新映像会下载到您的计算机。带有该标签的 `docker run` 命令的每次后续使用都从您的本地副本运行。

您可以使用两种类型的标签：

- `latest`— 为图像定义 AWS CLI 版本 2 的最新版本。我们建议您在需要 AWS CLI 版本 2 的最新版本时使用 `latest` 标签。但是，依赖此标签时不保证向后兼容性。默认情况下，在 `latest` 命令中使用 `docker run` 标签。要显式使用 `latest` 标记，请将标签附加到容器映像名称。

Amazon ECR Public

```
$ docker run --rm -it public.ecr.aws/aws-cli/aws-cli:latest command
```

## Docker Hub

```
$ docker run --rm -it amazon/aws-cli:latest command
```

- <major.minor.patch>— 为图像定义 AWS CLI 版本 2 的特定版本。如果您计划在生产环境中使用官方镜像，我们建议您使用版本 2 的特定 AWS CLI 版本，以确保向后兼容。例如，要运行版本 2.0.6，请将版本附加到容器映像名称中。

## Amazon ECR Public

```
$ docker run --rm -it public.ecr.aws/aws-cli/aws-cli:2.0.6 command
```

## Docker Hub

```
$ docker run --rm -it amazon/aws-cli:2.0.6 command
```

## 更新到最新的官方映像

由于最新的映像仅在您第一次使用 `docker run` 命令时下载到您的计算机，因此您需要手动提取更新的映像。要手动更新到最新版本，我们建议您拉取标记为 `latest` 的映像。拉取映像会将最新版本下载到您的计算机。

## Amazon ECR Public

```
$ docker pull public.ecr.aws/aws-cli/aws-cli:latest
```

## Docker Hub

```
$ docker pull amazon/aws-cli:latest
```

## 共享主机文件、凭据、环境变量和配置

由于 AWS CLI 版本 2 是在容器中运行的，因此默认情况下，CLI 无法访问主机文件系统，其中包括配置和凭据。要将主机文件系统、凭证和配置共享到容器，请将主机系统的 `~/.aws` 目录挂载到位于 `/root/.aws` 的容器，并将 `-v` 标志附加到 `docker run` 命令。这允许在容器中运行的 AWS CLI 版本 2 查找主机文件信息。

## Amazon ECR Public

### Linux 和 macOS

```
$ docker run --rm -it -v ~/.aws:/root/.aws public.ecr.aws/aws-cli/aws-cli command
```

### Windows 命令提示符

```
$ docker run --rm -it -v %userprofile%\aws:/root/.aws public.ecr.aws/aws-cli/aws-cli command
```

### 窗口 PowerShell

```
C:\> docker run --rm -it -v $env:userprofile\aws:/root/.aws public.ecr.aws/aws-cli/aws-cli command
```

## Docker Hub

### Linux 和 macOS

```
$ docker run --rm -it -v ~/.aws:/root/.aws amazon/aws-cli command
```

### Windows 命令提示符

```
$ docker run --rm -it -v %userprofile%\aws:/root/.aws amazon/aws-cli command
```

### 窗口 PowerShell

```
C:\> docker run --rm -it -v $env:userprofile\aws:/root/.aws amazon/aws-cli command
```

有关 `-v` 标志和挂载的更多信息，请参阅 [Docker 参考指南](#)。

#### Note

有关 `config` 和 `credentials` 文件的信息，请参阅 [the section called “配置和凭证文件设置”](#)。

## 示例 1：提供凭证和配置

在此示例中，我们在运行 `s3 ls` 命令时提供主机凭证和配置，以便列出 Amazon Simple Storage Service (Amazon S3) 中的存储桶。以下示例使用 AWS CLI 证书和配置文件的默认位置，要使用不同的位置，请更改文件路径。

### Amazon ECR Public

#### Linux 和 macOS

```
$ docker run --rm -it -v ~/.aws:/root/.aws public.ecr.aws/aws-cli/aws-cli s3 ls
2020-03-25 00:30:48 aws-cli-docker-demo
```

#### Windows 命令提示符

```
$ docker run --rm -it -v %userprofile%\..aws:/root/.aws public.ecr.aws/aws-cli/aws-
cli s3 ls
2020-03-25 00:30:48 aws-cli-docker-demo
```

#### 窗口 PowerShell

```
C:\> docker run --rm -it -v $env:userprofile\..aws:/root/.aws public.ecr.aws/aws-cli/
aws-cli s3 ls
```

### Docker Hub

#### Linux 和 macOS

```
$ docker run --rm -it -v ~/.aws:/root/.aws amazon/aws-cli s3 ls
2020-03-25 00:30:48 aws-cli-docker-demo
```

#### Windows 命令提示符

```
$ docker run --rm -it -v %userprofile%\..aws:/root/.aws amazon/aws-cli s3 ls
2020-03-25 00:30:48 aws-cli-docker-demo
```

#### 窗口 PowerShell

```
C:\> docker run --rm -it -v $env:userprofile\..aws:/root/.aws amazon/aws-cli s3 ls
```

您可以使用 `-e` 标志调用特定系统的环境变量。要使用环境变量，请按名称调用它。

## Amazon ECR Public

### Linux 和 macOS

```
$ docker run --rm -it -v ~/.aws:/root/.aws -e ENVVAR_NAME public.ecr.aws/aws-cli/
aws-cli s3 ls
2020-03-25 00:30:48 aws-cli-docker-demo
```

### Windows 命令提示符

```
$ docker run --rm -it -v %userprofile%\aws:/root/.aws -e ENVVAR_NAME
public.ecr.aws/aws-cli/aws-cli s3 ls
2020-03-25 00:30:48 aws-cli-docker-demo
```

### 窗口 PowerShell

```
C:\> docker run --rm -it -v $env:userprofile\aws:/root/.aws -e ENVVAR_NAME
public.ecr.aws/aws-cli/aws-cli s3 ls
```

## Docker Hub

### Linux 和 macOS

```
$ docker run --rm -it -v ~/.aws:/root/.aws -e ENVVAR_NAME amazon/aws-cli s3 ls
2020-03-25 00:30:48 aws-cli-docker-demo
```

### Windows 命令提示符

```
$ docker run --rm -it -v %userprofile%\aws:/root/.aws -e ENVVAR_NAME amazon/aws-cli
s3 ls
2020-03-25 00:30:48 aws-cli-docker-demo
```

### 窗口 PowerShell

```
C:\> docker run --rm -it -v $env:userprofile\aws:/root/.aws -e ENVVAR_NAME amazon/
aws-cli s3 ls
```

## 示例 2：将 Amazon S3 文件下载到您的主机系统

对于某些 AWS CLI 版本 2 命令，您可以从容器中的主机系统读取文件或将文件从容器写入主机系统。

在此示例中，我们通过将当前工作目录挂载到容器的 S3 目录，将 `s3://aws-cli-docker-demo/hello` 对象下载到本地文件系统。通过将 `hello` 对象下载到容器的 `/aws` 目录，文件也会保存到主机系统的当前工作目录中。

### Amazon ECR Public

#### Linux 和 macOS

```
$ docker run --rm -it -v ~/.aws:/root/.aws -v $(pwd):/aws public.ecr.aws/aws-cli/aws-cli s3 cp s3://aws-cli-docker-demo/hello .
download: s3://aws-cli-docker-demo/hello to ./hello
```

#### Windows 命令提示符

```
$ docker run --rm -it -v %userprofile%.aws:/root/.aws -v %cd%:/aws public.ecr.aws/aws-cli/aws-cli s3 cp s3://aws-cli-docker-demo/hello .
download: s3://aws-cli-docker-demo/hello to ./hello
```

#### 窗口 PowerShell

```
C:\> docker run --rm -it -v $env:userprofile\.aws:/root/.aws -v $pwd\aws:/aws public.ecr.aws/aws-cli/aws-cli s3 cp s3://aws-cli-docker-demo/hello .
```

### Docker Hub

#### Linux 和 macOS

```
$ docker run --rm -it -v ~/.aws:/root/.aws -v $(pwd):/aws amazon/aws-cli s3 cp s3://aws-cli-docker-demo/hello .
download: s3://aws-cli-docker-demo/hello to ./hello
```

#### Windows 命令提示符

```
$ docker run --rm -it -v %userprofile%.aws:/root/.aws -v %cd%:/aws amazon/aws-cli s3 cp s3://aws-cli-docker-demo/hello .
download: s3://aws-cli-docker-demo/hello to ./hello
```

## 窗口 PowerShell

```
C:\> docker run --rm -it -v $env:userprofile\.aws:/root/.aws -v $pwd\aws:/aws
amazon/aws-cli s3 cp s3://aws-cli-docker-demo/hello .
```

要确认本地文件系统中存在下载的文件，请运行以下命令。

## Linux 和 macOS

```
$ cat hello
Hello from Docker!
```

## 窗口 PowerShell

```
$ type hello
Hello from Docker!
```

## 示例 3：使用您的 AWS\_PROFILE 环境变量

您可以使用 `-e` 标志调用特定系统的环境变量。调用您想使用的每个环境变量。在本示例中，我们在运行 `s3 ls` 命令以在 Amazon Simple Storage Service (Amazon S3) 中列出存储桶时提供了主机凭证、配置和 `AWS_PROFILE` 环境变量。

## Amazon ECR Public

### Linux 和 macOS

```
$ docker run --rm -it -v ~/.aws:/root/.aws -e AWS_PROFILE public.ecr.aws/aws-cli/
aws-cli s3 ls
2020-03-25 00:30:48 aws-cli-docker-demo
```

### Windows 命令提示符

```
$ docker run --rm -it -v %userprofile%\.aws:/root/.aws -e AWS_PROFILE
public.ecr.aws/aws-cli/aws-cli s3 ls
2020-03-25 00:30:48 aws-cli-docker-demo
```

## 窗口 PowerShell

```
C:\> docker run --rm -it -v $env:userprofile\.aws:/root/.aws -e AWS_PROFILE
public.ecr.aws/aws-cli/aws-cli s3 ls
```

## Docker Hub

### Linux 和 macOS

```
$ docker run --rm -it -v ~/.aws:/root/.aws -e AWS_PROFILE amazon/aws-cli s3 ls
2020-03-25 00:30:48 aws-cli-docker-demo
```

### Windows 命令提示符

```
$ docker run --rm -it -v %userprofile%.aws:/root/.aws -e AWS_PROFILE amazon/aws-cli
s3 ls
2020-03-25 00:30:48 aws-cli-docker-demo
```

### 窗口 PowerShell

```
C:\> docker run --rm -it -v $env:userprofile\.aws:/root/.aws -e AWS_PROFILE amazon/
aws-cli s3 ls
```

## 缩短 docker run 命令

为了缩短 `docker run` 命令，我们建议您使用操作系统的功能在 Linux 和 macOS 中创建 [symbolic link](#)（符号链接）或 [alias](#)，或者在 Windows 中创建 [doskey](#)。要设置 `aws` 别名，可以运行以下命令之一。

- 要对 `aws` 命令进行基本访问，请运行以下命令。

### Amazon ECR Public

#### Linux 和 macOS

```
$ alias aws='docker run --rm -it public.ecr.aws/aws-cli/aws-cli'
```

#### Windows 命令提示符

```
C:\> doskey aws=docker run --rm -it public.ecr.aws/aws-cli/aws-cli $*
```



## 窗口 PowerShell

```
C:\> Function AWSCLI {docker run --rm -it public.ecr.aws/aws-cli/aws-cli $args}
Set-Alias -Name aws -Value AWSCLI
```

## Docker Hub

### Linux 和 macOS

```
$ alias aws='docker run --rm -it amazon/aws-cli'
```

### Windows 命令提示符

```
C:\> doskey aws=docker run --rm -it amazon/aws-cli $*
```

## 窗口 PowerShell

```
C:\> Function AWSCLI {docker run --rm -it amazon/aws-cli $args}
Set-Alias -Name aws -Value AWSCLI
```

- 要在使用 aws 命令时访问主机文件系统和配置设置，请运行以下命令：

## Amazon ECR Public

### Linux 和 macOS

```
$ alias aws='docker run --rm -it -v ~/.aws:/root/.aws -v $(pwd):/aws
public.ecr.aws/aws-cli/aws-cli'
```

### Windows 命令提示符

```
C:\> doskey aws=docker run --rm -it -v %userprofile%\aws:/root/.aws -v %cd%:/aws
public.ecr.aws/aws-cli/aws-cli $*
```

## 窗口 PowerShell

```
C:\> Function AWSCLI {docker run --rm -it -v $env:userprofile\aws:/root/.aws -v
$pwd\aws:/aws public.ecr.aws/aws-cli/aws-cli $args}
Set-Alias -Name aws -Value AWSCLI
```

## Docker Hub

### Linux 和 macOS

```
$ alias aws='docker run --rm -it -v ~/.aws:/root/.aws -v $(pwd):/aws amazon/aws-cli'
```

### Windows 命令提示符

```
C:\> doskey aws=docker run --rm -it -v %userprofile%.aws:/root/.aws -v %cd%:/aws amazon/aws-cli $*
```

### 窗口 PowerShell

```
C:\> Function AWSCLI {docker run --rm -it -v $env:userprofile\.aws:/root/.aws -v $pwd\aws:/aws amazon/aws-cli $args}
Set-Alias -Name aws -Value AWSCLI
```

- 要分配要在 aws 别名中使用的特定版本，请附加版本标签。

## Amazon ECR Public

### Linux 和 macOS

```
$ alias aws='docker run --rm -it -v ~/.aws:/root/.aws -v $(pwd):/aws public.ecr.aws/aws-cli/aws-cli:2.0.6'
```

### Windows 命令提示符

```
C:\> doskey aws=docker run --rm -it -v %userprofile%.aws:/root/.aws -v %cd%:/aws public.ecr.aws/aws-cli/aws-cli:2.0.6 $*
```

### 窗口 PowerShell

```
C:\> Function AWSCLI {docker run --rm -it -v $env:userprofile\.aws:/root/.aws -v $pwd\aws:/aws public.ecr.aws/aws-cli/aws-cli:2.0.6 $args}
Set-Alias -Name aws -Value AWSCLI
```

## Docker Hub

### Linux 和 macOS

```
$ alias aws='docker run --rm -it -v ~/.aws:/root/.aws -v $(pwd):/aws amazon/aws-cli:2.0.6'
```

### Windows 命令提示符

```
C:\> doskey aws=docker run --rm -it -v %userprofile%\aws:/root/.aws -v %cd%:/aws amazon/aws-cli:2.0.6 $*
```

### 窗口 PowerShell

```
C:\> Function AWSCLI {docker run --rm -it -v $env:userprofile\aws:/root/.aws -v $pwd\aws:/aws amazon/aws-cli:2.0.6 $args}
Set-Alias -Name aws -Value AWSCLI
```

设置别名后，您可以像安装在主机系统上一样从容器中运行 AWS CLI 版本 2。

```
$ aws --version
aws-cli/2.15.30 Python/3.7.3 Linux/4.9.184-linuxkit botocore/2.4.5dev10
```

## 设置 AWS CLI

本主题介绍如何快速配置 AWS Command Line Interface (AWS CLI) 用于交互的基本设置 AWS。其中包括您的安全证书、默认输出格式和默认 AWS 区域。

### 主题

- [收集您的凭证信息以进行编程访问](#)
- [设置新的配置和凭证](#)
- [使用现有配置和凭证文件](#)

## 收集您的凭证信息以进行编程访问

如果您想在 AWS 外部进行交互，则需要编程访问权限 AWS Management Console。对于身份验证和凭证说明，请选择以下选项之一：

身份验证类型	用途	说明
IAM 身份中心员工用户短期证书	<p>( 推荐 ) 为 IAM Identity Center 员工用户使用短期证书。</p> <p>安全最佳实践是与 IAM 身份中心 AWS Organizations 配合使用。它将短期证书与用户目录 ( 例如内置的 IAM 身份中心目录或 Active Directory ) 结合在一起。</p>	<a href="#">the section called “IAM Identity Center 身份验证”</a>
IAM 用户短期证书	使用 IAM 用户短期证书，它比长期证书更安全。如果您的凭证遭到泄露，则在证书过期之前的使用时间有限。	<a href="#">the section called “短期凭证”</a>
Amazon EC2 实例上的 IAM 或 IAM 身份中心用户。	使用分配给 Amazon EC2 实例的角色，使用 Amazon EC2 实例元数据查询临时证书。	<a href="#">the section called “将凭证用于 Amazon EC2 实例元数据”</a>
扮演角色以获得权限	将另一种凭证方法配对，并扮演一个角色来临时访问 AWS 服务 您的用户可能没有访问权限。	<a href="#">the section called “IAM 角色”</a>
IAM 用户长期证书	( 不推荐 ) 使用没有有效期的长期证书。	<a href="#">the section called “IAM 用户”</a>
IAM 或 IAM 身份中心员工用户的外部存储	( 不推荐 ) 将其他凭证方法配对，但要凭证值存储在以外的位置。AWS CLI此方法仅	<a href="#">the section called “外部凭证”</a>

身份验证类型	用途	说明
	与存储凭据的外部位置一样安全。	

## 设置新的配置和凭证

将您的配置和凭据信息 AWS CLI 存储在 `credentials` 和文件中的配置 `config` 文件（设置集合）中。

可通过两大方式进行快速设置：

- [使用 AWS CLI 命令进行配置](#)
- [手动编辑凭证和配置文件](#)

以下示例将示例值用于每种身份验证方法。请将示例值替换为您自己的值。

### 使用 AWS CLI 命令进行配置

对于一般用途，在您的首选终端中使用 `aws configure` 或 `aws configure sso` 命令是设置 AWS CLI 安装的最快方法。根据您的首选的凭证方法，AWS CLI 系统会提示您输入相关信息。默认情况下，当您运行未明确指定要使用的配置文件的 AWS CLI 命令时，将使用此配置文件中的信息。

有关 `credentials` 和 `config` 文件的更多信息，请参阅 [配置和凭证文件设置](#)。

### IAM Identity Center (SSO)

此示例用于 AWS IAM Identity Center 使用向 `aws configure sso` 导。有关更多信息，请参阅 [the section called “配置自动令牌刷新”](#)。

```
$ aws configure sso
SSO session name (Recommended): my-sso
SSO start URL [None]: https://my-sso-portal.awsapps.com/start
SSO region [None]: us-east-1

Attempting to automatically open the SSO authorization page in your default browser.

There are 2 AWS accounts available to you.
> DeveloperAccount, developer-account-admin@example.com (111122223333)
  ProductionAccount, production-account-admin@example.com (444455556666)
```

```
Using the account ID 111122223333

There are 2 roles available to you.
> ReadOnly
  FullAccess

Using the role name "ReadOnly"

CLI default client Region [None]: us-west-2
CLI default output format [None]: json
CLI profile name [123456789011_ReadOnly]: user1
```

## IAM Identity Center (Legacy SSO)

此示例适用于 AWS IAM Identity Center 使用 `aws configure sso` 向导的传统方法。要使用传统 SSO，请将会话名称留空。有关更多信息，请参阅 [the section called “配置旧版不可刷新”](#)。

```
$ aws configure sso
SSO session name (Recommended):
SSO start URL [None]: https://my-sso-portal.awsapps.com/start
SSO region [None]:us-east-1

SSO authorization page has automatically been opened in your default browser.
Follow the instructions in the browser to complete this authorization request.

There are 2 AWS accounts available to you.
> DeveloperAccount, developer-account-admin@example.com (111122223333)
  ProductionAccount, production-account-admin@example.com (444455556666)

Using the account ID 111122223333

There are 2 roles available to you.
> ReadOnly
  FullAccess

Using the role name "ReadOnly"

CLI default client Region [None]: us-west-2
CLI default output format [None]: json
CLI profile name [123456789011_ReadOnly]: user1
```



## Long-term credentials

### Warning

为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证，而是使用与身份提供商的联合身份验证，例如 [AWS IAM Identity Center](#)。

此示例介绍来自 AWS Identity and Access Management 的长期凭证。有关更多信息，请参阅 [the section called “IAM 用户”](#)。

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

有关身份验证和凭证方法的更多详细信息，请参阅 [身份验证和访问凭证](#)。

## 手动编辑凭证和配置文件

在复制和粘贴信息时，我们建议手动编辑 config 和 credentials 文件。根据您的首选的凭证方法，将通过不同的方式设置文件。

这些文件存储在您的主目录下的 .aws 文件夹中。主目录位置因操作系统而异，但在 Windows 中使用环境变量 %UserProfile% 引用，在基于 Unix 的系统中使用 \$HOME 或 ~ (波浪符) 引用。有关这些设置的存储位置的更多信息，请参阅 [the section called “配置设置存储在何处？”](#)。

以下示例显示了一个 default 配置文件和一个名为 user1 的配置文件，并且将使用示例值。请将示例值替换为您自己的值。有关 credentials 和 config 文件的更多信息，请参阅 [配置和凭证文件设置](#)。

## IAM Identity Center (SSO)

这个例子是为 AWS IAM Identity Center 了。有关更多信息，请参阅 [the section called “配置自动令牌刷新”](#)。

### 凭证文件

该 credentials 文件不用于此身份验证方法。



## Config 文件

```
[default]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = readOnly
region = us-west-2
output = text

[profile user1]
sso_session = my-sso
sso_account_id = 444455556666
sso_role_name = readOnly
region = us-east-1
output = json

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://my-sso-portal.awsapps.com/start
sso_registration_scopes = sso:account:access
```

## IAM Identity Center (Legacy SSO)

此示例适用于的传统方法 AWS IAM Identity Center。有关更多信息，请参阅 [the section called “配置旧版不可刷新”](#)。

### 凭证文件

该 `credentials` 文件不用于此身份验证方法。

### Config 文件

```
[default]
sso_start_url = https://my-sso-portal.awsapps.com/start
sso_region = us-east-1
sso_account_id = 111122223333
sso_role_name = readOnly
region = us-west-2
output = text

[profile user1]
sso_start_url = https://my-sso-portal.awsapps.com/start
sso_region = us-east-1
```





```
region=us-east-1
output=text
```

## Long-term credentials

### Warning

为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证，而是使用与身份提供商的联合身份验证，例如 [AWS IAM Identity Center](#)。

此示例介绍来自 AWS Identity and Access Management 的长期凭证。有关更多信息，请参阅 [the section called “IAM 用户”](#)。

### 凭证文件

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

[user1]
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
aws_secret_access_key=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
```

### Config 文件

```
[default]
region=us-west-2
output=json

[profile user1]
region=us-east-1
output=text
```

有关身份验证和凭证方法的更多详细信息，请参阅 [身份验证和访问凭证](#)。

## 使用现有配置和凭证文件

如果您现在拥有配置和凭证文件，这些文件可用于 AWS CLI。

要使用 `config` 和 `credentials` 文件，请将它们移至主目录中名为 `.aws` 的文件夹中。主目录位置因操作系统而异，但在 Windows 中使用环境变量 `%UserProfile%` 引用，在基于 Unix 的系统中使用 `$HOME` 或 `~`（波浪符）引用。

通过将 `AWS_CONFIG_FILE` 和 `AWS_SHARED_CREDENTIALS_FILE` 环境变量设置为另一个本地路径，可以为 `config` 和 `credentials` 文件指定非原定设置位置。有关详细信息，请参阅 [用于配置的环境变量 AWS CLI](#)。

有关配置和凭证文件的更多详细信息，请参阅 [the section called “配置和凭证文件设置”](#)。

## 配置 AWS CLI

本节介绍如何配置 AWS Command Line Interface (AWS CLI) 用于与之交互的设置 AWS。这些功能包括：

- 凭证将标识 API 的调用方。访问凭证用于对 AWS 服务器的请求进行加密，以确认您的身份并检索相关的权限策略。这些权限将决定您可以执行的操作。有关如何设置凭证的信息，请参阅 [身份验证和访问凭证](#)。
- 用于说明 AWS CLI 如何处理请求@@ 的其他配置详细信息，例如默认输出格式和默认 AWS 区域。

### Note

AWS 要求所有传入的请求都经过加密签名。他们为你 AWS CLI 做这件事。“签名”包括日期/时间戳。因此，您必须确保正确设置电脑的日期和时间。如果您不这样做，并且签名中的日期/时间与 AWS 服务识别的日期/时间相差太远，AWS 则会拒绝该请求。

## 配置和凭证优先顺序

凭据和配置设置位于多个位置，例如系统或用户环境变量、本地 AWS 配置文件或在命令行中明确声明为参数。某些位置优先于其他位置。AWS CLI 凭据和配置设置的优先顺序如下：

1. [命令行选项](#) – 覆盖任何其他位置的设置，例如 `--region`、`--output` 和 `--profile` 参数。
2. [环境变量](#) – 您可以在系统的环境变量中存储值。
3. [代入角色](#) – 通过配置或 `aws sts assume-role` 命令代入 IAM 角色的权限。
4. [使用 Web 身份代入角色](#) – 通过配置或 `aws sts assume-role` 命令使用 Web 身份代入 IAM 角色的权限。
5. [AWS IAM Identity Center](#)— 当您运行 `aws configure sso` 命令时，存储在 `config` 文件中的 IAM Identity Center 配置设置会更新。然后在您运行 `aws sso login` 命令时对凭据进行身份验证。`config` 文件位于 `~/.aws/config` (在 Linux 或 macOS 上) 或 `C:\Users\USERNAME\.aws\config` (在 Windows 上)。
6. [凭证文件](#) – 在运行命令 `aws configure` 时，将更新 `credentials` 和 `config` 文件。`credentials` 文件位于 `~/.aws/credentials` (在 Linux 或 macOS 上) 或 `C:\Users\USERNAME\.aws\credentials` (在 Windows 上)。

7. [自定义流程](#) – 从外部来源获取您的凭证。
8. [配置文件](#) – 在运行命令 `aws configure` 时，将更新 `credentials` 和 `config` 文件。`config` 文件位于 `~/.aws/config` (在 Linux 或 macOS 上) 或 `C:\Users\USERNAME\.aws\config` (在 Windows 上)。
9. [容器凭证](#) – 您可以将 IAM 角色与每个 Amazon Elastic Container Service (Amazon ECS) 任务定义关联。之后，该任务的容器就可以使用该角色的临时凭证。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的[任务的 IAM 角色](#)。
10. [Amazon EC2 实例配置文件凭证](#) – 您可以将 IAM 角色与每个 Amazon Elastic Compute Cloud (Amazon EC2) 实例关联。之后，在该实例上运行的代码就可以使用该角色的临时凭证。凭证通过 Amazon EC2 元数据服务提供。有关更多信息，请参阅[Amazon EC2 用户指南中的 Amazon EC2 的 IAM 角色](#)和 IAM 用户指南中的[使用实例配置文件](#)。

## 此部分中的其他主题

- [the section called “配置和凭证文件设置”](#)
- [the section called “环境变量”](#)
- [the section called “命令行选项”](#)
- [the section called “命令完成”](#)
- [the section called “重试”](#)
- [the section called “使用 HTTP 代理”](#)

## 配置和凭证文件设置

您可以将常用的配置设置和凭证保存在由 AWS CLI 维护的文件中。

这些文件将分成 `profiles`。默认情况下，AWS CLI 使用名为 `default` 的配置文件中的设置。要使用备用设置，您可以创建和引用其他配置文件。

您可以通过设置某个支持的环境变量或使用命令行参数来覆盖个别设置。有关配置设置优先顺序的更多信息，请参阅[配置 AWS CLI](#)。

### Note

有关如何设置凭证的信息，请参阅 [身份验证和访问凭证](#)。

## 主题

- [配置和凭证文件的格式](#)
- [配置设置存储在何处？](#)
- [使用命名配置文件](#)
- [使用命令设置和查看配置设置](#)
- [设置新的配置和凭证命令示例](#)
- [支持的 config 文件设置](#)

## 配置和凭证文件的格式

config 和 credentials 文件将归入各个节中。节包括 profiles、sso-sessions 和 services。节是一个命名的设置集合，它一直持续到遇到另一个节定义行为止。可将多个配置文件和节存储在 config 和 credentials 文件中。

这些文件是使用以下格式的纯文本文件：

- 节名称用方括号 [] 括起来，例如 [default]、[profile *user1*] 和 [sso-session]。
- 节中的所有条目均采用 setting\_name=value 的一般形式。
- 可以通过以井号字符 (#) 开头来注释掉行。

config 和 credentials 文件包含以下节类型：

- [节类型：profile](#)
- [节类型：sso-session](#)
- [节类型：services](#)

### 节类型：**profile**

这些 AWS CLI 商店

配置文件节名称使用以下格式，具体取决于文件：

- Config 文件：[default] [profile *user1*]
- 凭证文件：[default] [*user1*]

在 credentials 文件中创建条目时，请勿使用单词 profile。



每个配置文件都可以指定不同的凭证，还可以指定不同的 AWS 区域和输出格式。在 config 文件中命名配置文件时，请包括前缀词“profile”，但不要将它包括在 credentials 文件中。

以下示例显示指定了两个配置文件、区域和输出的 credentials 和 config 文件。第一个 [default] 配置文件在运行未指定配置文件的 AWS CLI 命令时使用。当你运行带有 `--profile user1` 参数的 AWS CLI 命令时，会使用第二个。

## IAM Identity Center (SSO)

这个例子是为 AWS IAM Identity Center 了。有关更多信息，请参阅 [the section called “配置自动令牌刷新”](#)。

### 凭证文件

该 credentials 文件不用于此身份验证方法。

### Config 文件

```
[default]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = readOnly
region = us-west-2
output = text

[profile user1]
sso_session = my-sso
sso_account_id = 444455556666
sso_role_name = readOnly
region = us-east-1
output = json

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://my-sso-portal.awsapps.com/start
sso_registration_scopes = sso:account:access
```

## IAM Identity Center (Legacy SSO)

此示例适用于的传统方法 AWS IAM Identity Center。有关更多信息，请参阅 [the section called “配置旧版不可刷新”](#)。

### 凭证文件





该 `credentials` 文件不用于此身份验证方法。

## Config 文件

```
[default]
role_arn=arn:aws:iam::123456789012:role/defaultrole
credential_source=Ec2InstanceMetadata
region=us-west-2
output=json

[profile user1]
role_arn=arn:aws:iam::777788889999:role/user1role
credential_source=Ec2InstanceMetadata
region=us-east-1
output=text
```

## Long-term credentials

### Warning

为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证，而是使用与身份提供商的联合身份验证，例如 [AWS IAM Identity Center](#)。

此示例介绍来自 AWS Identity and Access Management 的长期凭证。有关更多信息，请参阅 [the section called “IAM 用户”](#)。

## 凭证文件

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

[user1]
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
aws_secret_access_key=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
```

## Config 文件

```
[default]
region=us-west-2
output=json
```

```
[profile user1]
region=us-east-1
output=text
```

有关更多信息以及其他授权和凭证方法，请参阅 [the section called “IAM 用户”](#)。

## 节类型：**sso-session**

该config文件的sso-session部分用于对用于获取 SSO 访问令牌的配置变量进行分组，然后可以使用这些变量来获取 AWS 凭证。使用了以下设置：

- (必需) [sso\\_start\\_url](#)
- (必需) [sso\\_region](#)
- [sso\\_account\\_id](#)
- [sso\\_role\\_name](#)
- [sso\\_registration\\_scopes](#)

您定义一个 sso-session 部分并将其关联到配置文件。必须在 sso-session 部分内设置 sso\_region 和 sso\_start\_url。通常，sso\_account\_id 和 sso\_role\_name 必须在 profile 部分进行设置，以便 SDK 可以请求 SSO 凭证。

以下示例将 SDK 配置为请求 SSO 凭证并支持自动令牌刷新：

```
[profile dev]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://my-sso-portal.awsapps.com/start
```

这也允许在多个配置文件中重复使用 sso-session 配置：

```
[profile dev]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole
```

```
[profile prod]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole2

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://my-sso-portal.awsapps.com/start
```

但是，并不是所有 SSO 令牌配置场景都需要 `sso_account_id` 和 `sso_role_name`。如果您的应用程序仅使用支持所有者身份验证的 AWS 服务，则不需要传统 AWS 凭证。所有者身份验证是一种 HTTP 身份验证方案，它使用称为所有者令牌的安全令牌。在这种情况下，不需要 `sso_account_id` 和 `sso_role_name`。请参阅您的 AWS 服务的个人指南，以确定其是否支持不记名令牌授权。

此外，可以作为 `sso-session` 的一部分配置注册范围。范围是 OAuth 2.0 中的一种机制，用于限制应用程序对用户账户的访问。应用程序可以请求一个或多个范围，向应用程序签发的访问令牌将仅限于授予的范围。这些范围定义了为已注册的 OIDC 客户端请求授权的权限和客户端检索的访问令牌。以下示例将 `sso_registration_scopes` 设置为提供已列出账户/角色的访问权限：

```
[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://my-sso-portal.awsapps.com/start
sso_registration_scopes = sso:account:access
```

身份验证令牌缓存到 `~/.aws/sso/cache` 目录下的磁盘上，文件名基于会话名称。

有关此配置类型的更多信息，请参阅 [the section called “配置自动令牌刷新”](#)。

## 节类型：**services**

该 `services` 部分是一组设置，用于为 AWS 服务请求配置自定义终端节点。然后，配置文件将链接到 `services` 节。

```
[profile dev]
services = my-services
```

`services` 节按 `<SERVICE> =` 行分成多个小节，其中 `<SERVICE>` 是 AWS 服务标识键。标 AWS 服务标识符基于 API 模型，将所有空格 `serviceId` 替换为下划线，并将所有字母小写。有关 `services` 节中要使用的所有服务标识符密钥的列表，请参阅 [在中使用终端节点 AWS CLI](#)。服务标识符密钥后面是嵌套的设置，每个设置单独成行，缩进两个空格。

以下示例会配置端点，用于在 `dev` 配置文件中使用的 `my-services` 节中向服务发出的请求。后面紧跟的任何缩进行都包含在该小节中，并适用于该服务。

```
[profile dev]
services = my-services

[services my-services]
dynamodb =
  endpoint_url = http://localhost:8000
```

有关特定于服务的端点的更多信息，请参阅 [在中使用终端节点 AWS CLI](#)。

如果您的配置文件具有基于角色的凭证，而这些凭证是通过 IAM 代入角色功能的 `source_profile` 参数配置的，则开发工具包仅使用所指定配置文件的服务配置。它不使用关联有角色的配置文件。例如，使用以下共享 config 文件：

```
[profile A]
credential_source = Ec2InstanceMetadata
endpoint_url = https://profile-a-endpoint.aws/

[profile B]
source_profile = A
role_arn = arn:aws:iam::123456789012:role/roleB
services = profileB

[services profileB]
ec2 =
  endpoint_url = https://profile-b-ec2-endpoint.aws
```

如果您使用配置文件 B 并在代码中调用 Amazon EC2，则端点将解析为 `https://profile-b-ec2-endpoint.aws`。如果您的代码向其他任何服务发出请求，则端点解析将不遵循任何自定义逻辑。该端点不会解析到配置文件 A 中定义的全局端点。要使全局端点对配置文件 B 生效，您需要直接在配置文件 B 中设置 `endpoint_url`。

## 配置设置存储在何处？

将您指定的敏感凭证信息 AWS CLI 存储在名为的本地文件 `aws configure` 中 `credentials`，该文件位于您的主目录 `.aws` 中名为的文件夹中。使用 `aws configure` 指定的较不敏感的配置选项存储在名为 `config` 的本地文件中，该文件也存储在主目录的 `.aws` 文件夹中。

### 在 config 文件中存储凭证

您可以将所有配置文件设置保存在一个文件中，因为他们 AWS CLI 可以从 config 文件中读取凭据。如果两个文件中都有共享相同名称的配置文件的凭证，则凭证文件中的密钥优先。我们建议将凭证保存在 `credentials` 文件中。这些文件也被各种语言软件开发工具包 (SDK) 使用。如果您除使用以外的其他软件开发工具包之一 AWS CLI，请确认证书是否应存储在自己的文件中。

主目录位置因操作系统而异，但在 Windows 中使用环境变量 `%UserProfile%` 引用，在基于 Unix 的系统中使用 `$HOME` 或 `~` ( 波浪符 ) 引用。通过将 `AWS_CONFIG_FILE` 和 `AWS_SHARED_CREDENTIALS_FILE` 环境变量设置为另一个本地路径，可以为文件指定非默认位置。有关详细信息，请参阅 [用于配置的环境变量 AWS CLI](#)。

当您使用指定 AWS Identity and Access Management (IAM) 角色的共享配置文件时，会 AWS CLI 调用该 AWS STS `AssumeRole` 操作来检索临时证书。随后，这些凭证将存储起来 ( 存储在 `~/.aws/cli/cache` 中 )。后续 AWS CLI 命令使用缓存的临时证书，直到它们过期，此时会 AWS CLI 自动刷新证书。

## 使用命名配置文件

如果未明确定义配置文件，则使用 `default` 配置文件。

要使用命名配置文件，请向您的命令添加 `--profile profile-name` 选项。以下示例列出了使用 `user1` 配置文件中定义的凭证和设置的所有 Amazon EC2 实例。

```
$ aws ec2 describe-instances --profile user1
```

要为多个命令使用一个命名配置文件，可以通过将 `AWS_PROFILE` 环境变量设置为默认配置文件来避免在每个命令中指定配置文件。您可以使用 `--profile` 参数来覆盖此设置。

### Linux or macOS

```
$ export AWS_PROFILE=user1
```

### Windows

```
C:\> setx AWS_PROFILE user1
```



使用 `set` 设置环境变量会更改使用的值，直到当前命令提示符会话结束，或者直到您将该变量设置为其他值。

使用 `setx` 设置环境变量会更改运行命令后创建的所有命令 Shell 中的值。这不会影响运行命令时已在运行的任何命令 Shell。关闭并重新启动命令 Shell 可查看这一更改的效果。

设置环境变量会更改默认配置文件，直到 Shell 会话结束或直到您将该变量设置为其他值。通过将环境变量放在 shell 的启动脚本中，可使环境变量在未来的会话中继续有效。有关更多信息，请参阅 [用于配置的环境变量 AWS CLI](#)。

## 使用命令设置和查看配置设置

可通过多种方法使用命令来查看和设置配置设置。

### `aws configure`

运行此命令可快速设置和查看 凭证、区域和输出格式。以下示例显示了示例值。

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

### `aws configure set`

您可以使用 `aws configure set` 设置任何凭证或配置设置。使用 `--profile` 设置指定要查看或修改的配置文件。

例如，以下命令设置名为 `region` 的配置文件中的 `integ`。

```
$ aws configure set region us-west-2 --profile integ
```

要删除某个设置，请使用空字符串作为值，或在文本编辑器中手动从 `config` 和 `credentials` 文件中删除该设置。

```
$ aws configure set cli_pager "" --profile integ
```

## aws configure get

您可以检索已使用 `aws configure get` 设置的任何凭证或配置设置。使用 `--profile` 设置指定要查看或修改的配置文件。

例如，以下命令检索名为 `region` 的配置文件中的 `integ` 设置。

```
$ aws configure get region --profile integ  
us-west-2
```

如果输出为空，不会显式设置该设置，并将使用默认值。

## aws configure import

导入从 IAM Web 控制台生成的 CSV 凭证。这不适用于从 IAM Identity Center 生成的凭证；使用 IAM Identity Center 的客户应使用 `aws configure sso`。将导入一个配置文件名称与用户名匹配的 CSV 文件。CSV 文件必须包含以下标头。

- 用户名称
- 访问密钥 ID
- 秘密访问密钥

### Note

在创建初始密钥对期间，一旦您关闭 `Download .csv file` ( 下载 .csv 文件 ) 对话框，在关闭该对话框之后就无法访问秘密访问密钥了。如果您需要 .csv 文件，则需要自己创建一个包含所需标头和存储的密钥对信息的文件。如果您无法访问密钥对信息，则需要创建新的密钥对。

```
$ aws configure import --csv file://credentials.csv
```

## aws configure list

要列出配置数据，请使用 `aws configure list` 命令。此命令列出配置文件以及用于指定的配置文件的访问密钥、密钥和区域配置信息。对于每个配置项目，它会显示值、检索配置值的位置以及配置变量名称。

例如，如果您在环境变量 `AWS 区域` 中提供，则此命令会显示您配置的区域名称、该值来自环境变量以及环境变量的名称。

对于角色和 IAM Identity Center 等临时凭证方法，此命令显示临时缓存的访问密钥并显示秘密访问密钥。

```
$ aws configure list
      Name                Value                Type    Location
      ----                -
      profile              <not set>           None    None
      access_key           *****ABCD         shared-credentials-file
      secret_key           *****ABCD         shared-credentials-file
      region               us-west-2           env     AWS_DEFAULT_REGION
```

## aws configure list-profiles

要列出所有配置文件名称，请使用 `aws configure list-profiles` 命令。

```
$ aws configure list-profiles
default
test
```

## aws configure sso

运行此命令可快速设置和查看您的 AWS IAM Identity Center 凭证、区域和输出格式。以下示例显示了示例值。

```
$ aws configure sso
SSO session name (Recommended): my-sso
SSO start URL [None]: https://my-sso-portal.awsapps.com/start
SSO region [None]: us-east-1
SSO registration scopes [None]: sso:account:access
```

## aws configure sso-session

运行此命令可在和文件的 `sso-session` 部分快速设置和查看您的 AWS IAM Identity Center 凭证、区域和输出格式。credentials config 以下示例显示了示例值。

```
$ aws configure sso-session
SSO session name: my-sso
SSO start URL [None]: https://my-sso-portal.awsapps.com/start
SSO region [None]: us-east-1
SSO registration scopes [None]: sso:account:access
```

## 设置新的配置和凭证命令示例

以下示例说明如何使用为不同的身份验证方法指定的凭证、区域和输出配置默认配置文件。

### IAM Identity Center (SSO)

此示例用于 AWS IAM Identity Center 使用向 `aws configure sso` 向导。有关更多信息，请参阅 [the section called “配置自动令牌刷新”](#)。

```
$ aws configure sso
SSO session name (Recommended): my-sso
SSO start URL [None]: https://my-sso-portal.awsapps.com/start
SSO region [None]:us-east-1

Attempting to automatically open the SSO authorization page in your default browser.

There are 2 AWS accounts available to you.
> DeveloperAccount, developer-account-admin@example.com (111122223333)
  ProductionAccount, production-account-admin@example.com (444455556666)

Using the account ID 111122223333

There are 2 roles available to you.
> ReadOnly
  FullAccess

Using the role name "ReadOnly"

CLI default client Region [None]: us-west-2
CLI default output format [None]: json
CLI profile name [123456789011_ReadOnly]: user1
```

### IAM Identity Center (Legacy SSO)

此示例适用于 AWS IAM Identity Center 使用 `aws configure sso` 向导的传统方法。要使用传统 SSO，请将会话名称留空。有关更多信息，请参阅 [the section called “配置旧版不可刷新”](#)。

```
$ aws configure sso
SSO session name (Recommended):
SSO start URL [None]: https://my-sso-portal.awsapps.com/start
SSO region [None]:us-east-1

SSO authorization page has automatically been opened in your default browser.
```



```
$ aws configure set region us-west-2
$ aws configure set output json
```

## Amazon EC2 instance metadata credentials

此示例介绍从托管 Amazon EC2 实例元数据中获得的凭证。此过程没有向导，因此，将使用 `aws configure set` 命令设置每个值。有关更多信息，请参阅 [the section called “将凭证用于 Amazon EC2 实例元数据”](#)。

```
$ aws configure set role_arn arn:aws:iam::123456789012:role/defaultrole
$ aws configure set credential_source Ec2InstanceMetadata
$ aws configure set region us-west-2
$ aws configure set output json
```

## Long-term credentials

### Warning

为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证，而是使用与身份提供商的联合身份验证，例如 [AWS IAM Identity Center](#)。

此示例介绍来自 AWS Identity and Access Management 的长期凭证。有关更多信息，请参阅 [the section called “IAM 用户”](#)。

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

## 支持的 `config` 文件设置

### 主题

- [全局设置](#)
- [S3 自定义命令设置](#)

config 文件支持以下设置。将使用指定（或默认）配置文件中列出的值，除非它们被具有相同名称的环境变量或具有相同名称的命令行选项覆盖。有关哪些顺序设置优先的更多信息，请参阅[配置 AWS CLI](#)

## 全局设置

### aws\_access\_key\_id

指定用作证书一部分的 AWS 访问密钥，用于对命令请求进行身份验证。虽然它可以存储在 config 文件中，但我们建议您将其存储在 credentials 文件中。

可以被 AWS\_ACCESS\_KEY\_ID 环境变量覆盖。您不能将访问密钥 ID 指定为命令行选项。

```
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
```

### aws\_secret\_access\_key

指定用作证书一部分的 AWS 密钥，用于对命令请求进行身份验证。虽然它可以存储在 config 文件中，但我们建议您将其存储在 credentials 文件中。

可以被 AWS\_SECRET\_ACCESS\_KEY 环境变量覆盖。您不能将私有访问密钥指定为命令行选项。

```
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

### aws\_session\_token

指定会 AWS 话令牌。只有在手动指定临时安全凭证时才需要会话令牌。虽然它可以存储在 config 文件中，但我们建议您将其存储在 credentials 文件中。

可以被 AWS\_SESSION\_TOKEN 环境变量覆盖。您不能将会话令牌指定为命令行选项。

```
aws_session_token = AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT  
+FvwnKwRc0IfrRh3c/LTo6UDdyJw00vEVPvLXCrrrUtdnniCEXAMPLE/  
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJ0gQs8IZZaIv2BXIa2R40lgk
```

### ca\_bundle

指定用于验证 SSL 证书的 CA 证书捆绑包（具有 .pem 扩展名的文件）。

可以被 [AWS\\_CA\\_BUNDLE](#) 环境变量或 [--ca-bundle](#) 命令行选项覆盖。

```
ca_bundle = dev/apps/ca-certs/cabundle-2019mar05.pem
```

## cli\_auto\_prompt

启用 AWS CLI 版本 2 的自动提示。可以使用两种设置：

- **on** 每次尝试运行 `aws` 命令时都会使用完整的自动提示模式。这包括在完整或不完整的命令之后按 ENTER 键。

```
cli_auto_prompt = on
```

- **on-partial** 使用部分自动提示模式。如果命令不完整或由于客户端验证错误而无法运行，则使用自动提示。如果您已有现有脚本、运行手册，或者如果您希望只对于不熟悉的命令收到自动提示，而不是每个命令都收到提示，则此模式将非常有帮助。

```
cli_auto_prompt = on-partial
```

您可以使用 `aws_cli_auto_prompt` 环境变量或 `--cli-auto-prompt` 和 `--no-cli-auto-prompt` 命令行参数覆盖此设置。

有关 AWS CLI 版本 2 自动提示功能的信息，请参阅[让 AWS CLI 提示您输入命令](#)。

## cli\_binary\_format

指定 AWS CLI 版本 2 如何解释二进制输入参数。它可能为下列值之一：

- **base64** – 这是默认值。作为二进制大型对象 (BLOB) 键入的输入参数接受 base64 编码字符串。要传递真实的二进制内容，请将内容放在文件中，并提供文件的路径和名称，将 `fileb://` 前缀作为参数值。要传递文件中包含的 base64 编码文本，请提供文件的路径和名称，并将 `file://` 前缀作为参数值。
- **raw-in-base64** 输出- AWS CLI 版本 1 的默认值。如果设置的值为 `raw-in-base64-out`，则使用 `file://` 前缀引用的文件会作为文本进行读取，然后 AWS CLI 尝试将其编码为二进制。

此条目没有等效的环境变量。您可以使用 `--cli-binary-format raw-in-base64-out` 参数在单个命令上指定此值。

```
cli_binary_format = raw-in-base64-out
```

如果您使用 `fileb://` 前缀表示法在文件中引用二进制值，则 AWS CLI 始终期望该文件包含原始二进制内容，并且不会尝试转换该值。



如果您使用 `file://` 前缀表示法在文件中引用二进制值，则会根据当前 `cli_binary_format` 设置 AWS CLI 处理该文件。如果该设置的值为 `base64`（未明确设置时的默认值），则 AWS CLI 期望文件包含 `base64` 编码的文本。如果该设置的值为 `raw-in-base64-out`，则 AWS CLI 期望文件包含原始二进制内容。

## cli\_history

默认情况下禁用。此设置启用 AWS CLI 的命令历史记录。启用此设置后，会 AWS CLI 记录 `aws` 命令的历史记录。

```
cli_history = enabled
```

您可以使用 `aws history list` 命令列出您的历史记录，然后使用 `aws history show` 命令中生成的 `command_ids` 获取详细信息。有关更多信息，请参阅《AWS CLI 参考指南》中的 [aws history](#)。

## cli\_pager

指定用于处理输出的分页程序。默认情况下，AWS CLI 版本 2 通过操作系统的默认寻呼机程序返回所有输出。

可以被 `AWS_PAGER` 环境变量覆盖。

```
cli_pager=less
```

要完全禁用外部分页程序，请将该变量设置为空字符串，如以下示例所示。

```
cli_pager=
```

## cli\_timestamp\_format

指定输出中包含的时间戳值的格式。可以指定以下任一值：

- `iso8601` — 版本 2 的 AWS CLI 默认值。如果指定，则会根据 [ISO 8601](#) AWS CLI 重新格式化所有时间戳。

ISO 8601 格式的时间戳与以下示例类似。第一个示例通过在时间之后加入 `Z`，显示以 [协调世界时 \(UTC\)](#) 表示的时间。日期和时间由 `T` 分隔。

```
2019-10-31T22:21:41Z
```

要指定不同的时区，不是使用 Z，而是指定 + 或 - 以及所需时区在 UTC 之前或之后的小时数作为两位数值。以下示例显示的时间与上一个示例相同，但调整为太平洋标准时间（比 UTC 晚 8 小时）：

```
2019-10-31T14:21:41-08
```

- `wire` — AWS CLI 版本 1 的默认值。如果指定，则 AWS CLI 会显示与 HTTP 查询响应中收到的所有时间戳值完全相同。

该条目没有等效的环境变量或命令行选项。

```
cli_timestamp_format = iso8601
```

## credential\_process

指定一个外部命令，AWS CLI 运行该命令以生成或检索用于此命令的身份验证凭据。命令必须以特定格式返回凭证。有关如何使用该设置的更多信息，请参阅[使用外部进程获取凭证](#)。

该条目没有等效的环境变量或命令行选项。

```
credential_process = /opt/bin/awscreds-retriever --username susan
```

## credential\_source

在 Amazon EC2 实例或容器中使用，指定 AWS CLI 在何处可以找到要用于代入通过 `role_arn` 参数指定的角色的凭证。不能在同一配置文件中同时指定 `source_profile` 和 `credential_source`。

此参数具有三个值：

- `环境`—指定要从环境变量中检索源凭证。AWS CLI
- `Ec@@@ 2 InstanceMetadata` — 指定使用附加到 [EC2 实例配置文件](#) 的 IAM 角色来获取源证书。AWS CLI
- `EcsContainer`— 指定使用附加 AWS CLI 到 ECS 容器的 IAM 角色作为源证书。

```
credential_source = Ec2InstanceMetadata
```

## duration\_seconds

指定角色会话的最大持续时间（以秒为单位）。该值的范围在 900 秒（15 分钟）到角色的最大会话持续时间设置之间。此参数为可选参数，默认情况下，该值设置为 3600 秒。

## endpoint\_url

指定用于所有服务请求的端点。如果在 config 文件的 [services](#) 节中使用此设置，则该端点仅用于指定的服务。

以下示例使用全局端点 `http://localhost:1234` 和用于 Amazon S3 的特定于服务的端点 `http://localhost:4567`。

```
[profile dev]
endpoint_url = http://localhost:1234
services = s3-specific

[services s3-specific]
s3 =
    endpoint_url = http://localhost:4567
```

端点配置设置位于多个位置，例如系统或用户环境变量、本地 AWS 配置文件或在命令行中明确声明为参数。AWS CLI 端点配置设置的优先顺序如下：

1. [--endpoint-url](#) 命令行选项。
2. 如果启用，则 [AWS\\_IGNORE\\_CONFIGURED\\_ENDPOINT\\_URLS](#) 全局端点环境变量或配置文件设置 [ignore\\_configure\\_endpoint\\_urls](#) 将忽略自定义端点。
3. 由特定于服务的环境变量 [AWS\\_ENDPOINT\\_URL\\_<SERVICE>](#) 提供的值，例如 `AWS_ENDPOINT_URL_DYNAMODB`。
4. [AWS\\_USE\\_DUALSTACK\\_ENDPOINT](#)、[AWS\\_USE\\_FIPS\\_ENDPOINT](#) 和 [AWS\\_ENDPOINT\\_URL](#) 环境变量提供的值。
5. 共享 config 文件 `services` 节中的 [endpoint\\_url](#) 设置提供的特定于服务的端点值。
6. 共享 config 文件的 `profile` 中的 [endpoint\\_url](#) 设置提供的值。
7. [use\\_dualstack\\_endpoint](#)、[use\\_fips\\_endpoint](#) 和 [endpoint\\_url](#) 设置。
8. 最后使用相应 AWS 服务 端点 URL 的任何默认端点 URL。有关每个区域可用的标准服务端点的列表，请参阅《Amazon Web Services 一般参考》中的 [AWS 区域和端点](#)。

## ignore\_configure\_endpoint\_urls

如果启用，则 AWS CLI 会忽略 config 文件中指定的所有自定义终端节点配置。有效值为 **true** 和 **false**。

```
ignore_configure_endpoint_urls = true
```

端点配置设置位于多个位置，例如系统或用户环境变量、本地 AWS 配置文件或在命令行中明确声明为参数。AWS CLI 端点配置设置的优先顺序如下：

1. [--endpoint-url](#) 命令行选项。
2. 如果启用，则 [AWS\\_IGNORE\\_CONFIGURED\\_ENDPOINT\\_URLS](#) 全局端点环境变量或配置文件设置 [ignore\\_configure\\_endpoint\\_urls](#) 将忽略自定义端点。
3. 由特定于服务的环境变量 [AWS\\_ENDPOINT\\_URL\\_<SERVICE>](#) 提供的值，例如 [AWS\\_ENDPOINT\\_URL\\_DYNAMODB](#)。
4. [AWS\\_USE\\_DUALSTACK\\_ENDPOINT](#)、[AWS\\_USE\\_FIPS\\_ENDPOINT](#) 和 [AWS\\_ENDPOINT\\_URL](#) 环境变量提供的值。
5. 共享 config 文件 services 节中的 [endpoint\\_url](#) 设置提供的特定于服务的端点值。
6. 共享 config 文件的 profile 中的 [endpoint\\_url](#) 设置提供的值。
7. [use\\_dualstack\\_endpoint](#)、[use\\_fips\\_endpoint](#) 和 [endpoint\\_url](#) 设置。
8. 最后使用相应 AWS 服务 端点 URL 的任何默认端点 URL。有关每个区域可用的标准服务端点的列表，请参阅《Amazon Web Services 一般参考》中的 [AWS 区域和端点](#)。

## [external\\_id](#)

指定第三方用于在其客户账户中代入角色的唯一标识符。这将映射到 AssumeRole 操作中的 ExternalId 参数。仅当角色的信任策略为 ExternalId 指定值时，才需要此参数。有关更多信息，请参阅 IAM 用户指南中的 [如何在向第三方授予 AWS 资源访问权限时使用外部 ID](#)。

## [max\\_attempts](#)

指定重试处理程序使用的最大 AWS CLI 重试次数值，其中初始调用计入您 max\_attempts 提供的值。

您可以使用 AWS\_MAX\_ATTEMPTS 环境变量覆盖此值。

```
max_attempts = 3
```

## [mfa\\_serial](#)

代入角色时要使用的 MFA 设备的标识号。仅当代入角色的信任策略包含需要 MFA 身份验证的条件，此值才是必需的。该值可以是硬件设备（例如 GAHT12345678）的序列号，也可以是虚拟 MFA 设备（例如 arn:aws:iam::123456789012:mfa/*user*）的 Amazon Resource Name (ARN)。

## **output**

指定使用该配置文件请求的命令的默认输出格式。您可以指定以下任意值：

- **json** – 输出采用 [JSON](#) 字符串的格式。
- **yaml** – 输出采用 [YAML](#) 字符串的格式。
- **yaml-stream** – 输出被流式处理并采用 [YAML](#) 字符串的格式。串流支持更快地处理大型数据类型。
- **text** – 输出采用多个制表符分隔字符串值行的格式。这对于将输出传递到文本处理器 ( 如 `grep`、`sed` 或 `awk` ) 很有用。
- **table** – 输出采用表格形式，使用字符 `+|-` 以形成单元格边框。它通常以“人性化”格式呈现信息，这种格式比其他格式更容易阅读，但从编程方面来讲不是那么有用。

可以被 `AWS_DEFAULT_OUTPUT` 环境变量或 `--output` 命令行选项覆盖。

```
output = table
```

## parameter\_validation

指定 AWS CLI 客户端在将参数发送到 AWS 服务端点之前是否尝试验证参数。

- `true` – 这是默认值。如果指定，则 AWS CLI 会对命令行参数执行本地验证。
- `false` — 如果指定，则 AWS CLI 不会在将命令行参数发送到 AWS 服务端点之前对其进行验证。

该条目没有等效的环境变量或命令行选项。

```
parameter_validation = false
```

## region

对于使用此配置文件请求的命令，指定要向其发送请求。AWS 区域

- 您可以指定可用于所选服务的任何区域代码，有关区域代码的列表，请参阅《Amazon Web Services 一般参考》中的 [AWS 区域和端点](#)。
- `aws_global` 允许您为除区域终端节点之外还支持全局终端节点的服务指定全局终端节点，例如 AWS Security Token Service (AWS STS) 和亚马逊简单存储服务 (Amazon S3) Simple Service Amazon S3A。

您可以使用 `AWS_REGION` 环境变量、`AWS_DEFAULT_REGION` 环境变量或 `--region` 命令行选项覆盖此值。

```
region = us-west-2
```

## retry\_mode

指定 AWS CLI 使用哪种重试模式。有三种重试模式可用：旧模式（默认模式）、标准模式和自适应模式。有关重试的更多信息，请参阅[AWS CLI 重试次数](#)。

您可以使用 `AWS_RETRY_MODE` 环境变量覆盖此值。

```
retry_mode = standard
```

## role\_arn

指定要用于运行命令的 IAM 角色的 Amazon 资源名称 (ARN)。AWS CLI 此外，还必须指定以下参数之一以标识有权代入此角色的凭证：

- `source_profile`
- `credential_source`

```
role_arn = arn:aws:iam::123456789012:role/role-name
```

环境变量 [AWS\\_ROLE\\_ARN](#) 将覆盖此设置。

有关使用 Web 身份的更多信息，请参阅 [the section called “通过 Web 身份代入角色”](#)。

## role\_session\_name

指定要附加到角色会话的名称。此值在 `RoleSessionName` 调用 AWS CLI 操作时将提供给 `AssumeRole` 参数，并成为代入角色用户 ARN 的一部分：

`arn:aws:sts::123456789012:assumed-role/role_name/role_session_name`。此参数为可选参数。如果未提供此值，则将自动生成会话名称。此名称显示在与此会话关联的条目的 AWS CloudTrail 日志中。

```
role_session_name = maria_garcia_role
```

环境变量 [AWS\\_ROLE\\_SESSION\\_NAME](#) 将覆盖此设置。

有关使用 Web 身份的更多信息，请参阅 [the section called “通过 Web 身份代入角色”](#)。

## services

指定要用于您的配置文件的配置服务。

```
[profile dev-s3-specific-and-global]
```

```
endpoint_url = http://localhost:1234
services = s3-specific

[services s3-specific]
s3 =
    endpoint_url = http://localhost:4567
```

有关更多信息，请参阅 [the section called “services”](#) 中的 services 节。

环境变量 [AWS\\_ROLE\\_SESSION\\_NAME](#) 将覆盖此设置。

有关使用 Web 身份的更多信息，请参阅 [the section called “通过 Web 身份代入角色”](#)。

## [source\\_profile](#)

指定包含长期凭证的命名配置文件，AWS CLI 可使用这些凭证代入通过 `role_arn` 参数指定的角色。不能在同一配置文件中同时指定 `source_profile` 和 `credential_source`。

```
source_profile = production-profile
```

## [sso\\_account\\_id](#)

指定包含您要向关联的 IAM 身份中心用户授予权限的 IAM 角色的 AWS 账户 ID。

此设置没有环境变量或命令行选项。

```
sso_account_id = 123456789012
```

## [sso\\_region](#)

指定包含 AWS 访问门户主机的 AWS 区域。它与默认的 CLI `region` 参数是分开的，并且可以是不同的区域。

此设置没有环境变量或命令行选项。

```
sso_region = us-west-2
```

## [sso\\_registration\\_scopes](#)

要为 `sso-session` 授权的范围的逗号分隔列表。范围授权对 IAM Identity Center 持有者令牌授权终端节点的访问。有效范围是字符串，例如 `sso:account:access`。此设置不适用于遗留的不可刷新配置。

```
sso_registration_scopes = sso:account:access
```

### sso\_role\_name

指定使用此配置文件时定义用户权限的 IAM 角色的友好名称。

此设置没有环境变量或命令行选项。

```
sso_role_name = ReadAccess
```

### sso\_start\_url

指定指向组织 AWS 访问门户的 URL。AWS CLI 使用此 URL 与 IAM 身份中心服务建立会话以对其用户进行身份验证。要查找您的 AWS 访问门户 URL，请使用以下方法之一：

- 打开您的邀请电子邮件，将列出 AWS 访问门户 URL。
- 打开 AWS IAM Identity Center 控制台，[网址为 https://console.aws.amazon.com/singlesignon/](https://console.aws.amazon.com/singlesignon/)。AWS 访问门户 URL 列在您的设置中。

此设置没有环境变量或命令行选项。

```
sso_start_url = https://my-sso-portal.awsapps.com/start
```

### use\_dualstack\_endpoint

允许使用双堆栈端点发送 AWS 请求。要详细了解支持 IPv4 和 IPv6 流量的双堆栈端点，请参阅 Amazon Simple Storage Service 用户指南中的[使用 Amazon S3 双堆栈端点](#)。双堆栈端点适用于某些区域。如果服务不存在双栈端点或 AWS 区域，则请求将失败。默认情况下，将禁用该功能。

该设置与 `use_accelerate_endpoint` 设置互斥。

端点配置设置位于多个位置，例如系统或用户环境变量、本地 AWS 配置文件或在命令行中明确声明为参数。AWS CLI 端点配置设置的优先顺序如下：

1. `--endpoint-url` 命令行选项。
2. 如果启用，则 `AWS_IGNORE_CONFIGURED_ENDPOINT_URLS` 全局端点环境变量或配置文件设置 `ignore_configure_endpoint_urls` 将忽略自定义端点。



3. 由特定于服务的环境变量 [AWS\\_ENDPOINT\\_URL\\_<SERVICE>](#) 提供的值，例如 `AWS_ENDPOINT_URL_DYNAMODB`。
4. [AWS\\_USE\\_DUALSTACK\\_ENDPOINT](#)、[AWS\\_USE\\_FIPS\\_ENDPOINT](#) 和 [AWS\\_ENDPOINT\\_URL](#) 环境变量提供的值。
5. 共享 config 文件 `services` 节中的 [endpoint\\_url](#) 设置提供的特定于服务的端点值。
6. 共享 config 文件的 `profile` 中的 [endpoint\\_url](#) 设置提供的值。
7. [use\\_dualstack\\_endpoint](#)、[use\\_fips\\_endpoint](#) 和 [endpoint\\_url](#) 设置。
8. 最后使用相应 AWS 服务 端点 URL 的任何默认端点 URL。有关每个区域可用的标准服务端点的列表，请参阅《Amazon Web Services 一般参考》中的 [AWS 区域和端点](#)。

### *use\_fips\_endpoint*

有些 AWS 服务提供的端点支持[联邦信息处理标准 \(FIPS\) 140-2](#)。AWS 区域当 AWS 服务支持 FIPS 时，此设置指定 AWS CLI 应使用哪个 FIPS 端点。与标准 AWS 端点不同，FIPS 端点使用符合 FIPS 140-2 的 TLS 软件库。与美国政府有业务来往的企业可能需要使用这些端点。

如果启用了此设置，但您的服务中不存在 FIPS 端点 AWS 区域，则该 AWS 命令可能会失败。在这种情况下，请使用 [--endpoint-url](#) 选项手动指定要在命令中使用的端点，或者使用[特定于服务的端点](#)。

有关通过指定 FIPS 端点的更多信息 AWS 区域，请参阅按服务划分的 [FIPS 终端节点](#)。

端点配置设置位于多个位置，例如系统或用户环境变量、本地 AWS 配置文件或在命令行中明确声明为参数。AWS CLI 端点配置设置的优先顺序如下：

1. [--endpoint-url](#) 命令行选项。
2. 如果启用，则 [AWS\\_IGNORE\\_CONFIGURED\\_ENDPOINT\\_URLS](#) 全局端点环境变量或配置文件设置 [ignore\\_configure\\_endpoint\\_urls](#) 将忽略自定义端点。
3. 由特定于服务的环境变量 [AWS\\_ENDPOINT\\_URL\\_<SERVICE>](#) 提供的值，例如 `AWS_ENDPOINT_URL_DYNAMODB`。
4. [AWS\\_USE\\_DUALSTACK\\_ENDPOINT](#)、[AWS\\_USE\\_FIPS\\_ENDPOINT](#) 和 [AWS\\_ENDPOINT\\_URL](#) 环境变量提供的值。
5. 共享 config 文件 `services` 节中的 [endpoint\\_url](#) 设置提供的特定于服务的端点值。
6. 共享 config 文件的 `profile` 中的 [endpoint\\_url](#) 设置提供的值。
7. [use\\_dualstack\\_endpoint](#)、[use\\_fips\\_endpoint](#) 和 [endpoint\\_url](#) 设置。
8. 最后使用相应 AWS 服务 端点 URL 的任何默认端点 URL。有关每个区域可用的标准服务端点的列表，请参阅《Amazon Web Services 一般参考》中的 [AWS 区域和端点](#)。

## [web\\_identity\\_token\\_file](#)

指定一个文件的路径，该文件包含由身份提供商提供的 OAuth 2.0 访问令牌或 OpenID Connect ID 令牌。AWS CLI 加载此文件的内容，并将其作为 `WebIdentityToken` 参数传递给 `AssumeRoleWithWebIdentity` 操作。

环境变量 [AWS\\_WEB\\_IDENTITY\\_TOKEN\\_FILE](#) 将覆盖此设置。

有关使用 Web 身份的更多信息，请参阅 [the section called “通过 Web 身份代入角色”](#)。

## *tcp\_keepalive*

指定 AWS CLI 客户端是否使用 TCP 保持活动状态数据包。

该条目没有等效的环境变量或命令行选项。

```
tcp_keepalive = false
```

## S3 自定义命令设置

Amazon S3 支持多种设置，用于配置如何 AWS CLI 执行 Amazon S3 的操作。一些设置适用于 `s3api` 和 `s3` 命名空间中的所有 S3 命令。其他命令专门用于 S3 的“自定义”命令，这些命令抽象了常见的操作，其作用不仅仅是 one-to-one 映射到 API 操作。`aws s3` 传输命令 `cp`、`sync`、`mv` 和 `rm` 具有可用于控制 S3 传输的其他设置。

可以通过在 `config` 文件中指定 `s3` 嵌套设置来配置所有这些选项。每个设置在其自己的行上缩进。

### Note

这些设置完全是可选的。即使不配置这些设置中的任何一个，您也应该能够成功使用 `aws s3` 传输命令。提供这些设置是为了让您能够调整性能或匹配运行这些 `aws s3` 命令的特定环境。

这些设置都在 `config` 文件中的顶层 `s3` 键下设置，如以下 `development` 配置文件示例所示。

```
[profile development]
s3 =
  max_concurrent_requests = 20
  max_queue_size = 10000
  multipart_threshold = 64MB
  multipart_chunksize = 16MB
```

```
max_bandwidth = 50MB/s
use_accelerate_endpoint = true
addressing_style = path
```

以下设置适用于 s3 或 s3api 命名空间中的任何 S3 命令。

## addressing\_style

指定要使用的寻址样式。这将控制存储桶名称是位于主机名还是 URL 中。有效值包括 path、virtual 和 auto。默认值为 auto。

构造 Amazon S3 终端节点的样式有两种。第一种称为 virtual，它将存储桶名称包含为主机名的一部分。例如：`https://bucketname.s3.amazonaws.com`。另一种为 path 样式，您将存储桶名称视为 URI 中的路径，例如 `https://s3.amazonaws.com/bucketname`。CLI 中的默认值是使用 auto，它尝试尽可能使用 virtual 样式，但在需要时回退到 path 样式。例如，如果您的存储桶名称与 DNS 不兼容，则存储桶名称不能是主机名的一部分，而必须位于路径中。使用 auto 时，CLI 将检测这种情况并自动切换到 path 样式。如果您将寻址方式设置为 path，则必须确保您在中配置的 AWS 区域与存储桶的区域 AWS CLI 相匹配。

## payload\_signing\_enabled

指定是否对 sigv4 负载进行 SHA256 签名。默认情况下，使用 HTTPS 时，将对流式上传（UploadPart 和 PutObject）禁用该设置。默认情况下，对于流式上传（UploadPart 和 PutObject），此设置为 false，但仅限存在 ContentMD5（默认生成）并且终端节点使用 HTTPS 时。

如果设置为 true，则 S3 请求接收 SHA256 校验和形式的额外内容验证（替您计算并包含在请求签名中）。如果设置为 false，则不计算校验和。禁用该设置可减少校验和计算产生的性能开销。

## use\_accelerate\_endpoint

为所有 s3 和 s3api 命令使用 Amazon S3 加速终端节点。默认值为 False。该设置与 use\_dualstack\_endpoint 设置互斥。

如果设置为 true，则 AWS CLI 会将所有 Amazon S3 请求定向到的 S3 Accelerate 终端节点 `s3-accelerate.amazonaws.com`。要使用该终端节点，您必须让您的存储桶使用 S3 Accelerate。使用存储桶寻址的虚拟样式发送所有请求：`my-bucket.s3-accelerate.amazonaws.com`。不会将任何 ListBuckets、CreateBucket 和 DeleteBucket 请求发送到 S3 加速终端节点，因为该终端节点不支持这些操作。如果将任何 `--endpoint-url` 或 `https://s3-accelerate.amazonaws.com` 命令的 `http://s3-accelerate.amazonaws.com` 参数设置为 s3 或 s3api，也可以设置该行为。

以下设置仅适用于 s3 命名空间命令集中的命令。

## max\_bandwidth

指定向 Amazon S3 上传数据和从其下载数据可使用的最大带宽。默认为无限制。

这限制了 S3 命令可用于向 Amazon S3 传输数据和从 Amazon S3 传输数据的最大带宽。该值仅适用于上传和下载；它不适用于复制或删除。值以每秒字节数表示。该值可以指定为：

- 一个整数。例如，1048576 将最大带宽使用率设置为每秒 1 兆字节。
- 一个整数，后跟速率后缀。可以使用以下格式指定速率后缀：KB/s、MB/s 或 GB/s。例如，300KB/s 和 10MB/s。

通常，我们建议您先尝试通过降低 `max_concurrent_requests` 来降低带宽使用率。如果这样做没有充分地将带宽使用率限制到所需速率，您可以使用 `max_bandwidth` 设置进一步限制带宽使用率。这是因为 `max_concurrent_requests` 控制当前运行的线程数。如果您先降低 `max_bandwidth` 但保持较高的 `max_concurrent_requests` 设置，则可能导致线程不得不进行不必要的等待。这可能造成过多的资源消耗和连接超时。

## max\_concurrent\_requests

指定最大并发请求数。默认值是 10。

`aws s3` 传输命令是多线程的。在任意给定时间，都可以运行多个 Amazon S3 请求。例如，当您使用命令将文件上传 `aws s3 cp localdir s3://bucket/ --recursive` 到 S3 存储桶时，AWS CLI 可以并行上传文件 `localdir/file1`、`localdir/file2`、和 `localdir/file3`。设置 `max_concurrent_requests` 指定可同时运行的最大传输操作数。

您可能由于以下原因而需要更改该值：

- 减小该值 – 在某些环境中，默认的 10 个并发请求可能会占用过多的系统资源。这可能导致连接超时或系统响应速度变慢。减小该值可减少 S3 传输命令消耗的资源。但不利后果是 S3 传输可能需要更长时间才能完成。如果使用了限制带宽的工具，则可能需要减小该值。
- 增大该值 – 在某些情况下，您可能希望 Amazon S3 传输根据需要使用尽可能多的网络带宽，以尽可能快地完成任务。在这种情况下，默认的并发请求数可能不足以使用所有可用的网络带宽。增大该值可缩短完成 Amazon S3 传输所需的时间。

## max\_queue\_size

指定任务队列中的最大任务数。默认值是 1000。

AWS CLI 内部使用一种模型，它将 Amazon S3 任务排队，然后由数量受限的使用者执行 `max_concurrent_requests`。任务通常映射到单个 Amazon S3 操作。例如，任务可以是

PutObjectTask、GetObjectTask 或 UploadPartTask。任务添加到队列的速度可能比使用者完成任务的速度快得多。为避免无限制增长，任务队列大小设置了特定大小的上限。该设置用于更改该最大数量的值。

您通常不需要更改该设置。此设置还与他们知道需要运行的任务数量相对应。AWS CLI 这意味着默认情况下，他们 AWS CLI 只能看到前面的 1000 个任务。假设排队速度快于任务完成率，则增加此值意味着 AWS CLI 可以更快地知道所需的任务总数。但不利后果是更大的 `max_queue_size` 需要更多的内存。

## `multipart_chunksize`

指定用于单个文件的分段传输的区块大小。AWS CLI 默认值为 8 MB，最少为 5 MB。

当文件传输超出 `multipart_threshold` 时，AWS CLI 将文件分成该大小的块。可以使用与 `multipart_threshold` 相同的语法指定该值，即整数形式的字节数，或使用大小和后缀。

## `multipart_threshold`

指定用于单个文件分段传输的大小阈值。AWS CLI 默认值为 8 MB。

上传、下载或复制文件时，如果文件超出该大小，Amazon S3 命令将切换到分段操作。您可以通过以下两种方式之一指定该值：

- 文件大小（以字节为单位）。例如，1048576。
- 文件大小及大小后缀。您可以使用 KB、MB、GB 或 TB。例如，10MB、1GB。

### Note

S3 可能会对可用于分段操作的有效值施加约束。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的 [S3 分段上传文档](#)。

## 用于配置的环境变量 AWS CLI

环境变量提供了另一种指定配置选项和凭证的方法；若要编写脚本或将一个命名配置文件临时设置为默认配置文件，环境变量会很有用。

### 选项的优先顺序

- 如果您使用本主题中描述的某个环境变量指定选项，则它将在配置文件中覆盖从配置文件加载的任何值。

- 如果您在 AWS CLI 命令行中使用参数来指定选项，则该选项将覆盖配置文件中相应环境变量或配置文件中的任何值。

有关优先级以及如何 AWS CLI 决定使用哪些凭证的更多信息，请参阅[配置 AWS CLI](#)。

## 主题

- [如何设置环境变量](#)
- [AWS CLI 支持的环境变量](#)

## 如何设置环境变量

下面的示例介绍您如何可以为默认用户配置环境变量。

### Linux or macOS

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
$ export AWS_DEFAULT_REGION=us-west-2
```

设置环境变量会更改使用的值，直到 Shell 会话结束或直到您将该变量设置为其他值。通过在 shell 的启动脚本中设置变量，可使变量在未来的会话中继续有效。

### Windows Command Prompt

为所有会话设置

```
C:\> setx AWS_ACCESS_KEY_ID AKIAIOSFODNN7EXAMPLE
C:\> setx AWS_SECRET_ACCESS_KEY wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
C:\> setx AWS_DEFAULT_REGION us-west-2
```

使用 [setx](#) 设置环境变量会更改当前命令提示符会话和运行该命令后创建的所有命令提示符会话中使用的值。它不影响在运行该命令时已经运行的其他命令 shell。您可能需要重启终端来加载设置。

仅为当前会话设置

使用 [set](#) 设置环境变量会更改使用的值，直到当前命令提示符会话结束，或者直到您将该变量设置为其他值。

```
C:\> set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
C:\> set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
C:\> set AWS_DEFAULT_REGION=us-west-2
```

## PowerShell

```
PS C:\> $Env:AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
PS C:\> $Env:AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY"
PS C:\> $Env:AWS_DEFAULT_REGION="us-west-2"
```

如果您在 PowerShell 提示符处设置环境变量（如前面的示例所示），则它只会在当前会话的持续时间内保存该值。要使环境变量设置在所有会话 PowerShell 和命令提示符会话中保持不变，请使用控制面板中的系统应用程序将其存储。或者，您可以通过将变量添加到您的 PowerShell 个人资料中来为所有未来 PowerShell 会话设置该变量。有关存储环境变量或跨会话保存环境变量的更多信息，请参阅[PowerShell 文档](#)。

## AWS CLI 支持的环境变量

AWS CLI 支持以下环境变量。

### AWS\_ACCESS\_KEY\_ID

指定与 IAM 账户关联的 AWS 访问密钥。

如果已定义此环境变量，它将覆盖配置文件设置 `aws_access_key_id` 的值。您不能使用命令行选项来指定访问密钥 ID。

### AWS\_CA\_BUNDLE

指定要用于 HTTPS 证书验证的证书捆绑包的路径。

如果已定义此环境变量，它将覆盖配置文件设置 `ca_bundle` 的值。您可以使用 `--ca-bundle` 命令行参数覆盖此环境变量。

### AWS\_CLI\_AUTO\_PROMPT

启用 AWS CLI 版本 2 的自动提示。可以使用两种设置：

- **on** 每次尝试运行 `aws` 命令时都会使用完整的自动提示模式。这包括在完整或不完整的命令之后按 ENTER 键。



- **on-partial** 使用部分自动提示模式。如果命令不完整或由于客户端验证错误而无法运行，则使用自动提示。如果您有预先存在的脚本、运行手册，或者只想自动提示您输入不熟悉的命令，而不是在每个命令上都得到提示，则此模式非常有用。

如果已定义此环境变量，它将覆盖 `cli_auto_prompt` 配置文件设置的值。您可以使用 `--cli-auto-prompt` 和 `--no-cli-auto-prompt` 命令行参数覆盖此环境变量。

有关 AWS CLI 版本 2 自动提示功能的信息，请参阅[让 AWS CLI 提示您输入命令](#)。

## AWS\_CLI\_FILE\_ENCODING

指定用于文本文件的编码。默认情况下，编码与您的区域设置匹配。要设置与区域设置不同的编码，请使用 `aws_cli_file_encoding` 环境变量。例如，如果您使用 Windows 以及默认编码 CP1252，则 `aws_cli_file_encoding=UTF-8` 设置会将 CLI 设置为使用 UTF-8 打开文本文件。

## AWS\_CLI\_S3\_MV\_VALIDATE\_SAME\_S3\_PATHS

如果使用自定义 `s3 mv` 命令时源存储桶和目标存储桶相同，则可以将源文件或对象移到其自身上，这可能会导致源文件或对象意外删除。`AWS_CLI_S3_MV_VALIDATE_SAME_S3_PATHS` 环境变量和 `--validate-same-s3-paths` 选项指定是验证接入点 ARN 还是验证您的 Amazon S3 源或目标 URI 中的接入点别名。

### Note

的路径验证 `s3 mv` 需要额外的 API 调用。

## AWS\_CONFIG\_FILE

指定用于存储配置文件的 AWS CLI 文件的位置。默认路径为 `~/.aws/config`。

您不能在命名配置文件设置中或使用命令行参数来指定此值。

## AWS\_DATA\_PATH

加载 AWS CLI 数据 `~/.aws/models` 时需要在内置搜索路径之外检查的其他目录列表。设置此环境变量将指示在回滚到内置搜索路径前要先检查的其他目录。应使用 `os.pathsep` 字符（在 Linux 上为 `:`，在 Windows 上为 `;`）隔开多个条目。

## AWS\_DEFAULT\_OUTPUT

指定要使用的[输出格式](#)。



如果已定义此环境变量，它将覆盖配置文件设置 `output` 的值。您可以使用 `--output` 命令行参数覆盖此环境变量。

## AWS\_DEFAULT\_REGION

标 `Default region name` 识默认情况下您要向其服务器发送请求的 AWS 区域。通常是离您最近的区域，但可以是任意区域。例如，您可以键入 `us-west-2` 以使用美国西部（俄勒冈）。除非在命令中另行指定，否则这是所有后续请求将发送到的区域。

### Note

使用时，AWS 必须明确或通过设置默认区域来指定区域。AWS CLI 有关可用区域的列表，请参阅 [区域和终端节点](#)。使用的区域标识符与您在 AWS Management Console 网址和服务端点中看到的名称相同。AWS CLI

如果已定义此环境变量，它将覆盖配置文件设置 `region` 的值。您可以使用 `--region` 命令行参数和与 AWS SDK 兼容的环境变量来覆盖此 `AWS_REGION` 环境变量。

## AWS\_EC2\_METADATA\_DISABLED

禁用 Amazon EC2 实例元数据服务 (IMDS)。

如果设置为 `true`，则不会从 IMDS 请求用户凭证或配置（如区域）。

## AWS\_ENDPOINT\_URL

指定用于所有服务请求的端点。

端点配置设置位于多个位置，例如系统或用户环境变量、本地 AWS 配置文件或在命令行中明确声明为参数。AWS CLI 端点配置设置的优先顺序如下：

1. [--endpoint-url](#) 命令行选项。
2. 如果启用，则 [AWS\\_IGNORE\\_CONFIGURED\\_ENDPOINT\\_URLS](#) 全局端点环境变量或配置文件设置 [ignore\\_configure\\_endpoint\\_urls](#) 将忽略自定义端点。
3. 由特定于服务的环境变量 [AWS\\_ENDPOINT\\_URL\\_<SERVICE>](#) 提供的值，例如 `AWS_ENDPOINT_URL_DYNAMODB`。
4. [AWS\\_USE\\_DUALSTACK\\_ENDPOINT](#)、[AWS\\_USE\\_FIPS\\_ENDPOINT](#) 和 [AWS\\_ENDPOINT\\_URL](#) 环境变量提供的值。
5. 共享 config 文件 `services` 节中的 [endpoint\\_url](#) 设置提供的特定于服务的端点值。

6. 共享 config 文件的 profile 中的 [endpoint\\_url](#) 设置提供的值。
7. [use\\_dualstack\\_endpoint](#)、[use\\_fips\\_endpoint](#) 和 [endpoint\\_url](#) 设置。
8. 最后使用相应 AWS 服务 端点的所有默认端点 URL。有关每个区域可用的标准服务端点的列表，请参阅《Amazon Web Services 一般参考》中的 [AWS 区域和端点](#)。

### AWS\_ENDPOINT\_URL\_<SERVICE>

指定用于特定服务的自定义终端节点，其中替换<SERVICE>为标 AWS 服务 标识符。例如，Amazon DynamoDB 有 `serviceId` 个 [DynamoDB](#)。对于此服务，端点 URL 环境变量为 `AWS_ENDPOINT_URL_DYNAMODB`。

有关特定于服务的所有环境变量的列表，请参阅[特定于服务的标识符列表](#)。

端点配置设置位于多个位置，例如系统或用户环境变量、本地 AWS 配置文件或在命令行中明确声明为参数。AWS CLI 端点配置设置的优先顺序如下：

1. [--endpoint-url](#) 命令行选项。
2. 如果启用，则 [AWS\\_IGNORE\\_CONFIGURED\\_ENDPOINT\\_URLS](#) 全局端点环境变量或配置文件设置 [ignore\\_configure\\_endpoint\\_urls](#) 将忽略自定义端点。
3. 由特定于服务的环境变量 [AWS\\_ENDPOINT\\_URL\\_<SERVICE>](#) 提供的值，例如 `AWS_ENDPOINT_URL_DYNAMODB`。
4. [AWS\\_USE\\_DUALSTACK\\_ENDPOINT](#)、[AWS\\_USE\\_FIPS\\_ENDPOINT](#) 和 [AWS\\_ENDPOINT\\_URL](#) 环境变量提供的值。
5. 共享 config 文件 `services` 节中的 [endpoint\\_url](#) 设置提供的特定于服务的端点值。
6. 共享 config 文件的 profile 中的 [endpoint\\_url](#) 设置提供的值。
7. [use\\_dualstack\\_endpoint](#)、[use\\_fips\\_endpoint](#) 和 [endpoint\\_url](#) 设置。
8. 最后使用相应 AWS 服务 端点的所有默认端点 URL。有关每个区域可用的标准服务端点的列表，请参阅《Amazon Web Services 一般参考》中的 [AWS 区域和端点](#)。

### AWS\_IGNORE\_CONFIGURED\_ENDPOINT\_URLS

如果启用，则会 AWS CLI 忽略所有自定义终端节点配置。有效值为 **true** 和 **false**。

端点配置设置位于多个位置，例如系统或用户环境变量、本地 AWS 配置文件或在命令行中明确声明为参数。AWS CLI 端点配置设置的优先顺序如下：

1. [--endpoint-url](#) 命令行选项。
2. 如果启用，则 [AWS\\_IGNORE\\_CONFIGURED\\_ENDPOINT\\_URLS](#) 全局端点环境变量或配置文件设置 [ignore\\_configure\\_endpoint\\_urls](#) 将忽略自定义端点。

3. 由特定于服务的环境变量 [AWS\\_ENDPOINT\\_URL\\_<SERVICE>](#) 提供的值，例如 `AWS_ENDPOINT_URL_DYNAMODB`。
4. [AWS\\_USE\\_DUALSTACK\\_ENDPOINT](#)、[AWS\\_USE\\_FIPS\\_ENDPOINT](#) 和 [AWS\\_ENDPOINT\\_URL](#) 环境变量提供的值。
5. 共享 config 文件 `services` 节中的 [endpoint\\_url](#) 设置提供的特定于服务的端点值。
6. 共享 config 文件的 `profile` 中的 [endpoint\\_url](#) 设置提供的值。
7. [use\\_dualstack\\_endpoint](#)、[use\\_fips\\_endpoint](#) 和 [endpoint\\_url](#) 设置。
8. 最后使用相应 AWS 服务 端点的所有默认端点 URL。有关每个区域可用的标准服务端点的列表，请参阅《Amazon Web Services 一般参考》中的 [AWS 区域和端点](#)。

## [AWS\\_MAX\\_ATTEMPTS](#)

指定重试处理程序使用的最大 AWS CLI 重试次数值，其中初始调用计入您提供的值。有关重试的更多信息，请参阅[AWS CLI 重试次数](#)。

如果已定义此环境变量，它将覆盖配置文件设置 `max_attempts` 的值。

## [AWS\\_METADATA\\_SERVICE\\_NUM\\_ATTEMPTS](#)

尝试在配置了 IAM 角色的 Amazon EC2 实例上检索证书时，会在停止之前 AWS CLI 尝试从实例元数据服务中检索一次证书。如果您知道您的命令将在 Amazon EC2 实例上运行，则可增大此值，以使 AWS CLI 在放弃前重试多次。

## [AWS\\_METADATA\\_SERVICE\\_TIMEOUT](#)

与实例元数据服务的连接超时前等待的秒数。尝试在已配置 IAM 角色的 Amazon EC2 实例上检索凭证时，默认情况下，与实例元数据服务的连接将在 1 秒后超时。如果您知道您正在已配置 IAM 角色的 Amazon EC2 实例上运行，如有必要，可增大此值。

## [AWS\\_PAGER](#)

指定用于处理输出的分页程序。默认情况下，AWS CLI 版本 2 通过操作系统的默认寻呼机程序返回所有输出。

要完全禁用外部分页程序，请将该变量设置为空字符串。

如果已定义此环境变量，它将覆盖配置文件设置 `cli_pager` 的值。

## [AWS\\_PROFILE](#)

指定 AWS CLI 配置文件的名称以及要使用的凭据和选项。可以是存储在 `credentials` 或 `config` 文件中的配置文件的名称，也可以是值 `default`，后者使用默认配置文件。

如果您定义了此环境变量，它将在配置文件中覆盖使用名为 [default] 的配置文件的配置文件的行。您可以使用 `--profile` 命令行参数覆盖此环境变量。

## AWS\_REGION

与 AWS SDK 兼容的环境变量，用于指定要向其发送请求的 AWS 区域。

如果已定义此环境变量，它将覆盖环境变量 `AWS_DEFAULT_REGION` 和配置文件设置 `region` 中的值。您可以使用 `--region` 命令行参数覆盖此环境变量。

## AWS\_RETRY\_MODE

指定 AWS CLI 使用哪种重试模式。有三种重试模式可用：旧模式（默认模式）、标准模式和自适应模式。有关重试的更多信息，请参阅[AWS CLI 重试次数](#)。

如果已定义此环境变量，它将覆盖配置文件设置 `retry_mode` 的值。

## AWS\_ROLE\_ARN

指定具有网络身份提供商的 IAM 角色的 Amazon 资源名称 (ARN)，您要使用该提供商来运行命令。AWS CLI

结合使用 `AWS_WEB_IDENTITY_TOKEN_FILE` 和 `AWS_ROLE_SESSION_NAME` 环境变量。

如果已定义此环境变量，它将覆盖配置文件设置 `role_arn` 的值。不能将角色会话名称指定为命令行参数。

### Note

此环境变量仅适用于使用 Web 身份提供商的代入角色，而不适用于常规代入角色提供商配置。

有关使用 Web 身份的更多信息，请参阅 [the section called “通过 Web 身份代入角色”](#)。

## AWS\_ROLE\_SESSION\_NAME

指定要附加到角色会话的名称。该值在 AWS CLI 调用 `AssumeRole` 操作时提供给 `RoleSessionName` 参数，并成为代入角色用户 ARN: 的一部分。

`arn:aws:sts::123456789012:assumed-role/role_name/role_session_name` 此参数为可选参数。如果未提供此值，则将自动生成会话名称。此名称会出现在与此会话相关的条目的 AWS CloudTrail 日志中。

如果已定义此环境变量，它将覆盖配置文件设置 [role\\_session\\_name](#) 的值。

结合使用 `AWS_ROLE_ARN` 和 `AWS_WEB_IDENTITY_TOKEN_FILE` 环境变量。

有关使用 Web 身份的更多信息，请参阅 [the section called “通过 Web 身份代入角色”](#)。

#### Note

此环境变量仅适用于使用 Web 身份提供商的代入角色，而不适用于常规代入角色提供商配置。

## **AWS\_SECRET\_ACCESS\_KEY**

指定与访问密钥关联的私有密钥。这基本上是访问密钥的“密码”。

如果已定义此环境变量，它将覆盖配置文件设置 `aws_secret_access_key` 的值。您不能将秘密访问密钥 ID 指定为命令行选项。

## **AWS\_SESSION\_TOKEN**

指定在使用您直接从 AWS STS 操作中检索的临时安全凭证时需要的会话令牌值。有关更多信息，请参阅 AWS CLI 命令引用中的 [代入角色命令的输出部分](#)。

如果已定义此环境变量，它将覆盖配置文件设置 `aws_session_token` 的值。

## **AWS\_SHARED\_CREDENTIALS\_FILE**

指定用于存储访问密钥的文件的位置。AWS CLI 默认路径为 `~/.aws/credentials`。

您不能在命名配置文件设置中或使用命令行参数来指定此值。

## **AWS\_USE\_DUALSTACK\_ENDPOINT**

允许使用双堆栈端点发送 AWS 请求。要详细了解支持 IPv4 和 IPv6 流量的双堆栈端点，请参阅 Amazon Simple Storage Service 用户指南中的 [使用 Amazon S3 双堆栈端点](#)。双堆栈端点适用于某些区域。如果服务不存在双栈端点或 AWS 区域，则请求将失败。默认情况下，将禁用该功能。

端点配置设置位于多个位置，例如系统或用户环境变量、本地 AWS 配置文件或在命令行中明确声明为参数。AWS CLI 端点配置设置的优先顺序如下：

1. [--endpoint-url](#) 命令行选项。

2. 如果启用，则 [AWS\\_IGNORE\\_CONFIGURED\\_ENDPOINT\\_URLS](#) 全局端点环境变量或配置文件设置 [ignore\\_configure\\_endpoint\\_urls](#) 将忽略自定义端点。
3. 由特定于服务的环境变量 [AWS\\_ENDPOINT\\_URL\\_<SERVICE>](#) 提供的值，例如 `AWS_ENDPOINT_URL_DYNAMODB`。
4. [AWS\\_USE\\_DUALSTACK\\_ENDPOINT](#)、[AWS\\_USE\\_FIPS\\_ENDPOINT](#) 和 [AWS\\_ENDPOINT\\_URL](#) 环境变量提供的值。
5. 共享 config 文件 `services` 节中的 [endpoint\\_url](#) 设置提供的特定于服务的端点值。
6. 共享 config 文件的 `profile` 中的 [endpoint\\_url](#) 设置提供的值。
7. [use\\_dualstack\\_endpoint](#)、[use\\_fips\\_endpoint](#) 和 [endpoint\\_url](#) 设置。
8. 最后使用相应 AWS 服务 端点的所有默认端点 URL。有关每个区域可用的标准服务端点的列表，请参阅《Amazon Web Services 一般参考》中的 [AWS 区域和端点](#)。

## AWS\_USE\_FIPS\_ENDPOINT

有些 AWS 服务提供的端点支持[联邦信息处理标准 \(FIPS\) 140-2](#)。AWS 区域当 AWS 服务支持 FIPS 时，此设置指定 AWS CLI 应使用哪个 FIPS 端点。与标准 AWS 端点不同，FIPS 端点使用符合 FIPS 140-2 的 TLS 软件库。与美国政府有业务来往的企业可能需要使用这些端点。

如果启用了此设置，但您的服务中不存在 FIPS 端点 AWS 区域，则该 AWS 命令可能会失败。在这种情况下，请使用 [--endpoint-url](#) 选项手动指定要在命令中使用的端点，或者使用[特定于服务的端点](#)。

有关通过指定 FIPS 端点的更多信息 AWS 区域，请参阅按服务划分的 [FIPS 终端节点](#)。

端点配置设置位于多个位置，例如系统或用户环境变量、本地 AWS 配置文件或在命令行中明确声明为参数。AWS CLI 端点配置设置的优先顺序如下：

1. [--endpoint-url](#) 命令行选项。
2. 如果启用，则 [AWS\\_IGNORE\\_CONFIGURED\\_ENDPOINT\\_URLS](#) 全局端点环境变量或配置文件设置 [ignore\\_configure\\_endpoint\\_urls](#) 将忽略自定义端点。
3. 由特定于服务的环境变量 [AWS\\_ENDPOINT\\_URL\\_<SERVICE>](#) 提供的值，例如 `AWS_ENDPOINT_URL_DYNAMODB`。
4. [AWS\\_USE\\_DUALSTACK\\_ENDPOINT](#)、[AWS\\_USE\\_FIPS\\_ENDPOINT](#) 和 [AWS\\_ENDPOINT\\_URL](#) 环境变量提供的值。
5. 共享 config 文件 `services` 节中的 [endpoint\\_url](#) 设置提供的特定于服务的端点值。
6. 共享 config 文件的 `profile` 中的 [endpoint\\_url](#) 设置提供的值。

7. [use\\_dualstack\\_endpoint](#)、[use\\_fips\\_endpoint](#) 和 [endpoint\\_url](#) 设置。
8. 最后使用相应 AWS 服务 端点的所有默认端点 URL。有关每个区域可用的标准服务端点的列表，请参阅《Amazon Web Services 一般参考》中的 [AWS 区域和端点](#)。

## [AWS\\_WEB\\_IDENTITY\\_TOKEN\\_FILE](#)

指定一个文件的路径，该文件包含由身份提供商提供的 OAuth 2.0 访问令牌或 OpenID Connect ID 令牌。AWS CLI 加载此文件的内容，并将其作为 `WebIdentityToken` 参数传递给 `AssumeRoleWithWebIdentity` 操作。

结合使用 `AWS_ROLE_ARN` 和 `AWS_ROLE_SESSION_NAME` 环境变量。

如果已定义此环境变量，它将覆盖配置文件设置 `web_identity_token_file` 的值。

有关使用 Web 身份的更多信息，请参阅 [the section called “通过 Web 身份代入角色”](#)。

### Note

此环境变量仅适用于使用 Web 身份提供商的代入角色，而不适用于常规代入角色提供商配置。

## 命令行选项

在 AWS CLI 中，命令行选项是您用来覆盖该单个命令的原定设置配置设置、任何相应的配置文件设置或环境变量设置的全局参数。虽然您可以指定要使用的配置文件，但无法使用命令行选项直接指定凭证。

### 主题

- [如何使用命令行选项](#)
- [AWS CLI 支持的全局命令行选项](#)
- [命令行选项的常见用法](#)

## 如何使用命令行选项

大多数命令行选项都是简单的字符串，例如，以下示例中的配置文件名 `profile1`：

```
$ aws s3 ls --profile profile1
```



```
example-bucket-1
example-bucket-2
...
```

每个带参数的选项都需要一个空格或等号 (=) 将参数与选项名称分开。如果参数值为包含空格的字符串，则必须使用引号将参数引起来。有关参数类型和参数格式的详细信息，请参阅 [为 AWS CLI 指定数值](#)。

## AWS CLI 支持的全局命令行选项

在 AWS CLI 中，您可以使用以下命令行选项来覆盖该单个命令的原定设置配置设置、任何相应的配置文件设置或环境变量设置。

`--ca-bundle <string>`

指定验证 SSL 证书时要使用的证书颁发机构 (CA) 证书捆绑包。

如果已定义，此选项将覆盖配置文件设置 `ca_bundle` 和 `AWS_CA_BUNDLE` 环境变量的值。

`--cli-auto-prompt`

为单个命令启用自动提示模式。如以下示例所示，您可以随时指定它。

```
$ aws --cli-auto-prompt
$ aws dynamodb --cli-auto-prompt
$ aws dynamodb describe-table --cli-auto-prompt
```

此选项将覆盖 `aws_cli_auto_prompt` 环境变量和 `cli_auto_prompt` 配置文件设置。

有关 AWS CLI 版本 2 自动提示特征的信息，请参阅 [让 AWS CLI 提示您输入命令](#)。

`--cli-binary-format`

指定 AWS CLI 版本 2 如何解释二进制输入参数。它可能为下列值之一：

- `base64` – 这是默认值。作为二进制大型对象 (BLOB) 键入的输入参数接受 `base64` 编码字符串。要传递真实的二进制内容，请将内容放在文件中，并提供文件的路径和名称，将 `file://` 前缀作为参数值。要传递文件中包含的 `base64` 编码文本，请提供文件的路径和名称，并将 `file://` 前缀作为参数值。
- `raw-in-base64-out` – AWS CLI 版本 1 的默认值。如果设置的值为 `raw-in-base64-out`，则使用 `file://` 前缀引用的文件会作为文本进行读取，然后 AWS CLI 尝试将其编码为二进制。



这将覆盖 `cli_binary_format` 文件配置设置。

```
$ aws lambda invoke \  
  --cli-binary-format raw-in-base64-out \  
  --function-name my-function \  
  --invocation-type Event \  
  --payload '{ "name": "Bob" }' \  
  response.json
```

如果使用 `fileb://` 前缀表示法引用文件中的二进制值，AWS CLI 始终希望文件中包含原始二进制内容，并且不会尝试转换该值。

如果使用 `file://` 前缀表示法引用文件中的二进制值，AWS CLI 会根据当前 `cli_binary_format` 设置来处理此文件。如果该设置的值为 `base64`（未明确设置时的默认值），AWS CLI 希望该文件中包含 `base64` 编码文本。如果该设置的值为 `raw-in-base64-out`，AWS CLI 希望文件中包含原始二进制内容。

`--cli-connect-timeout` *<integer>*

指定最大套接字连接时间（以秒为单位）。如果该值设置为零（0），则套接字连接将无限等待（阻塞），不会超时。

`--cli-read-timeout` *<integer>*

指定最大套接字读取时间（以秒为单位）。如果该值设置为零（0），则套接字读取将无限等待（阻塞），不会超时。

`--color` *<string>*

指定对彩色输出的支持。有效值包括 `on`、`off` 和 `auto`。原设定值为 `auto`。

`--debug`

启用调试日志记录的布尔开关。默认情况下，AWS CLI 提供有关命令输出中命令结果的任何成功或失败的清理信息。`--debug` 选项提供完整的 Python 日志。这包括有关命令操作的额外 `stderr` 诊断信息，这些信息在排查命令提供意外结果的原因时非常有用。为了轻松查看调试日志，我们建议将日志发送到文件，这样可以更轻松地搜索信息。您可以使用以下方法之一实现这一点。

要仅发送 `stderr` 诊断信息，请附加 `2> debug.txt`，其中 `debug.txt` 是您要用于调试文件的名称：

```
$ aws servicename commandname options --debug 2> debug.txt
```

要同时发送输出信息和 `stderr` 诊断信息，请附加 `&> debug.txt`，其中 `debug.txt` 是您要用于调试文件的名称：

```
$ aws servicename commandname options --debug &> debug.txt
```

### `--endpoint-url <string>`

指定要将请求发送到的 URL。对于大多数命令，AWS CLI 会根据所选服务和指定的 AWS 区域自动确定 URL。但是，某些命令需要您指定账户专用 URL。您还可以配置一些 AWS 服务[直接在您的私有 VPC 中托管终端节点](#)（然后可能需要指定该终端节点）。

以下命令示例使用自定义 Amazon S3 端点 URL。

```
$ aws s3 ls --endpoint-url http://localhost:4567
```

端点配置设置位于多个位置（例如，系统变量或用户环境变量、本地 AWS 配置文件），或者在命令行上显式声明为参数。AWS CLI 端点配置设置的优先顺序如下：

1. `--endpoint-url` 命令行选项。
2. 如果启用，则 `AWS_IGNORE_CONFIGURED_ENDPOINT_URLS` 全局端点环境变量或配置文件设置 `ignore_configure_endpoint_urls` 将忽略自定义端点。
3. 由特定于服务的环境变量 `AWS_ENDPOINT_URL_<SERVICE>` 提供的值，例如 `AWS_ENDPOINT_URL_DYNAMODB`。
4. `AWS_USE_DUALSTACK_ENDPOINT`、`AWS_USE_FIPS_ENDPOINT` 和 `AWS_ENDPOINT_URL` 环境变量提供的值。
5. 共享 config 文件 `services` 节中的 `endpoint_url` 设置提供的特定于服务的端点值。
6. 共享 config 文件的 `profile` 中的 `endpoint_url` 设置提供的值。
7. `use_dualstack_endpoint`、`use_fips_endpoint` 和 `endpoint_url` 设置。
8. 最后使用相应的 AWS 服务的任何默认端点 URL。有关每个区域可用的标准服务端点的列表，请参阅《Amazon Web Services 一般参考》中的 [AWS 区域和端点](#)。

### `--no-cli-auto-prompt`

禁用单个命令的自动提示模式。

```
$ aws dynamodb describe-table --table-name Table1 --no-cli-auto-prompt
```

此选项将覆盖 [aws\\_cli\\_auto\\_prompt](#) 环境变量和 [cli\\_auto\\_prompt](#) 配置文件设置。

有关 AWS CLI 版本 2 自动提示特征的信息，请参阅 [让 AWS CLI 提示您输入命令](#)。

`--no-clipager`

一个布尔开关，它对命令输出禁用分页程序。

`--no-paginate`

一个布尔开关，它禁用多次调用，AWS CLI 自动发出这些调用来接收创建输出分页的所有命令结果。这意味着只显示您的输出的第一页。

`--no-sign-request`

对 AWS 服务终端节点的 HTTP 请求禁用签名的布尔开关。这可避免加载凭证。

`--no-verify-ssl`

默认情况下，AWS CLI 在与 AWS 服务通信时使用 SSL。对于每个 SSL 连接和调用，AWS CLI 都会验证 SSL 证书。使用此选项会覆盖验证 SSL 证书的默认行为。

#### Warning

此选项不是最佳做法。如果您使用 `--no-verify-ssl`，则您的客户端和 AWS 服务之间的流量将不再受到保护。这意味着您的流量存在安全风险，容易受到中间人攻击。如果您遇到证书问题，最好解决这些问题。有关证书故障排除步骤，请参阅 [the section called “SSL 证书错误”](#)。

`--output <string>`

指定用于该命令的输出格式。您可以指定以下任意值：

- [json](#) – 输出采用 [JSON](#) 字符串的格式。
- [yaml](#) – 输出采用 [YAML](#) 字符串的格式。
- [yaml-stream](#) – 输出被流式处理并采用 [YAML](#) 字符串的格式。串流支持更快地处理大型数据类型。
- [text](#) – 输出采用多个制表符分隔字符串值行的格式。这对于将输出传递到文本处理器（如 `grep`、`sed` 或 `awk`）很有用。
- [table](#) – 输出采用表格形式，使用字符 `+|-` 以形成单元格边框。它通常以“人性化”格式呈现信息，这种格式比其他格式更容易阅读，但从编程方面来讲不是那么有用。

**--profile <string>**

指定用于该命令的[命名配置文件](#)。要设置其他命名配置文件，可以在 `aws configure` 命令中使用 `--profile` 选项。

```
$ aws configure --profile <profilename>
```

**--query <string>**

指定用于筛选响应数据的 [JMESPath 查询](#)。有关更多信息，请参阅[过滤器 AWS CLI 输出](#)。

**--region <string>**

指定要将该命令的 AWS 请求发送到的 AWS 区域。有关您可以指定的所有区域的列表，请参阅《Amazon Web Services 一般参考》中的[AWS 区域和端点](#)。

**--version**

显示正在运行的 AWS CLI 程序的当前版本的布尔开关。

## 命令行选项的常见用法

常见的命令行选项用法包括在编写脚本时检查多个 AWS 区域中的资源，以及更改输出格式使其易于阅读或使用。在以下示例中，我们对每个区域运行 `describe-instances` 命令，直到我们找到实例所在的区域。

```
$ aws ec2 describe-instances --output table --region us-west-1
-----
|DescribeInstances|
+-----+
$ aws ec2 describe-instances --output table --region us-west-2
-----
|
|                                DescribeInstances                                |
+-----+
||                                Reservations                                ||
|+-----+-----+-----+-----+|
||  OwnerId                       | 012345678901                               ||
||  ReservationId                  | r-abcdefgh                               ||
|+-----+-----+-----+-----+|
|||                                Instances                                ||| |
||+-----+-----+-----+-----+||
|||  AmiLaunchIndex                | 0                                          |||
|||  Architecture                   | x86_64                                   |||
```

...

## 命令完成

AWS Command Line Interface (AWS CLI) 包含一个与 Bash 兼容的命令完成功能，让您可以使用 Tab 键完成部分输入的命令。在大多数系统上，您需要手动配置此功能。

有关 AWS CLI 版本 2 自动提示功能的信息，请改为参阅 [让 AWS CLI 提示您输入命令](#)。

### 主题

- [工作原理](#)
- [在 Linux 或 macOS 上配置命令完成](#)
- [在 Windows 上配置命令完成](#)

## 工作原理

当您部分输入命令、参数或选项时，命令完成功能会自动完成您的命令或显示建议的命令列表。要提示命令完成，请输入命令的一部分并按完成键（在大多数 Shell 中，它通常为 **Tab** 键）。

以下示例显示了可以使用命令完成的不同方法：

- 部分输入命令，然后按 **Tab** 键以显示建议的命令列表。

```
$ aws dynamodb dTAB
delete-backup                describe-global-table
delete-item                  describe-global-table-settings
delete-table                 describe-limits
describe-backup              describe-table
describe-continuous-backups  describe-table-replica-auto-scaling
describe-contributor-insights describe-time-to-live
describe-endpoints
```

- 部分输入参数，然后按 **Tab** 键显示建议的参数列表。

```
$ aws dynamodb delete-table --TAB
--ca-bundle                --endpoint-url            --profile
--cli-connect-timeout      --generate-cli-skeleton  --query
--cli-input-json           --no-paginate            --region
--cli-read-timeout         --no-sign-request        --table-name
--color                    --no-verify-ssl          --version
```

```
--debug          --output
```

- 输入参数并按 **Tab** 键以显示建议的资源值列表。此功能仅在 AWS CLI 版本 2 中可用。

```
$ aws dynamodb db delete-table --table-name TAB
Table 1          Table 2          Table 3
```

## 在 Linux 或 macOS 上配置命令完成

要在 Linux 或 macOS 上配置命令完成，您必须知道所使用的 Shell 的名称和 `aws_completer` 脚本的位置。

### Note

默认情况下，在运行 Amazon Linux 的 Amazon EC2 实例上自动配置和启用命令完成。

### 主题

- [确认完成标签的文件夹在您的路径中](#)
- [启用命令完成](#)
- [验证命令完成](#)

## 确认完成标签的文件夹在您的路径中

要让 AWS 完成标签成功运行，`aws_completer` 必须在您的 Shell 路径中。`which` 命令可以检查完成标签是否在您的路径中。

```
$ which aws_completer
/usr/local/bin/aws_completer
```

如果 `which` 命令找不到完成标签，则按照以下步骤将完成标签的文件夹添加到您的路径中。

### 步骤 1：定位 AWS 完成标签

AWS 完成标签的位置可能随所用安装方法而异。

- 程序包管理器 – `pip`、`yum`、`brew` 和 `apt-get` 等程序通常在标准路径位置安装 AWS 完成标签（或其符号链接）。

- 如果您使用没有 pip 参数的 `--user`，则默认路径为 `/usr/local/bin/aws_completer`。
- 如果您使用包含 pip 参数的 `--user`，则默认路径为 `/home/username/.local/bin/aws_completer`。
- 捆绑安装程序 – 如果您使用捆绑安装程序，则默认路径为 `/usr/local/bin/aws_completer`。

如果所有其他操作都失败，您可以使用 `find` 命令在您的文件系统中搜索 AWS 完成标签。

```
$ find / -name aws_completer
/usr/local/bin/aws_completer
```

## 步骤 2：识别 Shell

要识别您正在使用的 Shell，可以使用以下命令之一。

- `echo $SHELL` – 显示 Shell 的程序文件名称。这通常会与所使用的 Shell 的名称匹配，除非您在登录后启动了不同的 Shell。

```
$ echo $SHELL
/bin/bash
```

- `ps` – 显示为当前用户运行的进程。其中之一是 Shell。

```
$ ps
  PID TTY          TIME CMD
 2148 pts/1    00:00:00 bash
 8756 pts/1    00:00:00 ps
```

## 步骤 3：将完成标签添加到您的路径中

1. 在您的用户文件夹中查找 Shell 的配置文件脚本。

```
$ ls -a ~/
.  ..  .bash_logout  .bash_profile  .bashrc  Desktop  Documents  Downloads
```

- Bash– `.bash_profile`、`.profile` 或 `.bash_login`
- Zsh– `.zshrc`
- Tcsh– `.tcshrc`、`.cshrc` 或 `.login`

- 在配置文件脚本末尾添加与以下示例类似的导出命令。将 `/usr/local/bin/` 替换为您在上一部分中找到的文件夹。

```
export PATH=/usr/local/bin/:$PATH
```

- 将配置文件重新加载到当前会话中，以使更改生效。将 `.bash_profile` 替换为您在第一部分中找到的 shell 脚本的名称。

```
$ source ~/.bash_profile
```

## 启用命令完成

确认完成标签在您的路径中后，通过运行正在使用的 Shell 的相应命令来启用命令完成。您可以将命令添加到 Shell 的配置文件中，以便在每次打开一个新 Shell 时运行它。在每个命令中，将 `/usr/local/bin/` 路径替换为 [确认完成标签的文件夹在您的路径中](#) 中在您的系统上找到的那个。

- bash** – 使用内置命令 `complete`。

```
$ complete -C '/usr/local/bin/aws_completer' aws
```

将之前的命令添加到 `~/.bashrc` 中，以便在每次打开一个新外壳程序时运行它。您的 `~/.bash_profile` 应指定源 `~/.bashrc`，以确保该命令也在登录 Shell 中运行。

- zsh** – 要运行命令完成功能，您需要在 `~/.zshrc` 配置文件脚本的末尾添加以下自动加载行来运行 `bashcompinit`。

```
$ autoload bashcompinit && bashcompinit
$ autoload -Uz compinit && compinit
```

要启用命令完成，请使用内置命令 `complete`。

```
$ complete -C '/usr/local/bin/aws_completer' aws
```

将之前的命令添加到 `~/.zshrc` 中，以便在每次打开一个新外壳程序时运行它。

- tcsh** – `tcsh` 的完成采用字类型和样式来定义完成行为。

```
> complete aws 'p/*/'`aws_completer`/`
```



将之前的命令添加到 `~/.tschrc` 中，以便在每次打开一个新外壳程序时运行它。

启用命令完成后，[验证命令完成](#) 正在工作。

## 验证命令完成

启用命令完成后，重新加载 Shell，输入部分命令并按 Tab 查看可用命令。

```
$ aws sTAB
s3          ses          sqs          sts          swf
s3api       sns          storagegateway support
```

## 在 Windows 上配置命令完成

### Note

有关 PowerShell 如何处理其完成情况（包括各种完成密钥）的信息，请参阅 Microsoft PowerShell Docs 中的 [about\\_Tab\\_Expansion](#)。

要在 Windows 上启用 PowerShell 的命令完成，请在 PowerShell 中完成以下步骤。

1. 使用以下命令打开你的 `$PROFILE`。

```
PS C:\> Notepad $PROFILE
```

如果没有 `$PROFILE`，请使用以下命令创建用户配置文件。

```
PS C:\> if (!(Test-Path -Path $PROFILE ))
{ New-Item -Type File -Path $PROFILE -Force }
```

有关 PowerShell 配置文件的详细信息，请参阅 Microsoft Docs 网站上的[如何在 Windows PowerShell ISE 中使用配置文件](#)。

2. 要启用命令完成，请将以下代码块添加到您的配置文件中，保存，然后关闭文件。

```
Register-ArgumentCompleter -Native -CommandName aws -ScriptBlock {
    param($commandName, $wordToComplete, $cursorPosition)
```

```

$env:COMP_LINE=$wordToComplete
if ($env:COMP_LINE.Length -lt $cursorPosition){
    $env:COMP_LINE=$env:COMP_LINE + " "
}
$env:COMP_POINT=$cursorPosition
aws_completer.exe | ForEach-Object {
    [System.Management.Automation.CompletionResult]::new($_, $_,
'ParameterValue', $_)
}
Remove-Item Env:\COMP_LINE
Remove-Item Env:\COMP_POINT
}

```

3. 启用命令完成功能后，重新加载 Shell，输入命令的一部分并按 Tab 可循环浏览可用命令。

```
$ aws sTab
```

```
$ aws s3
```

要查看完成后可用的所有命令，请输入命令的一部分并按 Ctrl + 空格键。

```

$ aws sCtrl + Space
s3          ses          sqs          sts          swf
s3api       sns          storagegateway support

```

## AWS CLI 重试次数

本主题介绍在 AWS CLI 中，可能由于意外问题而导致 AWS 服务调用失败。这些问题可能发生在服务器端，也可能是由于您尝试调用的 AWS 服务存在速率限制而失败。这些类型的故障通常不需要特殊处理，并且通常在短暂的等待时间段后会重新发出调用。AWS CLI 提供了许多功能，在出现这些类型的错误或异常的情况下，帮助重试对 AWS 服务进行的客户端调用。

### 主题

- [可用重试模式](#)
- [配置重试模式](#)
- [查看重试日志](#)

## 可用重试模式

AWS CLI 有多种模式可供选择，具体取决于您的版本：

- [传统重试模式](#)
- [标准重试模式](#)
- [自适应重试模式](#)

### 传统重试模式

传统模式使用旧的重试处理程序，其功能有限，其中包括：

- 最大重试次数的默认值为 4，总共可发出 5 次调用尝试。此值可通过 `max_attempts` 配置参数覆盖。
- DynamoDB 的最大重试次数的原定设置值为 9，总共可发出 10 次调用尝试。此值可通过 `max_attempts` 配置参数覆盖。
- 重试以下有限数量的错误/异常：
  - 常规套接字/连接错误：
    - `ConnectionError`
    - `ConnectionClosedError`
    - `ReadTimeoutError`
    - `EndpointConnectionError`
  - 服务端限制错误和异常：
    - `Throttling`
    - `ThrottlingException`
    - `ThrottledException`
    - `RequestThrottledException`
    - `ProvisionedThroughputExceededException`
- 对多个 HTTP 状态代码执行重试操作，包括 429、500、502、503、504 和 509。
- 任何重试都将包含基准因子为 2 的指数回退。

## 标准重试模式

标准模式是一组跨 AWS SDK 的标准重试规则，其功能比传统模式更多。此模式是 AWS CLI 版本 2 的原定设置模式。标准模式为 AWS CLI 版本 2 而创建，并已反向移植到 AWS CLI 版本 1。标准模式的功能包括：

- 最大重试次数的默认值为 2，总共可发出 3 次调用尝试。此值可通过 `max_attempts` 配置参数覆盖。
- 对以下更加广泛的错误/异常列表重试操作：
  - 瞬时错误/异常
    - RequestTimeout
    - RequestTimeoutException
    - PriorRequestNotComplete
    - ConnectionError
    - HTTPClientError
  - 服务端限制错误和异常：
    - Throttling
    - ThrottlingException
    - ThrottledException
    - RequestThrottledException
    - TooManyRequestsException
    - ProvisionedThroughputExceededException
    - TransactionInProgressException
    - RequestLimitExceeded
    - BandwidthLimitExceeded
    - LimitExceededException
    - RequestThrottled
    - SlowDown
    - EC2ThrottledException
- 对非描述性的瞬时错误代码进行重试。具体来说，这些 HTTP 状态代码包括：

500、502、503、504。

- 任何重试都将包含基准因子为 2 的指数回退，最长回退时间为 20 秒。

## 自适应重试模式

### Warning

自适应模式是一种试验模式，在特征和行为方面可能会发生变化。

自适应重试模式是一种试验性重试模式，包括标准模式的所有功能。除了标准模式功能外，自适应模式还通过使用令牌存储桶和速率限制变量引入了客户端速率限制，这些变量会随着每次重试而动态更新。此模式为客户端重试提供了灵活性，能够适应AWS服务的错误/异常状态响应。

对于每个新的重试，自适应模式都会根据 AWS 服务响应中显示的错误、异常或 HTTP 状态代码修改速率限制变量。然后，使用这些速率限制变量来计算客户端的新调用速率。对于AWS服务的每个异常/错误或不成功 HTTP 响应（在上面的列表中提供），都会在重试时更新速率限制变量，直到达到成功、令牌存储桶用尽或达到配置的最大尝试次数值。

## 配置重试模式

AWS CLI 包括各种重试配置，以及创建客户端对象时要考虑的配置方法。

### 可用配置方法

在 AWS CLI 中，用户可以通过以下方式配置重试：

- 环境变量
- AWS CLI 配置文件

用户可以自定义以下重试选项：

- 重试模式 - 指定 AWS CLI 使用的重试模式。如上所述，有三种重试模式可用：传统模式、标准模式和自适应模式。AWS CLI 版本 2 的原定设置值为标准模式。
- 最大尝试次数 - 指定 AWS CLI 重试处理程序使用的最大重试次数值，其中初始调用会计入您提供的值。默认值是 5。

## 在环境变量中定义重试配置

要为 AWS CLI 定义您的重试配置，请更新操作系统的环境变量。

重试环境变量包括：

- AWS\_RETRY\_MODE
- AWS\_MAX\_ATTEMPTS

有关环境变量的更多信息，请参阅[用于配置的环境变量 AWS CLI](#)。

## 在AWS配置文件中定义重试配置

要更改重试配置，请更新您的全局 AWS 配置文件。您的AWS配置文件的默认位置为 `~/.aws/config`。

下面是AWS配置文件的示例：

```
[default]
retry_mode = standard
max_attempts = 6
```

有关配置文件的更多信息，请参阅[配置和凭证文件设置](#)。

## 查看重试日志

AWS CLI 使用 Boto3 的重试方法和日志记录。您可以对任何命令使用 `--debug` 选项来接收调试日志。有关如何使用 `--debug` 选项的更多信息，请参阅[命令行选项](#)。

如果您在调试日志中搜索“retry”，将会找到所需的重试信息。重试操作的客户端日志条目取决于您启用的重试模式。

传统模式：

重试消息是由 `botocore.retryhandler` 生成的。您将看到以下三个消息之一：

- No retry needed
- Retry needed, action of: *<action\_name>*
- Reached the maximum number of retry attempts: *<attempt\_number>*

标准模式或自适应模式：

重试消息是由 `botocore.retries.standard` 生成的。您将看到以下三个消息之一：

- No retrying request
- Retry needed, retrying request after delay of: `<delay_value>`
- Retry needed but retry quota reached, not retrying request

有关 botocore 重试的完整定义文件，请参阅 botocore GitHub 存储库 上的 [\\_retry.json](#)。

## 使用 HTTP 代理

要通过代理服务器访问 AWS，您可以使用代理服务器使用的 DNS 域名或 IP 地址和端口号配置 `HTTP_PROXY` 和 `HTTPS_PROXY` 环境变量。

主题

- [使用示例](#)
- [向代理进行身份验证](#)
- [对 Amazon EC2 实例使用代理](#)
- [故障排除](#)

## 使用示例

### Note

以下示例显示了全部使用大写字母的环境变量名称。但是，如果使用不同的大小写指定一个变量两次，则优先使用小写字母。建议您只定义每个变量一次，以避免系统混淆和意外行为。

以下示例显示如何使用代理的显式 IP 地址或解析为代理 IP 地址的 DNS 名称。两种情况都可以后跟冒号和应将查询发送到的端口号。

Linux or macOS

```
$ export HTTP_PROXY=http://10.15.20.25:1234
$ export HTTP_PROXY=http://proxy.example.com:1234
```

```
$ export HTTPS_PROXY=http://10.15.20.25:5678
$ export HTTPS_PROXY=http://proxy.example.com:5678
```

## Windows Command Prompt

为所有会话设置

```
C:\> setx HTTP_PROXY http://10.15.20.25:1234
C:\> setx HTTP_PROXY http://proxy.example.com:1234
C:\> setx HTTPS_PROXY http://10.15.20.25:5678
C:\> setx HTTPS_PROXY http://proxy.example.com:5678
```

使用 [setx](#) 设置环境变量会更改当前命令提示符会话和运行该命令后创建的所有命令提示符会话中使用的值。它不影响在运行该命令时已经运行的其他命令 shell。

仅为当前会话设置

使用 [set](#) 设置环境变量会更改使用的值，直到当前命令提示符会话结束，或者直到您将该变量设置为其他值。

```
C:\> set HTTP_PROXY=http://10.15.20.25:1234
C:\> set HTTP_PROXY=http://proxy.example.com:1234
C:\> set HTTPS_PROXY=http://10.15.20.25:5678
C:\> set HTTPS_PROXY=http://proxy.example.com:5678
```

## 向代理进行身份验证

### Note

AWS CLI 不支持 NTLM 代理。如果使用 NTLM 或 Kerberos 协议代理，则可以通过身份验证代理（如 [Cntlm](#)）进行连接。

AWS CLI 支持 HTTP 基本身份验证。在代理 URL 中指定用户名和密码，如下所示。

Linux or macOS

```
$ export HTTP_PROXY=http://username:password@proxy.example.com:1234
$ export HTTPS_PROXY=http://username:password@proxy.example.com:5678
```



## Windows Command Prompt

为所有会话设置

```
C:\> setx HTTP_PROXY http://username:password@proxy.example.com:1234  
C:\> setx HTTPS_PROXY http://username:password@proxy.example.com:5678
```

仅为当前会话设置

```
C:\> set HTTP_PROXY=http://username:password@proxy.example.com:1234  
C:\> set HTTPS_PROXY=http://username:password@proxy.example.com:5678
```

## 对 Amazon EC2 实例使用代理

如果是在使用附加 IAM 角色启动的 Amazon EC2 实例上配置代理，请确保排除用于访问[实例元数据](#)的地址。为此，请将 NO\_PROXY 环境变量设置为实例元数据服务的 IP 地址 169.254.169.254。该地址保持不变。

Linux or macOS

```
$ export NO_PROXY=169.254.169.254
```

## Windows Command Prompt

为所有会话设置

```
C:\> setx NO_PROXY 169.254.169.254
```

仅为当前会话设置

```
C:\> set NO_PROXY=169.254.169.254
```

## 故障排除

如果您遇到有关 AWS CLI 的问题，请参阅[排查错误](#)以了解故障排除步骤。有关相关性最高的故障排除步骤，请参阅[the section called “SSL 证书错误”](#)。

## 在中使用终端节点 AWS CLI

要以编程方式连接到 AWS 服务，请使用终端节点。端点是 AWS Web 服务入口点的 URL。AWS Command Line Interface (AWS CLI) 会自动为中的每项服务使用默认终端节点 AWS 区域，但您可以为 API 请求指定备用终端节点。

### 端点主题

- [为单个命令设置端点](#)
- [为所有人设置全局终端节点 AWS 服务](#)
- [设置成为所有 AWS 服务使用 FIPS 端点](#)
- [设置成为所有 AWS 服务使用双堆栈端点](#)
- [设置特定于服务的端点](#)
  - [特定于服务的端点：环境变量](#)
  - [特定于服务的端点：共享 config 文件](#)
  - [特定于服务的端点：特定于服务的标识符列表](#)
- [端点配置和设置优先级](#)

### 为单个命令设置端点

要覆盖单个命令的任何端点设置或环境变量，请使用 [--endpoint-url](#) 命令行选项。以下命令示例使用自定义 Amazon S3 端点 URL。

```
$ aws s3 ls --endpoint-url http://localhost:4567
```

### 为所有人设置全局终端节点 AWS 服务

要将所有服务的请求路由到自定义端点 URL，请使用以下设置之一：

- 环境变量：
  - [AWS\\_IGNORE\\_CONFIGURED\\_ENDPOINT\\_URLS](#) – 忽略已配置的端点 URL。

Linux or macOS

```
$ export AWS_IGNORE_CONFIGURED_ENDPOINT_URLS=true
```

## Windows Command Prompt

为所有会话设置

```
C:\> setx AWS_IGNORE_CONFIGURED_ENDPOINT_URLS true
```

仅为当前会话设置

```
C:\> set AWS_IGNORE_CONFIGURED_ENDPOINT_URLS=true
```

## PowerShell

```
PS C:\> $Env:AWS_IGNORE_CONFIGURED_ENDPOINT_URLS="true"
```

- [AWS\\_ENDPOINT\\_URL](#) – 设置全局端点 URL。

Linux or macOS

```
$ export AWS_ENDPOINT_URL=http://localhost:4567
```

## Windows Command Prompt

为所有会话设置

```
C:\> setx AWS_ENDPOINT_URL http://localhost:4567
```

仅为当前会话设置

```
C:\> set AWS_ENDPOINT_URL=http://localhost:4567
```

## PowerShell

```
PS C:\> $Env:AWS_ENDPOINT_URL="http://localhost:4567"
```

- config 文件：
- [ignore\\_configure\\_endpoint\\_urls](#) – 忽略已配置的端点 URL。

```
ignore_configure_endpoint_urls = true
```

- [endpoint\\_url](#) – 设置全局端点 URL。

```
endpoint_url = http://localhost:4567
```

特定于服务的端点和 `--endpoint-url` 命令行选项会覆盖所有全局端点。

## 设置成为所有 AWS 服务使用 FIPS 端点

要路由所有服务的请求以使用 FIPS 端点，请使用以下设置之一：

- [AWS\\_USE\\_FIPS\\_ENDPOINT](#) 环境变量。

Linux or macOS

```
$ export AWS_USE_FIPS_ENDPOINT=true
```

Windows Command Prompt

为所有会话设置

```
C:\> setx AWS_USE_FIPS_ENDPOINT true
```

仅为当前会话设置

```
C:\> set AWS_USE_FIPS_ENDPOINT=true
```

PowerShell

```
PS C:\> $Env:AWS_USE_FIPS_ENDPOINT="true"
```

- [use\\_fips\\_endpoint](#) 文件设置。

```
use_fips_endpoint = true
```

有些 AWS 服务提供的端点支持[联邦信息处理标准 \(FIPS\) 140-2](#)。AWS 区域当 AWS 服务支持 FIPS 时，此设置指定 AWS CLI 应使用哪个 FIPS 端点。与标准 AWS 端点不同，FIPS 端点使用符合 FIPS 140-2 的 TLS 软件库。与美国政府有业务来往的企业可能需要使用这些端点。

如果启用了此设置，但您的服务中不存在 FIPS 端点 AWS 区域，则该 AWS 命令可能会失败。在这种情况下，请使用 [--endpoint-url](#) 选项手动指定要在命令中使用的端点，或者使用[特定于服务的端点](#)。

有关通过指定 FIPS 端点的更多信息 AWS 区域，请参阅按服务划分的 [FIPS 终端节点](#)。

## 设置成为所有 AWS 服务使用双堆栈端点

要路由所有服务的请求以在可用时使用双堆栈端点，请使用以下设置之一：

- [AWS\\_USE\\_DUALSTACK\\_ENDPOINT](#) 环境变量。

Linux or macOS

```
$ export AWS_USE_DUALSTACK_ENDPOINT=true
```

Windows Command Prompt

为所有会话设置

```
C:\> setx AWS_USE_DUALSTACK_ENDPOINT true
```

仅为当前会话设置

```
C:\> set AWS_USE_DUALSTACK_ENDPOINT=true
```

PowerShell

```
PS C:\> $Env:AWS_USE_DUALSTACK_ENDPOINT="true"
```

- [use\\_dualstack\\_endpoint](#) 文件设置。

```
use_dualstack_endpoint = true
```

允许使用双堆栈端点发送 AWS 请求。要详细了解支持 IPv4 和 IPv6 流量的双堆栈端点，请参阅 Amazon Simple Storage Service 用户指南中的[使用 Amazon S3 双堆栈端点](#)。双堆栈端点适用于某些区域。如果服务不存在双堆栈端点或 AWS 区域，则请求将失败。默认情况下，将禁用该功能。

## 设置特定于服务的端点

服务特定的终端节点配置提供了使用您选择的永久终端节点来 AWS CLI 处理请求的选项。这些设置可以灵活地支持本地终端节点、VPC 终端节点和第三方本地 AWS 开发环境。不同的端点可分别用于测试环境和生产环境。您可以为个别 AWS 服务指定端点 URL。

可通过以下方式指定特定于服务的端点：

- 单个命令的命令行选项 [--endpoint-url](#)。
- 环境变量：
  - [AWS\\_IGNORE\\_CONFIGURED\\_ENDPOINT\\_URLS](#) – 忽略所有已配置的端点 URL，除非在命令行中指定。
  - [AWS\\_ENDPOINT\\_URL\\_<SERVICE>](#) – 指定用于特定服务的自定义端点，在该服务中 <SERVICE> 替换为 AWS 服务 标识符。有关所有特定于服务的变量，请参阅[the section called “特定于服务的标识符列表”](#)。
- config 文件：
  - [ignore\\_configure\\_endpoint\\_urls](#) – 忽略所有已配置的端点 URL，除非使用环境变量或在命令行中指定。
  - config 文件的 [services](#) 节与 [endpoint\\_url](#) 文件设置相结合。

特定于服务的端点主题：

- [特定于服务的端点：环境变量](#)
- [特定于服务的端点：共享 config 文件](#)
- [特定于服务的端点：特定于服务的标识符列表](#)

### 特定于服务的端点：环境变量

环境变量会覆盖配置文件中的设置，但不会覆盖命令行上指定的选项。如果您希望所有配置文件在设备上使用相同的端点，请使用环境变量。

以下是特定于服务的环境变量：

- [AWS\\_IGNORE\\_CONFIGURED\\_ENDPOINT\\_URLS](#) – 忽略所有已配置的端点 URL，除非在命令行中指定。

## Linux or macOS

```
$ export AWS_IGNORE_CONFIGURED_ENDPOINT_URLS=true
```

## Windows Command Prompt

为所有会话设置

```
C:\> setx AWS_IGNORE_CONFIGURED_ENDPOINT_URLS true
```

仅为当前会话设置

```
C:\> set AWS_IGNORE_CONFIGURED_ENDPOINT_URLS=true
```

## PowerShell

```
PS C:\> $Env:AWS_IGNORE_CONFIGURED_ENDPOINT_URLS="true"
```

- [AWS\\_ENDPOINT\\_URL\\_<SERVICE>](#)-指定用于特定服务的自定义终端节点，其中替换<SERVICE>为标 AWS 服务 标识符。有关所有特定于服务的变量，请参阅[the section called “特定于服务的标识符列表”](#)。

以下环境变量示例设置 AWS Elastic Beanstalk的端点。

## Linux or macOS

```
$ export AWS_ENDPOINT_URL_ELASTIC_BEANSTALK=http://localhost:4567
```

## Windows Command Prompt

为所有会话设置

```
C:\> setx AWS_ENDPOINT_URL_ELASTIC_BEANSTALK http://localhost:4567
```

仅为当前会话设置

```
C:\> set AWS_ENDPOINT_URL_ELASTIC_BEANSTALK=http://localhost:4567
```

## PowerShell

```
PS C:\> $Env:AWS_ENDPOINT_URL_ELASTIC_BEANSTALK="http://localhost:4567"
```

有关设置环境变量的更多信息，请参阅[the section called “环境变量”](#)。

### 特定于服务的端点：共享 **config** 文件

在共享 config 文件中，`endpoint_url` 用在多个节中。要设置特定于服务的端点，请使用嵌套在 `services` 节内服务标识符密钥下的 `endpoint_url` 设置。有关在共享 config 文件中定义 `services` 节的详细信息，请参阅 [the section called “services”](#)。

以下示例使用 `services` 节为 Amazon S3 配置服务特定的端点 URL 和用于其他所有服务的自定义全局端点：

```
[profile dev1]  
endpoint_url = http://localhost:1234  
services = s3-specific  
  
[services testing-s3]  
s3 =  
    endpoint_url = http://localhost:4567
```

单个配置文件可以为多个服务配置端点。以下示例在同一配置文件中为 Amazon S3 和 AWS Elastic Beanstalk 设置服务特定的端点 URL。

有关 `services` 节中要使用的所有服务标识符密钥的列表，请参阅[特定于服务的标识符列表](#)。

```
[profile dev1]  
services = testing-s3-and-eb  
  
[services testing-s3-and-eb]  
s3 =  
    endpoint_url = http://localhost:4567  
elastic_beanstalk =  
    endpoint_url = http://localhost:8000
```

“服务配置”节可以在多个配置文件中使用。以下示例有两个配置文件使用相同的 `services` 定义：

```
[profile dev1]
```



```
output = json
services = testing-s3

[profile dev2]
output = text
services = testing-s3

[services testing-s3]
s3 =
  endpoint_url = https://localhost:4567
```

## 特定于服务的端点：特定于服务的标识符列表

标 AWS 服务 标识符基于 API 模型，将所有空格 `serviceId` 替换为下划线，并将所有字母小写。

以下服务标识符示例使用 AWS Elastic Beanstalk。AWS Elastic Beanstalk 有 `o serviceId` `fElastic Beanstalk`，因此服务标识符密钥为 `elastic_beanstalk`。

下表列出了所有特定于服务的标识符、config 文件密钥和环境变量。

## 端点配置和设置优先级

端点配置设置位于多个位置，例如系统或用户环境变量、本地 AWS 配置文件或在命令行中明确声明为参数。AWS CLI 端点配置设置的优先顺序如下：

1. `--endpoint-url` 命令行选项。
2. 如果启用，则 `AWS_IGNORE_CONFIGURED_ENDPOINT_URLS` 全局端点环境变量或配置文件设置 `ignore_configure_endpoint_urls` 将忽略自定义端点。
3. 由特定于服务的环境变量 `AWS_ENDPOINT_URL_<SERVICE>` 提供的值，例如 `AWS_ENDPOINT_URL_DYNAMODB`。
4. `AWS_USE_DUALSTACK_ENDPOINT`、`AWS_USE_FIPS_ENDPOINT` 和 `AWS_ENDPOINT_URL` 环境变量提供的值。
5. 共享 config 文件 `services` 节中的 `endpoint_url` 设置提供的特定于服务的端点值。
6. 共享 config 文件的 `profile` 中的 `endpoint_url` 设置提供的值。
7. `use_dualstack_endpoint`、`use_fips_endpoint` 和 `endpoint_url` 设置。
8. 最后使用相应 AWS 服务 端点 URL 的任何默认端点 URL。有关每个区域可用的标准服务端点的列表，请参阅《Amazon Web Services 一般参考》中的 [AWS 区域和端点](#)。

## 身份验证和访问凭证

当你使用 AWS 服务 AWS CLI 进行开发 AWS 时，你必须确定如何进行身份验证。要为的编程访问配置凭证 AWS CLI，请选择以下选项之一。这些选项按推荐顺序排列。

哪个用户需要程式访问权限？	用途	说明
员工身份 ( AWS IAM Identity Center 用户 )	( 推荐 ) 使用短期凭证。	<a href="#">the section called “IAM Identity Center 身份验证”</a>
IAM	使用短期凭证。	<a href="#">the section called “短期凭证”</a>
IAM 或员工身份 ( AWS IAM Identity Center 用户 )	将 Amazon EC2 实例元数据用于凭证。	<a href="#">the section called “将凭证用于 Amazon EC2 实例元数据”</a>
IAM 或员工身份 ( AWS IAM Identity Center 用户 )	将另一个凭证方法配对并代入一个角色以获得权限。	<a href="#">the section called “IAM 角色”</a>
IAM	( 不推荐 ) 使用长期凭证。	<a href="#">the section called “IAM 用户”</a>
IAM 或员工身份 ( AWS IAM Identity Center 用户 )	( 不推荐 ) 将另一个凭证方法配对，但使用存储在 AWS CLI 的外部位置的凭证值。	<a href="#">the section called “外部凭证”</a>

## 配置和凭证优先顺序

凭据和配置设置位于多个位置，例如系统或用户环境变量、本地 AWS 配置文件或在命令行中明确声明为参数。某些身份验证优先于其他身份验证。AWS CLI 身份验证设置的优先顺序如下：

1. [命令行选项](#) – 覆盖任何其他位置的设置，例如 `--region`、`--output` 和 `--profile` 参数。
2. [环境变量](#) – 您可以在系统的环境变量中存储值。
3. [代入角色](#) – 通过配置或 `aws sts assume-role` 命令代入 IAM 角色的权限。

4. [使用 Web 身份代入角色](#) – 通过配置或 [aws sts assume-role](#) 命令使用 Web 身份代入 IAM 角色的权限。
5. [AWS IAM Identity Center](#)— 当您运行 `aws configure sso` 命令时，存储在 `config` 文件中的 IAM Identity Center 配置设置会更新。然后在您运行 `aws sso login` 命令时对凭据进行身份验证。`config` 文件位于 `~/.aws/config` (在 Linux 或 macOS 上) 或 `C:\Users\USERNAME\.aws\config` (在 Windows 上)。
6. [凭证文件](#) – 在运行命令 `aws configure` 时，将更新 `credentials` 和 `config` 文件。`credentials` 文件位于 `~/.aws/credentials` (在 Linux 或 macOS 上) 或 `C:\Users\USERNAME\.aws\credentials` (在 Windows 上)。
7. [自定义流程](#) – 从外部来源获取您的凭证。
8. [配置文件](#) – 在运行命令 `aws configure` 时，将更新 `credentials` 和 `config` 文件。`config` 文件位于 `~/.aws/config` (在 Linux 或 macOS 上) 或 `C:\Users\USERNAME\.aws\config` (在 Windows 上)。
9. [容器凭证](#) – 您可以将 IAM 角色与每个 Amazon Elastic Container Service (Amazon ECS) 任务定义关联。之后，该任务的容器就可以使用该角色的临时凭证。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的 [任务的 IAM 角色](#)。
- 10 [Amazon EC2 实例配置文件凭证](#) – 您可以将 IAM 角色与每个 Amazon Elastic Compute Cloud (Amazon EC2) 实例关联。之后，在该实例上运行的代码就可以使用该角色的临时凭证。凭证通过 Amazon EC2 元数据服务提供。有关更多信息，请参阅 [Amazon EC2 用户指南中的 Amazon EC2 的 IAM 角色](#) 和 IAM 用户指南中的 [使用实例配置文件](#)。

## 此部分中的其他主题

- [the section called “IAM Identity Center 身份验证”](#)
- [the section called “短期凭证”](#)
- [the section called “IAM 角色”](#)
- [the section called “IAM 用户”](#)
- [the section called “将凭证用于 Amazon EC2 实例元数据”](#)
- [the section called “外部凭证”](#)

## 配置 AWS CLI 为使用 AWS IAM Identity Center

主要有两种方法可以通过 AWS IAM Identity Center ( IAM Identity Center ) 对用户进行身份验证，以获取通过 config 文件运行 AWS Command Line Interface (AWS CLI) 命令的证书：

- ( 推荐 ) [SSO 令牌提供程序配置](#)。SSO 令牌提供商配置、您的 AWS SDK 或工具可以自动检索刷新的身份验证令牌。
- [遗留的不可刷新配置](#)。使用遗留的不可刷新配置时，您需要手动刷新令牌，因为它会定期过期。

使用 IAM Identity Center 时，您可以登录到 Active Directory、内置 IAM Identity Center 目录或[已连接到 IAM Identity Center 的另一个 IdP](#)。您可以将这些证书映射到 AWS Identity and Access Management (IAM) 角色以供您运行 AWS CLI 命令。

无论您使用哪个 IdP，IAM Identity Center 都会将这些区别抽象出来。例如，您可以连接 Microsoft Azure AD，如博客文章 [IAM Identity Center 的下一个演进](#) 中所述。

### Note

有关使用不使用账户 ID 和角色的不记名身份验证的信息，请参阅 Amazon CodeCatalyst 用户指南 CodeCatalyst 中的 [设置 AWS CLI 与一起使用](#)。

本节中的主题

- [配置 AWS CLI 为使用 IAM Identity Center 令牌提供商凭证，并自动刷新身份验证](#)
- [AWS IAM Identity Center 的旧版不可刷新配置](#)
- [使用 IAM Identity Center 命名配置文件](#)

## 配置 AWS CLI 为使用 IAM Identity Center 令牌提供商凭证，并自动刷新身份验证

本主题介绍如何配置以使用 AWS IAM Identity Center ( IAM Identity Center ) 令牌提供商配置对用户进行身份验证。使用此 SSO 令牌提供程序配置，您的 AWS SDK 或工具可以自动检索刷新的身份验证令牌。

使用 IAM Identity Center 时，您可以登录到 Active Directory、内置 IAM Identity Center 目录或[已连接到 IAM Identity Center 的另一个 IdP](#)。您可以将这些证书映射到 AWS Identity and Access Management (IAM) 角色以供您运行 AWS CLI 命令。

无论您使用哪个 IdP，IAM Identity Center 都会将这些区别抽象出来。例如，您可以连接 Microsoft Azure AD，如博客文章 [IAM Identity Center 的下一个演进](#) 中所述。

### Note

有关使用不使用账户 ID 和角色的不记名身份验证的信息，请参阅 Amazon CodeCatalyst 用户指南 CodeCatalyst 中的 [设置 AWS CLI 与一起使用](#)。

您可以使用 SSO 令牌提供程序配置根据应用程序的需要自动刷新身份验证令牌，也可以使用 [延长会话持续时间选项](#)。您可以通过下列方式来执行此配置：

- 自动，使用 `aws configure sso` 和 `aws configure sso-session` 命令。以下命令是向导，指导您完成配置文件和 `sso-session` 信息的配置：
  - 使用 [aws configure sso](#) 创建或编辑您的 `config` 配置文件和 `sso-session` 部分。
  - 使用 [aws configure sso-session](#) 仅创建或编辑 `sso-session` 部分。
- [手动](#)，通过编辑存储命名配置文件的 `config` 文件。

### 先决条件

- 安装 AWS CLI。有关更多信息，请参阅 [the section called “安装/更新”](#)。
- 您必须先有权访问 IAM Identity Center 中的 SSO 身份验证。选择以下方法之一来访问您的 AWS 证书。

我尚未通过 IAM Identity Center 确立访问权限

请按照《AWS IAM Identity Center 用户指南》的 [入门](#) 中的说明操作。此过程激活 IAM Identity Center，创建管理用户并添加适当的最低权限集。

### Note

创建应用最低权限权限的权限集。除非您的雇主已为此目的创建自定义权限集，否则建议使用预定义的 `PowerUserAccess` 权限集。

退出门户并再次登录以查看您的 AWS 账户 和 Administrator 或选项 `PowerUserAccess`。在使用开发工具包时选择 `PowerUserAccess`。这也可以帮助您找到有关编程访问的详细信息。

我已经 AWS 可以通过雇主管理的联合身份提供商 ( 例如 Azure AD 或 Okta ) 进行访问

AWS 通过身份提供商的门户网站登录。如果您的云管理员已授予您 `PowerUserAccess` ( 开发者 ) 权限，则您 AWS 账户 会看到您有权访问的权限和权限集。在您的权限集名称旁边，可以看到有关使用该权限集手动或以编程方式访问账户的选项。

自定义实现可能会产生不同的体验，例如不同的权限集名称。如果您不确定要使用哪个权限集，请联系 IT 团队以寻求帮助。

我已经 AWS 可以通过雇主管理的 AWS 访问门户进行访问

AWS 通过 AWS 访问门户登录。如果您的云管理员已授予您 `PowerUserAccess` ( 开发者 ) 权限，则您 AWS 账户 会看到您有权访问的权限和权限集。在您的权限集名称旁边，可以看到有关使用该权限集手动或以编程方式访问账户的选项。

我已经 AWS 可以通过雇主管理的联合自定义身份提供商进行访问

请联系您的 IT 团队以寻求帮助。

## 使用 `aws configure sso` 向导配置您的配置文件

在 AWS CLI 中同时配置 IAM Identity Center 配置文件和 `sso-session`

1. 通过执行以下操作收集您的 IAM Identity Center 信息：
  1. 在您的 AWS 访问门户中，选择用于开发的权限集，然后选择访问密钥链接。
  2. 在“获取凭据”对话框中，选择与您的操作系统匹配的选项卡。
  3. 选择 IAM Identity Center 凭证方法以获取运行 `aws configure sso` 所需的 SSO Start URL 和 SSO Region 值。
  4. 有关要注册哪些范围值的信息，请参阅《IAM Identity Center 用户指南》中的 [OAuth 2.0 访问范围](#)。
2. 在您的首选终端中，运行 `aws configure sso` 命令并提供您的 IAM Identity Center 起始 URL 和托管身份中心目录的 AWS 区域。

```
$ aws configure sso
SSO session name (Recommended): my-sso
SSO start URL [None]: https://my-sso-portal.awsapps.com/start
SSO region [None]: us-east-1
SSO registration scopes [None]: sso:account:access
```

### 3. AWS CLI 尝试打开您的默认浏览器并开始您的 IAM Identity Center 账户的登录过程。

```
Attempting to automatically open the SSO authorization page in your default browser.
```

如果 AWS CLI 无法打开浏览器，则会显示以下消息，其中包含有关如何手动启动登录过程的说明。

```
If the browser does not open or you wish to use a different device to authorize this request, open the following URL:
```

```
https://device.sso.us-west-2.amazonaws.com/
```

```
Then enter the code:
```

```
QCFK-N451
```

IAM Identity Center 使用代码将 IAM Identity Center 会话与当前 AWS CLI 会话关联。IAM Identity Center 浏览器页面会提示您使用 IAM Identity Center 凭证登录。这授予了检索和显示您有权在 IAM Identity Center 中使用的 AWS 账户和角色的权限。AWS CLI

#### Note

登录过程可能会提示您允许 AWS CLI 访问您的数据。由于 AWS CLI 是在适用于 Python 的 SDK 之上构建的，因此权限消息可能包含 botocore 名称的变体。

### 4. AWS CLI 显示可供您使用的 AWS 帐户。如果您只能使用一个帐户，则会自动为您 AWS CLI 选择该帐户并跳过提示。可供您使用的 AWS 帐户由您在 IAM Identity Center 中的用户配置决定。

```
There are 2 AWS accounts available to you.
```

```
> DeveloperAccount, developer-account-admin@example.com (123456789011)  
   ProductionAccount, production-account-admin@example.com (123456789022)
```

使用箭头键选择要使用的帐户。左侧的“>”字符指向当前选择。按 Enter 进行选择。

### 5. AWS CLI 确认您的帐户选择，并显示所选帐户中可供您使用的 IAM 角色。如果所选帐户仅列出一个角色，则会自动为您 AWS CLI 选择该角色并跳过提示。可供您使用的角色由 IAM Identity Center 中的用户配置决定。

```
Using the account ID 123456789011
```



```
There are 2 roles available to you.  
> ReadOnly  
   FullAccess
```

使用箭头键选择要使用的 IAM 角色，然后按 <ENTER> 键。

- 指定默认输出格式、要向其发送命令的默认 AWS 区域，并提供配置文件名称，以便您可以从本地计算机上定义的所有配置文件中引用此配置文件。在以下示例中，用户输入默认区域、默认输出格式和配置文件的名称。如果您具有以前的现有配置，也可以按下 <ENTER> 以选择方括号之间显示的任何默认值。建议的配置文件名称是账户 ID 号，并依次后跟下划线和角色名称。

```
CLI default client Region [None]: us-west-2<ENTER>  
CLI default output format [None]: json<ENTER>  
CLI profile name [123456789011_ReadOnly]: my-dev-profile<ENTER>
```

#### Note

如果您指定 default 为配置文件名称，则当您运行 AWS CLI 命令而不指定配置文件名称时，该配置文件将成为使用的配置文件。

- 最后一条消息描述了完成的配置文件配置。

To use this profile, specify the profile name using `--profile`, as shown:

```
aws s3 ls --profile my-dev-profile
```

- 这会导致在 `~/.aws/config` 中创建 `sso-session` 部分和命名配置文件，具体如下所示：

```
[profile my-dev-profile]  
sso_session = my-sso  
sso_account_id = 123456789011  
sso_role_name = readOnly  
region = us-west-2  
output = json  
  
[sso-session my-sso]  
sso_region = us-east-1  
sso_start_url = https://my-sso-portal.awsapps.com/start  
sso_registration_scopes = sso:account:access
```



现在，您可以使用此 `sso-session` 和配置文件来请求刷新的凭证。使用 `aws sso login` 命令请求和检索运行命令所需的凭证。有关说明，请参阅[使用 IAM Identity Center 命名配置文件](#)。

## 使用 `aws configure sso-session` 向导仅配置您的 `sso-session` 部分

`aws configure sso-session` 命令仅更新 `~/.aws/config` 文件中的 `sso-session` 部分。此命令可用于创建或更新您的会话。如果您已有现有的配置设置并且想要创建新的 `sso-session` 配置或编辑现有的这一配置，这很有用。

运行 `aws configure sso-session` 命令并提供您的 IAM Identity Center 起始 URL 和托管身份中心目录的 AWS 区域。

```
$ aws configure sso-session
SSO session name: my-sso
SSO start URL [None]: https://my-sso-portal.awsapps.com/start
SSO region [None]: us-east-1
SSO registration scopes [None]: sso:account:access
```

输入您的信息后，会有一条消息描述已完成的配置文件配置。

```
Completed configuring SSO session: my-sso
Run the following to login and refresh access token for this session:

aws sso login --sso-session my-sso
```

### Note

如果您已登录到您正在更新的 `sso-session`，请运行 `aws sso login` 命令来刷新您的令牌。

## 使用 `config` 文件手动配置

该 `config` 文件的 `sso-session` 部分用于对用于获取 SSO 访问令牌的配置变量进行分组，然后可以使用这些变量来获取 AWS 凭证。使用了以下设置：

- (必需) [sso\\_start\\_url](#)
- (必需) [sso\\_region](#)

- [sso\\_account\\_id](#)
- [sso\\_role\\_name](#)
- [sso\\_registration\\_scopes](#)

您定义一个 `sso-session` 部分并将其关联到配置文件。必须在 `sso-session` 部分内设置 `sso_region` 和 `sso_start_url`。通常，`sso_account_id` 和 `sso_role_name` 必须在 `profile` 部分进行设置，以便 SDK 可以请求 SSO 凭证。

以下示例将 SDK 配置为请求 SSO 凭证并支持自动令牌刷新：

```
[profile dev]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://my-sso-portal.awsapps.com/start
```

这也允许在多个配置文件中重复使用 `sso-session` 配置：

```
[profile dev]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole

[profile prod]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole2

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://my-sso-portal.awsapps.com/start
```

但是，并不是所有 SSO 令牌配置场景都需要 `sso_account_id` 和 `sso_role_name`。如果您的应用程序仅使用支持所有者身份验证的 AWS 服务，则不需要传统 AWS 凭证。所有者身份验证是一种 HTTP 身份验证方案，它使用称为所有者令牌的安全令牌。在这种情况下，不需要 `sso_account_id` 和 `sso_role_name`。请参阅 AWS 服务的单独指南，以确定其是否支持所有者令牌授权。

此外，可以作为 `sso-session` 的一部分配置注册范围。范围是 OAuth 2.0 中的一种机制，用于限制应用程序对用户账户的访问。应用程序可以请求一个或多个范围，向应用程序签发的访问令牌将仅限于授予的范围。这些范围定义了为已注册的 OIDC 客户端请求授权的权限和客户端检索的访问令牌。以下示例将 `sso_registration_scopes` 设置为提供已列出账户/角色的访问权限：

```
[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://my-sso-portal.awsapps.com/start
sso_registration_scopes = sso:account:access
```

身份验证令牌缓存到 `~/.aws/sso/cache` 目录下的磁盘上，文件名基于会话名称。

## AWS IAM Identity Center 的旧版不可刷新配置

本主题介绍如何配置以通过 AWS IAM Identity Center (IAM Identity Center) 对用户进行身份验证，以获取使用传统方法运行 AWS CLI 命令的证书。AWS CLI 使用遗留的不可刷新配置时，您需要手动刷新令牌，因为它会定期过期。

使用 IAM Identity Center 时，您可以登录到 Active Directory、内置 IAM Identity Center 目录或[已连接到 IAM Identity Center 的另一个 IdP](#)。您可以将这些证书映射到可以在其中运行 AWS CLI 命令的 AWS Identity and Access Management (IAM) 角色。

无论您使用哪个 IdP，IAM Identity Center 都会将这些区别抽象出来。例如，您可以连接 Microsoft Azure AD，如博客文章 [IAM Identity Center 的下一个演进](#) 中所述。

### Note

有关使用不使用账户 ID 和角色的不记名身份验证的信息，请参阅 Amazon CodeCatalyst 用户指南 [CodeCatalyst 中的设置 AWS CLI 与使用](#)。

您可以通过以下方式将一个或多个已 AWS CLI [命名的配置文件配置](#) 为使用旧版 IAM Identity Center 中的角色：

- [自动](#)，使用 `aws configure sso` 命令。
- [手动](#)，通过编辑存储命名配置文件的 `config` 文件。

## 先决条件

- 安装 AWS CLI。有关更多信息，请参阅 [the section called “安装/更新”](#)。
- 您必须先有权访问 IAM Identity Center 中的 SSO 身份验证。选择以下方法之一来访问您的 AWS 证书。

我尚未通过 IAM Identity Center 确立访问权限

请按照《AWS IAM Identity Center 用户指南》的[入门](#)中的说明操作。此过程激活 IAM Identity Center，创建管理用户并添加适当的最低权限集。

### Note

创建应用最低权限权限的权限集。除非您的雇主已为此目的创建自定义权限集，否则建议使用预定义的 PowerUserAccess 权限集。

退出门户并再次登录以查看您的 AWS 账户 和 Administrator 或选项 PowerUserAccess。在使用开发工具包时选择 PowerUserAccess。这也可以帮助您找到有关编程访问的详细信息。

我已经 AWS 可以通过雇主管理的联合身份提供商（例如 Azure AD 或 Okta）进行访问

AWS 通过您的身份提供商的门户网站登录。如果您的云管理员已授予您 PowerUserAccess（开发者）权限，则您 AWS 账户 会看到您有权访问的权限和权限集。在您的权限集名称旁边，可以看到有关使用该权限集手动或以编程方式访问账户的选项。

自定义实现可能会产生不同的体验，例如不同的权限集名称。如果您不确定要使用哪个权限集，请联系 IT 团队以寻求帮助。

我已经 AWS 可以通过雇主管理的 AWS 访问门户进行访问

AWS 通过 AWS 访问门户登录。如果您的云管理员已授予您 PowerUserAccess（开发者）权限，则您 AWS 账户 会看到您有权访问的权限和权限集。在您的权限集名称旁边，可以看到有关使用该权限集手动或以编程方式访问账户的选项。

我已经 AWS 可以通过雇主管理的联合自定义身份提供商进行访问

请联系您的 IT 团队以寻求帮助。

## 遗留配置的自动配置

### 为您配置 IAM 身份中心配置文件 AWS CLI

1. 运行 `aws configure sso` 命令并提供您的 IAM Identity Center 起始 URL 和托管身份中心目录的 AWS 区域。

```
$ aws configure sso
SSO session name (Recommended):
SSO start URL [None]: https://my-sso-portal.awsapps.com/start
SSO region [None]:us-east-1
```

2. AWS CLI 尝试打开您的默认浏览器并开始您的 IAM Identity Center 账户的登录过程。

```
SSO authorization page has automatically been opened in your default browser.
Follow the instructions in the browser to complete this authorization request.
```

如果 AWS CLI 无法打开浏览器，则会显示以下消息，其中包含有关如何手动启动登录过程的说明。

```
Using a browser, open the following URL:

https://device.sso.us-west-2.amazonaws.com/

and enter the following code:
QCFK-N451
```

IAM Identity Center 使用代码将 IAM Identity Center 会话与当前 AWS CLI 会话关联。IAM Identity Center 浏览器页面会提示您使用 IAM Identity Center 凭证登录。这授予了检索和显示您有权在 IAM Identity Center 中使用的 AWS 账户和角色的权限。AWS CLI

3. 接下来，AWS CLI 显示可供您使用的 AWS 帐户。如果您只能使用一个帐户，则会自动为您 AWS CLI 选择该帐户并跳过提示。可供您使用的 AWS 帐户由您在 IAM Identity Center 中的用户配置决定。

```
There are 2 AWS accounts available to you.
> DeveloperAccount, developer-account-admin@example.com (123456789011)
  ProductionAccount, production-account-admin@example.com (123456789022)
```

使用箭头键选择要用于此配置文件的账户。左侧的“>”字符指向当前选择。按 Enter 进行选择。

4. 接下来，AWS CLI 确认您的账户选择，并显示所选账户中可供您使用的 IAM 角色。如果所选账户仅列出一个角色，则会自动为您 AWS CLI 选择该角色并跳过提示。可供您使用的角色由 IAM Identity Center 中的用户配置决定。

```
Using the account ID 123456789011
There are 2 roles available to you.
> ReadOnly
  FullAccess
```

使用箭头键选择要与这个配置文件一起使用的 IAM 角色，然后按 <ENTER> 键。

5. AWS CLI 确认您的角色选择。

```
Using the role name "ReadOnly"
```

6. 通过指定默认输出格式、向其发送命令的默认 AWS 区域 格式以及提供配置文件的名称来完成配置文件的配置，以便您可以从本地计算机上定义的所有配置文件中引用此配置文件。在以下示例中，用户输入原定设置区域、原定设置输出格式和配置文件的名称。也可以按下 <ENTER> 以选择方括号之间显示的任何原定设置值。建议的配置文件名称是账户 ID 号，并依次后跟下划线和角色名称。

```
CLI default client Region [None]: us-west-2<ENTER>
CLI default output format [None]: json<ENTER>
CLI profile name [123456789011_ReadOnly]: my-dev-profile<ENTER>
```

#### Note

如果您指定 default 为配置文件名称，则当您运行 AWS CLI 命令而不指定配置文件名称时，该配置文件将成为使用的配置文件。

7. 最后一条消息描述了完成的配置文件配置。

要使用此配置文件，请使用 `--profile` 指定配置文件名称，如下所示：

```
aws s3 ls --profile my-dev-profile
```

8. 前面的示例条目将在 `~/.aws/config` 中生成一个命名的配置文件，类似于以下示例。

```
[profile my-dev-profile]
sso_start_url = https://my-sso-portal.awsapps.com/start
```

```
sso_region = us-east-1
sso_account_id = 123456789011
sso_role_name = readOnly
region = us-west-2
output = json
```

此时，您有一个可用于请求临时凭证的配置文件。您必须使用 `aws sso login` 命令实际请求和检索运行命令所需的临时凭证。有关说明，请参阅[使用 IAM Identity Center 命名配置文件](#)。

## 遗留配置的手动配置

使用遗留的不可刷新配置不支持自动令牌刷新。我们建议使用 SSO 令牌配置。

要手动向命名的配置文件添加 IAM Identity Center 支持，必须将以下密钥和值添加到文件 `~/.aws/config` (Linux 或 macOS) 或 `%USERPROFILE%/.aws/config` (Windows) 中的配置文件定义中。

- [sso\\_start\\_url](#)
- [sso\\_region](#)
- [sso\\_account\\_id](#)
- [sso\\_role\\_name](#)

您可以在 `.aws/config` 文件中包括有效的任何其他键和值，例如 [region](#)、[output](#) 或 [s3](#)。为防止出错，不要包括任何与凭证相关的值，例如 [role\\_arn](#) 或 [aws\\_secret\\_access\\_key](#)。

以下是 `.aws/config` 中的示例 IAM Identity Center 配置文件：

```
[profile my-sso-profile]
sso_start_url = https://my-sso-portal.awsapps.com/start
sso_region = us-west-2
sso_account_id = 111122223333
sso_role_name = SSOReadOnlyRole
region = us-west-2
output = json
```

您的临时凭证配置文件已完成。

要运行命令，您必须首先使用 `aws sso login` 命令请求和检索临时凭证。有关说明，请参阅下一节：[使用 IAM Identity Center 命名配置文件](#)。身份验证令牌缓存到磁盘中的 `~/.aws/sso/cache` 目录下面，文件名基于 `sso_start_url`。

## 使用 IAM Identity Center 命名配置文件

本主题介绍如何使用 AWS CLI 通过 AWS IAM Identity Center ( IAM Identity Center ) 对用户进行身份验证，以获取用于运行 AWS CLI 命令的凭证。

### Note

您的凭证是临时还是自动刷新，这取决于您之前如何对配置文件进行配置。

### 主题

- [先决条件](#)
- [登录和获取凭证](#)
- [使用 IAM Identity Center 配置文件运行命令](#)
- [退出您的 IAM Identity Center 会话](#)

### 先决条件

您配置了 IAM Identity Center 配置文件。有关更多信息，请参阅 [the section called “配置自动令牌刷新”](#) 和 [the section called “配置旧版不可刷新”](#)。

### 登录和获取凭证

### Note

登录过程可能会提示您允许 AWS CLI 访问您的数据。由于 AWS CLI 基于 SDK for Python 而构建，因此权限消息可能包含 `botocore` 名称的变体。

配置命名配置文件后，您可以调用它以从 AWS 请求凭证。您必须检索和缓存一组凭证，然后才能运行 AWS CLI 服务命令。要获取这些凭证，请运行以下命令。

```
$ aws sso login --profile my-dev-profile
```

AWS CLI 打开原定设置浏览器并验证您的 IAM Identity Center 登录。

```
SSO authorization page has automatically been opened in your default browser.
```



```
Follow the instructions in the browser to complete this authorization request.  
Successfully logged into Start URL: https://my-sso-portal.awsapps.com/start
```

如果您当前未登录 IAM Identity Center，则必须提供您的 IAM Identity Center 凭证。

如果 AWS CLI 无法打开浏览器，则系统会提示您自行打开浏览器并输入指定的代码。

```
$ aws sso login --profile my-dev-profile  
Using a browser, open the following URL:  
  
https://device.sso.us-west-2.amazonaws.com/  
  
and enter the following code:  
QCFK-N451
```

AWS CLI 将您的原定设置浏览器打开（或您手动将选择的浏览器打开）到指定页面，然后输入提供的代码。然后，网页会提示您输入 IAM Identity Center 凭证。

缓存 IAM Identity Center 会话凭证。如果这些凭证是临时性的，则会包括过期时间戳，并且在凭证过期后，AWS CLI 会请求您再次登录 IAM Identity Center。

如果您的 IAM Identity Center 凭证有效，则 AWS CLI 使用它们安全地检索在配置文件中指定的用于 IAM 角色的 AWS 凭证。

```
Welcome, you have successfully signed-in to the AWS-CLI.
```

在使用 `aws sso login` 命令的 `--sso-session` 参数登录时，您还可以指定使用哪个 `sso-session` 配置文件。

```
$ aws sso login --sso-session my-dev-session  
Attempting to automatically open the SSO authorization page in your default browser.  
If the browser does not open or you wish to use a different device to authorize this  
request, open the following URL:  
  
https://device.sso.us-west-2.amazonaws.com/  
  
and enter the following code:  
QCFK-N451  
  
Successfully logged into Start URL: https://cli-reinvent.awsapps.com/start
```

## 使用 IAM Identity Center 配置文件运行命令

您可以使用这些凭证通过关联的命名配置文件来调用 AWS CLI 命令。以下示例显示该命令是在作为指定账户一部分的代入角色下运行的。

```
$ aws sts get-caller-identity --profile my-dev-profile
{
  "UserId": "AROA12345678901234567:test-user@example.com",
  "Account": "123456789011",
  "Arn": "arn:aws:sts::123456789011:assumed-role/
AWSPeregrine_readOnly_12321abc454d123/test-user@example.com"
}
```

只要您登录 IAM Identity Center 并且这些缓存的凭证未过期，AWS CLI 就会在需要时自动续订过期的 AWS 凭证。但是，如果您的 IAM Identity Center 凭证过期，则必须通过再次登录您的 IAM Identity Center 账户来显式续订它们。

```
$ aws s3 ls --profile my-sso-profile
Your short-term credentials have expired. Please sign-in to renew your credentials
SSO authorization page has automatically been opened in your default browser.
Follow the instructions in the browser to complete this authorization request.
```

## 退出您的 IAM Identity Center 会话

完成使用 IAM Identity Center 命名配置文件后，您可以选择不执行任何操作，而让 AWS 临时凭证和您的 IAM Identity Center 凭证过期。但是，您也可以选择运行以下命令来立即删除 SSO 凭证缓存文件夹中的所有缓存凭证和基于 IAM Identity Center 凭证的所有 AWS 临时凭证。这使得这些凭证不可用于将来的任何命令。

```
$ aws sso logout
Successfully signed out of all SSO profiles.
```

如果稍后要使用 IAM Identity Center 命名配置文件之一来运行命令，则必须再次运行 `aws sso login` 命令（请参阅上一节）并指定要使用的配置文件。

## 使用短期凭证进行身份验证

我们建议将您的开发工具包或工具配置为使用带延长会话持续时间选项的 [IAM Identity Center 身份验证](#)。但是，您可以复制和使用 AWS 访问门户中提供的临时凭证。在这些临时凭证过期后，将需要复制新凭证。您可以在配置文件中使临时凭证，也可以将其用作系统属性和环境变量的值。

1. [登录到 AWS 访问门户](#)。
2. 按照[以下说明](#)操作，从 AWS 访问门户中复制 IAM 角色凭证。
  1. 对于链接的说明中的步骤 2，选择可授予访问权限以满足您的开发需求的 AWS 账户和 IAM 角色名称。此角色的名称通常类似于 PowerUserAccess 或 Developer。
  2. 对于步骤 4，选择将配置文件添加到您的 AWS 凭证文件选项并复制内容。
3. 创建或打开共享 credentials 文件。此文件在 Linux 和 macOS 系统上为 ~/.aws/credentials，在 Windows 上为 %USERPROFILE%\aws\credentials。有关更多信息，请参阅[the section called “配置和凭证文件设置”](#)。
4. 将以下文本添加到共享 credentials 文件中。将示例值替换为您复制的凭证。

```
[default]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token =
IQoJb3JpZ2luX2l0oJb3JpZ2luX2l0oJb3JpZ2luX2l0oJb3JpZ2luX2l0oJb3JpZ2l0ZVVERYLONGSTRINGEXAMPLE
```

5. 将您的首选默认区域和格式添加到共享 config 文件中。

```
[default]
region=us-west-2
output=json

[profile user1]
region=us-east-1
output=text
```

当 SDK 创建服务客户端时，它将访问这些临时凭证并将它们用于每个请求。在步骤 2a 中选择的 IAM 角色的设置决定了[临时凭证的有效时间](#)。最长持续时间为 12 小时。

每次您的凭证到期时都重复这些步骤。

## 在 AWS CLI 中使用 IAM 角色

[AWS Identity and Access Management \(IAM\) 角色](#)是一种授权工具，可让用户获得额外（或不同）的权限或者获取使用其他 AWS 账户执行操作的权限。

主题

- [先决条件](#)
- [IAM 角色的用法概览](#)
- [配置和使用角色](#)
- [使用多重验证](#)
- [跨账户角色和外部 ID](#)
- [指定角色会话名称以便于审核](#)
- [通过 Web 身份代入角色](#)
- [清除缓存的凭证](#)

## 先决条件

要运行 `iam` 命令，您需要安装和配置 AWS CLI。有关更多信息，请参阅[the section called “安装/更新”](#)。

## IAM 角色的用法概览

通过在 `~/.aws/config` 文件中为某个 IAM 角色定义配置文件，您可以配置 AWS Command Line Interface (AWS CLI) 以使用该 IAM 角色。

以下示例显示了一个名为 `marketingadmin` 的角色配置文件。如果使用 `--profile marketingadmin` (或使用 [AWS\\_PROFILE 环境变量](#) 指定它) 运行命令，则 AWS CLI 使用在单独的配置文件 `user1` 中定义的凭证代入 Amazon Resource Name (ARN) 为 `arn:aws:iam::123456789012:role/marketingadminrole` 的角色。您可以运行分配给该角色的权限所允许的任何操作。

```
[profile marketingadmin]  
role_arn = arn:aws:iam::123456789012:role/marketingadminrole  
source_profile = user1
```

然后，您可以指定一个指向单独的命名配置文件的 `source_profile`，此配置文件包含用户凭证及使用该角色的权限。在上一个示例中，`marketingadmin` 配置文件使用 `user1` 配置文件中的凭证。当您指定某个 AWS CLI 命令将使用配置文件 `marketingadmin` 时，AWS CLI 会自动查找链接的 `user1` 配置文件的凭证，并使用它们为指定的 IAM 角色请求临时凭证。CLI 在后台使用 [sts:AssumeRole](#) 操作来完成该操作。然后，使用这些临时凭证来运行请求的 AWS CLI 命令。指定的角色必须附加有允许运行请求的 AWS CLI 命令的 IAM 权限策略。

要从 Amazon Elastic Compute Cloud (Amazon EC2) 实例或 Amazon Elastic Container Service (Amazon ECS) 容器中运行 AWS CLI 命令，您可以使用附加到实例配置文件或容器的 IAM 角色。如果未指定配置文件或未设置环境变量，则将直接使用角色。这让您能够避免在实例上存储长时间生存的访问密钥。您也可以使用这些实例或容器角色仅获取其他角色的凭证。为此，请使用 `credential_source`（而不是 `source_profile`）指定如何查找凭证。`credential_source` 属性支持以下值：

- `Environment` – 从环境变量检索源凭证。
- `Ec2InstanceMetadata` – 使用附加到 Amazon EC2 实例配置文件的 IAM 角色。
- `EcsContainer` – 使用附加到 Amazon ECS 容器的 IAM 角色。

以下示例显示通过引用 Amazon EC2 实例配置文件来使用同一个 `marketingadminrole` 角色。

```
[profile marketingadmin]
role_arn = arn:aws:iam::123456789012:role/marketingadminrole
credential_source = Ec2InstanceMetadata
```

当您调用角色时，您可以要求使用其他选项，例如，使用多重身份验证和外部 ID（供第三方公司用于访问其客户的资源）。也可以指定更易于在 AWS CloudTrail 日志中进行审核的唯一角色会话名称。

## 配置和使用角色

在使用指定 IAM 角色的配置文件运行命令时，AWS CLI 将使用源配置文件的凭证调用 AWS Security Token Service (AWS STS) 并为指定角色请求临时凭证。源配置文件中的用户必须具有为指定配置文件中的角色调用 `sts:assume-role` 的权限。该角色必须具有允许源配置文件中的用户使用角色的信任关系。检索角色的临时凭证然后使用临时凭证的过程通常称为代入角色。

您可以通过执行《AWS Identity and Access Management 用户指南》中的[创建向 IAM 用户委派权限的角色](#)下的过程，在 IAM 中创建一个您希望用户代入的具有该权限的角色。如果该角色和源配置文件的用户在同一个账户中，在配置角色的信任关系时，您可以输入自己的账户 ID。

在创建角色后，请修改信任关系以允许用户担任该角色。

以下示例显示了一个可附加到角色的信任策略。该策略允许账户 123456789012 中的任何用户代入该角色，前提是该账户的管理员向该用户显式授予了 `sts:AssumeRole` 权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456789012:root"
  },
  "Action": "sts:AssumeRole"
}
```

信任策略不会实际授予权限。账户管理员必须通过附加具有适当权限的策略才能将代入角色的权限委派给各个用户。以下示例显示了一个可附加到用户的策略，该策略仅允许用户代入 `marketingadminrole` 角色。有关授予用户代入角色的访问权限的更多信息，请参阅 IAM 用户指南中的[向用户授予切换角色的权限](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::123456789012:role/marketingadminrole"
    }
  ]
}
```

用户无需拥有附加权限，即可使用角色配置文件运行 AWS CLI 命令。相反，运行命令的权限来自附加到角色的权限。您可以将权限策略附加到角色，以指定可以针对哪些 AWS 资源执行哪些操作。有关向角色附加权限（与向用户附加权限的操作相同）的更多信息，请参阅《IAM 用户指南》中的[更改 IAM 用户的权限](#)。

既然您已正确地配置角色配置文件、角色权限、角色信任关系和用户权限，那么就可以通过调用 `--profile` 选项在命令中使用该角色了。例如，下面的内容使用附加到 `ls` 角色（由本主题开头的示例定义）的权限调用 Amazon S3 `marketingadmin` 命令。

```
$ aws s3 ls --profile marketingadmin
```

要对多个调用使用角色，您可以从命令行设置当前会话的 `AWS_PROFILE` 环境变量。定义该环境变量后，就不必对每个命令都指定 `--profile` 选项。

Linux 或 macOS

```
$ export AWS_PROFILE=marketingadmin
```

## Windows

```
C:\> setx AWS_PROFILE marketingadmin
```

有关配置用户和角色的更多信息，请参阅《IAM 用户指南》中的[用户和组](#)和[角色](#)。

## 使用多重验证

为了提高安全性，当用户尝试使用角色配置文件进行调用时，您可以要求用户提供从多重验证 (MFA) 设备 (一种 U2F 设备) 或移动应用程序生成的一次性密钥。

首先，您可以选择将与 IAM 角色有关的信任关系修改为需要 MFA。这可以防止任何人在未首先使用 MFA 进行身份验证的情况下使用该角色。有关示例，请参阅下面示例中的 Condition 行。此策略允许名为 anika 的用户代入策略所附加的角色，但前提是用户使用 MFA 进行身份验证。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::123456789012:user/anika" },
      "Action": "sts:AssumeRole",
      "Condition": { "Bool": { "aws:multifactorAuthPresent": true } }
    }
  ]
}
```

其次，为角色配置文件添加一行，用来指定用户的 MFA 设备的 ARN。以下示例 config 文件条目显示两个角色配置文件，它们都使用用户 anika 的访问密钥来请求角色 cli-role 的临时凭证。用户 anika 有权代入角色，这是由角色的信任策略授予的。

```
[profile role-without-mfa]
region = us-west-2
role_arn= arn:aws:iam::128716708097:role/cli-role
source_profile=cli-user

[profile role-with-mfa]
```

```
region = us-west-2
role_arn= arn:aws:iam::128716708097:role/cli-role
source_profile = cli-user
mfa_serial = arn:aws:iam::128716708097:mfa/cli-user

[profile cli-user]
region = us-west-2
output = json
```

该 `mfa_serial` 设置可以采取如图所示的 ARN 或硬件 MFA 令牌的序列号。

第一个配置文件 `role-without-mfa` 不需要 MFA。但是，由于附加到角色的先前示例信任策略需要 MFA，因此使用此配置文件运行命令的任何尝试都将失败。

```
$ aws iam list-users --profile role-without-mfa
```

```
An error occurred (AccessDenied) when calling the AssumeRole operation: Access denied
```

第二个配置文件条目 `role-with-mfa` 标识要使用的 MFA 设备。当此用户尝试使用此配置文件运行 AWS CLI 命令时，AWS CLI 会提示用户输入 MFA 设备提供的一次性密码 (OTP)。如果 MFA 身份验证成功，此命令将执行所请求的操作。OTP 未显示在屏幕上。

```
$ aws iam list-users --profile role-with-mfa
Enter MFA code for arn:aws:iam::123456789012:mfa/cli-user:
{
  "Users": [
    {
      ...
    }
  ]
}
```

## 跨账户角色和外部 ID

通过将角色配置为跨账户角色，您可以让 用户使用属于不同账户的角色。在创建角色期间，将角色类型设置为其他AWS账户（[如创建向 IAM 用户委派权限的角色中所述](#)）。（可选）选择 Require MFA（需要 MFA）。Require MFA（需要 MFA）将按照 [使用多重验证](#) 中所述在信任关系中配置相应条件。

如果使用[外部 ID](#) 来加强控制可跨账户使用角色的人员，则还必须将 `external_id` 参数添加到角色配置文件。通常情况下，仅应在其他账户由公司或组织外部的人员控制时才使用该功能。

```
[profile crossaccountrole]
role_arn = arn:aws:iam::234567890123:role/SomeRole
```



```
source_profile = default
mfa_serial = arn:aws:iam::123456789012:mfa/saanvi
external_id = 123456
```

## 指定角色会话名称以便于审核

当许多人共享一个角色时，审核会变得比较困难。您希望将调用的每个操作与调用该操作的个人关联。但是，当个人使用角色时，个人代入角色是一项独立于调用操作的行为，您必须手动将这两者关联起来。

通过在用户代入角色时指定唯一的角色会话名称，您可以简化此过程。只需向指定某一角色的 `role_session_name` 文件中的每个命名配置文件添加 `config` 参数，即可实现这一点。`role_session_name` 值将传递给 `AssumeRole` 操作，并成为角色会话 ARN 的一部分。该值也包含在所有已记录操作的 AWS CloudTrail 日志中。

例如，您可以创建基于角色的配置文件，如下所示。

```
[profile namedsessionrole]
role_arn = arn:aws:iam::234567890123:role/SomeRole
source_profile = default
role_session_name = Session_Maria_Garcia
```

这会导致角色会话具有以下 ARN。

```
arn:aws:iam::234567890123:assumed-role/SomeRole/Session_Maria_Garcia
```

此外，所有 AWS CloudTrail 日志都在为每个操作捕获的信息中包含角色会话名称。

## 通过 Web 身份代入角色

您可以配置一个配置文件，以指示 AWS CLI 应使用 [Web 联合身份验证和 Open ID Connect \(OIDC\)](#) 代入角色。当您在配置文件中指定此选项时，AWS CLI 会自动为您发出相应的 AWS STS `AssumeRoleWithWebIdentity` 调用。

### Note

当您指定使用 IAM 角色的配置文件时，AWS CLI 会发出相应的调用来检索临时凭证。这些凭证存储在 `~/.aws/cli/cache` 中。指定同一个配置文件的后续 AWS CLI 命令将使用缓存的临时凭证，直到它们过期。这时，AWS CLI 将自动刷新这些凭证。

要通过 Web 联合身份验证检索和使用临时凭证，您可以在共享配置文件中指定以下配置值。

### [role\\_arn](#)

指定要代入的角色的 ARN。

### [web\\_identity\\_token\\_file](#)

指定一个文件的路径，该文件包含由身份提供商提供的 OAuth 2.0 访问令牌或 OpenID Connect ID 令牌。AWS CLI 加载此文件，并将其内容作为 WebIdentityToken 操作的 AssumeRoleWithWebIdentity 参数传递。

### [role\\_session\\_name](#)

指定应用于此代入角色会话的可选名称。

以下是使用 Web 身份配置文件配置代入角色所需的最少量配置的示例。

```
# In ~/.aws/config

[profile web-identity]
role_arn=arn:aws:iam:123456789012:role/RoLeNameToAssume
web_identity_token_file=/path/to/a/token
```

您也可以使用[环境变量](#)提供此配置。

### [AWS\\_ROLE\\_ARN](#)

要代入的角色的 ARN。

### [AWS\\_WEB\\_IDENTITY\\_TOKEN\\_FILE](#)

Web 身份令牌文件的路径。

### [AWS\\_ROLE\\_SESSION\\_NAME](#)

应用于此代入角色会话的名称。

#### Note

这些环境变量当前仅适用于通过 Web 身份提供程序代入角色。它们不适用于常规的代入角色提供程序配置。

## 清除缓存的凭证

当您使用一个角色时，AWS CLI 会在本地缓存临时凭证，直到这些凭证过期。当您下次尝试使用它们时，AWS CLI 会尝试代表您续订这些凭证。

如果您的角色的临时凭证已吊销，它们不会自动续订，并且使用它们的尝试将失败。但是，您可以删除缓存以强制 AWS CLI 检索新凭证。

Linux 或 macOS

```
$ rm -r ~/.aws/cli/cache
```

Windows

```
C:\> del /s /q %UserProfile%\aws\cli\cache
```

## 使用 IAM 用户凭证进行身份验证

### Warning

为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证，而是使用与身份提供商的联合身份验证，例如 [AWS IAM Identity Center](#)。

此部分介绍如何使用 IAM 用户配置基本设置。其中包括使用 config 和 credentials 文件的安全凭证。如需改为查看 AWS IAM Identity Center 的配置说明，请参阅 [the section called “IAM Identity Center 身份验证”](#)。

主题

- [步骤 1：创建您的 IAM 用户](#)
- [步骤 2：获取您的访问密钥](#)
- [配置 AWS CLI](#)
  - [使用 aws configure](#)
  - [通过 .CSV 文件导入访问密钥](#)
  - [直接编辑 config 和 credentials 文件](#)

## 步骤 1：创建您的 IAM 用户

按照《IAM 用户指南》中的[创建 IAM 用户（控制台）](#)过程操作来创建 IAM 用户。

- 对于权限选项，选择直接附加策略以了解如何向该用户分配权限。
- 大多数“入门”开发工具包教程都使用 Amazon S3 服务作为示例。要向应用程序提供对 Amazon S3 的完全访问权限，请选择要附加到此用户的 AmazonS3FullAccess 策略。

## 步骤 2：获取您的访问密钥

1. 登录 AWS Management Console 并打开 IAM 控制台，[网址为 https://console.aws.amazon.com/iam/](https://console.aws.amazon.com/iam/)。
2. 在 IAM 控制台的导航窗格中，选择用户，然后选择您之前创建用户的 **User name**。
3. 在用户的页面上，选择安全凭证页面。然后，在访问密钥下，选择创建访问密钥。
4. 对于创建访问密钥步骤 1，选择命令行界面 (CLI)。
5. 对于创建访问密钥步骤 2，输入可选标记并选择下一步。
6. 对于创建访问密钥步骤 3，选择下载.csv 文件以保存包含您的 IAM 用户访问密钥和秘密访问密钥的 .csv 文件。稍后您将需要此信息。
7. 选择 Done (完成)。

## 配置 AWS CLI

一般而言，AWS CLI 需要以下信息：

- 访问密钥 ID
- 秘密访问密钥
- AWS 区域
- 输出格式

将此信息 AWS CLI 存储在文件中命名的配置credentials文件（设置集合）default中。默认情况下，当您运行未明确指定要使用的配置文件的 AWS CLI 命令时，将使用此配置文件中的信息。有关credentials文件的更多信息，请参阅[配置和凭证文件设置](#)。

要配置 AWS CLI，请使用以下过程之一：

## 主题

- [使用 aws configure](#)
- [通过 .CSV 文件导入访问密钥](#)
- [直接编辑 config 和 credentials 文件](#)

## 使用 aws configure

一般而言，该 `aws configure` 命令是设置 AWS CLI 安装的最快方法。此配置向导将提示您输入入门所需的每条信息。除非使用 `--profile` 选项另行指定，否则会将此信息 AWS CLI 存储在 `default` 配置文件中。

以下示例使用示例值配置 `default` 配置文件。将它们替换为您自己的值，如以下部分所述。

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

以下示例使用示例值配置名为 `userprod` 的配置文件。将它们替换为您自己的值，如以下部分所述。

```
$ aws configure --profile userprod
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

## 通过 .CSV 文件导入访问密钥

您可以导入创建访问密钥后下载的纯文本 `.csv` 文件，而不必使用 `aws configure` 输入访问密钥。

`.csv` 文件必须包含以下标头。

- 用户名 - 必须将此列添加到您的 `.csv`。这用于创建导入时在 `config` 和 `credentials` 文件中使用的配置文件名称。
- 访问密钥 ID
- 秘密访问密钥

**Note**

在创建初始访问密钥的过程中，一旦您关闭下载 .csv 文件对话框，就无法访问秘密访问密钥。如果您需要 .csv 文件，则需要自己创建一个包含所需标头和存储的访问密钥信息的文件。如果您不具有对访问密钥信息的访问权限，则需要创建新的访问密钥。

要导入 .csv 文件，请将 `aws configure import` 命令与 `--csv` 选项结合使用，如下所示：

```
$ aws configure import --csv file://credentials.csv
```

有关更多信息，请参阅 [aws\\_configure\\_import](#)。

## 直接编辑 `config` 和 `credentials` 文件

要直接编辑 `config` 和 `credentials` 文件，请执行以下操作。

1. 创建或打开共享 AWS `credentials` 文件。此文件在 Linux 和 macOS 系统上为 `~/.aws/credentials`，在 Windows 上为 `%USERPROFILE%\aws\credentials`。有关更多信息，请参阅 [the section called “配置和凭证文件设置”](#)。
2. 将以下文本添加到共享 `credentials` 文件中。替换您之前下载的 .csv 文件中的示例值并保存该文件。

```
[default]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

## 将凭证用于 Amazon EC2 实例元数据

当您在亚马逊弹性计算云 (Amazon EC2) 实例中运行时，可以简化为命令提供凭证的过程。AWS CLI 每个 Amazon EC2 实例都包含 AWS CLI 能够直接查询临时凭证的元数据。将 IAM 角色附加到实例后，AWS CLI 会自动安全地从实例元数据中检索证书。

要禁用此服务，请使用 [AWS\\_EC2\\_METADATA\\_DISABLED](#) 环境变量。

### 主题

- [先决条件](#)

- [配置 Amazon EC2 元数据的配置文件](#)

## 先决条件

要将 Amazon EC2 凭证与一起使用 AWS CLI，您需要完成以下操作：

- 安装和配置 AWS CLI。有关更多信息，请参阅 [the section called “安装/更新”](#) 和 [身份验证和访问凭证](#)。
- 您了解配置文件和命名配置文件。有关更多信息，请参阅 [配置和凭证文件设置](#)。
- 您已经创建了一个可以访问所需资源的 AWS Identity and Access Management (IAM) 角色，并在启动 Amazon EC2 实例时将该角色附加到 Amazon EC2 实例。有关更多信息，请参阅 [Amazon EC2 用户指南中的 Amazon EC2 的 IAM 策略](#) 和 IAM 用户指南中的 [授予在亚马逊 EC2 实例上运行的应用程序访问 AWS 资源的权限](#)。

## 配置 Amazon EC2 元数据的配置文件

要指定需要使用在托管 Amazon EC2 实例配置文件中提供的凭证，请在配置文件的命名配置文件中使以下语法。有关更多说明，请参阅以下步骤。

```
[profile profilename]  
role_arn = arn:aws:iam::123456789012:role/rolename  
credential_source = Ec2InstanceMetadata  
region = region
```

1. 在配置文件中创建配置文件。

```
[profile profilename]
```

2. 添加有权访问所需资源的 IAM arn 角色。

```
role_arn = arn:aws:iam::123456789012:role/rolename
```

3. 指定 Ec2InstanceMetadata 作为凭证源。

```
credential_source = Ec2InstanceMetadata
```

4. 设置您的区域。

```
region = region
```

## 示例

以下示例代入 *marketingadminrole* 角色并在名为 *marketingadmin* 的 Amazon EC2 实例配置文件中 使用 *us-west-2* 区域。

```
[profile marketingadmin]  
role_arn = arn:aws:iam::123456789012:role/marketingadminrole  
credential_source = Ec2InstanceMetadata  
region = us-west-2
```

## 使用外部进程获取凭证

### Warning

本主题讨论从外部进程获取凭证。如果生成凭证的命令可由未经批准的进程或用户访问，则可能存在安全风险。我们建议您使用 AWS CLI 和 AWS 提供的支持的安全替代方案，以降低泄露凭证的风险。请务必保管好 config 文件及任何支持文件和工具，以防泄露。

确保您的自定义凭证工具不会将任何秘密信息写入 StdErr，因为开发工具包和 AWS CLI 可以捕获和记录此类信息，可能会将其向未经授权的用户公开。

如果您有 AWS CLI 不直接支持的生成或查找凭证的方法，则可以通过在 config 文件中配置 `credential_process` 设置来配置 AWS CLI 使用它。

例如，您可以在 config 文件中包含类似于以下内容的条目。

```
[profile developer]  
credential_process = /opt/bin/awscreds-custom --username helen
```

## 语法

要以与任何操作系统兼容的方式创建此字符串，请遵循以下规则：

- 如果路径或文件名包含空格，请将完整路径和文件名用双引号 (" ") 括起来。该路径和文件名仅包含以下字符：A-Z a-z 0-9 - \_ . 空格



- 如果参数名称或参数值包含空格，则用双引号 (" ") 将该元素括起来。仅括起来名称或值，而不是名称值对。
- 请勿在字符串中包含任何环境变量。例如，您不能包含 \$HOME 或 %USERPROFILE%。
- 不要将主文件夹指定为 ~。您必须指定完整路径。

## Windows 示例

```
credential_process = "C:\Path\To\credentials.cmd" parameterWithoutSpaces "parameter with spaces"
```

## Linux 或 macOS 示例

```
credential_process = "/Users/Dave/path/to/credentials.sh" parameterWithoutSpaces "parameter with spaces"
```

## 凭证计划的预期输出

AWS CLI 按照配置文件中指定的方式运行该命令，然后从 STDOUT 读取数据。您指定的命令必须在 STDOUT 上生成符合以下语法的 JSON 输出。

```
{
  "Version": 1,
  "AccessKeyId": "an AWS access key",
  "SecretAccessKey": "your AWS secret access key",
  "SessionToken": "the AWS session token for temporary credentials",
  "Expiration": "ISO8601 timestamp when the credentials expire"
}
```

### Note

截至撰写本文之时，Version 密钥必须设置为 1。随时间推移和该结构的发展，该值可能会增加。

Expiration 密钥是采用 [ISO8601](#) 格式的时间戳。如果工具的输出中不存在 Expiration 键，则 CLI 假定凭证是不刷新的长期凭证。否则，将其视为临时凭证，并通过在其过期前重新运行 credential\_process 命令来自动刷新凭证。

**Note**

AWS CLI 不缓存外部进程凭据，这一点不同于代入角色凭证。如果需要缓存，则必须在外部进程中实现。

外部进程可以返回非零返回代码，以指示在检索凭证时发生错误。

# 使用 AWS CLI

除“配置 [the section called “端点”](#)”部分的内容外，本部分还提供有关 AWS Command Line Interface (AWS CLI) 的一般用法、常用功能和可用选项的信息。说明包括如何编写命令、基本结构、格式化、筛选以及查找命令的帮助内容或文档。

有关 AWS 服务的具体示例，请参阅[代码示例](#)或 [AWS CLI 版本 2 参考指南](#)。

## Note

默认情况下，AWS CLI 通过在 TCP 端口 443 上使用 HTTPS，将请求发送到 AWS 服务。要成功使用 AWS CLI，您必须能够在 TCP 端口 443 上建立出站连接。

本指南中的主题

- [获取有关 AWS CLI 的帮助](#)
- [AWS CLI 中的命令结构](#)
- [为 AWS CLI 指定参数值](#)
- [让 AWS CLI 提示您输入命令](#)
- [控制来自的命令输出 AWS CLI](#)
- [从 AWS CLI 返回代码](#)
- [使用 AWS CLI 向导的交互式命令](#)
- [创建和使用名为别名的 AWS CLI 命令快捷方式](#)

## 获取有关 AWS CLI 的帮助

本主题介绍如何访问 AWS Command Line Interface (AWS CLI) 的帮助内容。

主题

- [内置 AWS CLI 帮助命令](#)
- [AWS CLI 参考指南](#)
- [API 文档](#)
- [纠正错误](#)

- [其他帮助](#)

## 内置 AWS CLI 帮助命令

使用 AWS Command Line Interface (AWS CLI) 时，您可以获得任何命令的帮助。为此，只需在命令名称末尾键入 help。

例如，以下命令显示常规 AWS CLI 选项和可用顶层命令的帮助。

```
$ aws help
```

以下命令显示可用的 Amazon Elastic Compute Cloud (Amazon EC2) 特定命令。

```
$ aws ec2 help
```

以下示例显示 Amazon EC2 DescribeInstances 操作的详细帮助。帮助包括对其输入参数、可用筛选条件以及作为输出包含的内容的描述。它还包含说明如何键入命令的常见变体的示例。

```
$ aws ec2 describe-instances help
```

每个命令的帮助分为六个部分：

### 名称

命令的名称。

```
NAME
    describe-instances -
```

### 说明

命令调用的 API 操作的描述。

```
DESCRIPTION
    Describes one or more of your instances.

    If you specify one or more instance IDs, Amazon EC2 returns information for those instances. If you do not specify instance IDs, Amazon EC2 returns information for all relevant instances. If you specify an instance ID that is not valid, an error is returned. If you specify an
```

```
instance that you do not own, it is not included in the returned
results.
```

```
...
```

## 摘要

使用命令及其选项的基本语法。如果某个选项显示在方括号中，则表示该选项是可选的、具有默认值或具有可使用的替代选项。

### SYNOPSIS

```
describe-instances
[--dry-run | --no-dry-run]
[--instance-ids <value>]
[--filters <value>]
[--cli-input-json <value>]
[--starting-token <value>]
[--page-size <value>]
[--max-items <value>]
[--generate-cli-skeleton]
```

例如，`describe-instances` 具有描述当前账户和 AWS 区域中的所有实例的默认行为。您可以选择指定 `instance-ids` 列表来描述一个或多个实例；`dry-run` 是不接受值的可选布尔标志。要使用布尔标志，请指定其中一个显示的值，在本例中为 `--dry-run` 或 `--no-dry-run`。同样，`--generate-cli-skeleton` 不使用值。如果某个选项的使用存在条件，则在 `OPTIONS` 部分中描述这些条件，或在示例中显示这些条件。

## 选项

对摘要中显示的每个选项的描述。

### OPTIONS

```
--dry-run | --no-dry-run (boolean)
Checks whether you have the required permissions for the action,
without actually making the request, and provides an error response.
If you have the required permissions, the error response is DryRun-
Operation . Otherwise, it is UnauthorizedOperation .
```

```
--instance-ids (list)
One or more instance IDs.

Default: Describes all your instances.
```

```
...
```

## 示例

一些示例，用于显示命令及其选项的使用。如果您需要的命令或用例没有示例可用，请使用本页面上的反馈链接请求一个示例，或在命令的帮助页上的 AWS CLI 命令参考中请求一个示例。

### EXAMPLES

#### To describe an Amazon EC2 instance

Command:

```
aws ec2 describe-instances --instance-ids i-5203422c
```

#### To describe all instances with the instance type m1.small

Command:

```
aws ec2 describe-instances --filters "Name=instance-type,Values=m1.small"
```

#### To describe all instances with an Owner tag

Command:

```
aws ec2 describe-instances --filters "Name=tag-key,Values=Owner"
```

...

## 输出

来自的响应中包含的每个字段和数据类型的描述AWS

对于 `describe-instances`，输出是预留对象的列表，每个列表都包含若干字段和对象，这些字段和对象包含与其关联的实例的相关信息。此信息来自 Amazon EC2 使用的[预留数据类型的 API 文档](#)。

### OUTPUT

```
Reservations -> (list)
  One or more reservations.

  (structure)
    Describes a reservation.

    ReservationId -> (string)
      The ID of the reservation.
```

```
OwnerId -> (string)
    The ID of the AWS account that owns the reservation.

RequesterId -> (string)
    The ID of the requester that launched the instances on your
    behalf (for example, AWS Management Console or Auto Scaling).

Groups -> (list)
    One or more security groups.

    (structure)
        Describes a security group.

        GroupName -> (string)
            The name of the security group.

        GroupId -> (string)
            The ID of the security group.

Instances -> (list)
    One or more instances.

    (structure)
        Describes an instance.

        InstanceId -> (string)
            The ID of the instance.

        ImageId -> (string)
            The ID of the AMI used to launch the instance.

        State -> (structure)
            The current state of the instance.

            Code -> (integer)
                The low byte represents the state. The high byte
                is an opaque internal value and should be ignored.

...

```

当 AWS CLI 将输出呈现为 JSON 时，输出将成为预留对象的数组，类似于以下示例。

```
{
  "Reservations": [
```

```
{
  "OwnerId": "012345678901",
  "ReservationId": "r-4c58f8a0",
  "Groups": [],
  "RequesterId": "012345678901",
  "Instances": [
    {
      "Monitoring": {
        "State": "disabled"
      },
      "PublicDnsName": "ec2-52-74-16-12.us-
west-2.compute.amazonaws.com",
      "State": {
        "Code": 16,
        "Name": "running"
      },
    },
    ...
  ]
}
```

每个预留对象都包含一些描述预留的字段和一组实例对象，每个实例对象又带有用来描述它的字段（如 `PublicDnsName`）和对象（如 `State`）。

### Windows 用户

您可以通过管道 (`|`) 将 `help` 命令的输出发送到 `more` 命令以便每次查看一页帮助文件。按空格键或 `PgDn` 键可查看文档的更多内容，按 `q` 可退出。

```
C:\> aws ec2 describe-instances help | more
```

## AWS CLI 参考指南

帮助文件包含无法通过命令行查看或导航至的链接。您可以使用线上 [AWS CLI 版本 2 参考指南](#) 查看这些链接，并与其交互。参考指南还包含所有 AWS CLI 命令的帮助内容。将显示这些说明以方便在手机、平板电脑或桌面屏幕进行浏览和查看。

## API 文档

AWS CLI 中的所有命令对应于对 AWS 服务的公用 API 发出的请求。具有公用 API 的每项服务都有一个 API 引用，可从 [AWS 文档网站](#) 上该服务的主页找到。API 参考的内容因 API 的构造方式以及所用协



议而有所不同。通常，API 参考包含有关该 API 支持的操作、发送到该服务和从该服务发送的数据以及该服务可能报告的任何错误状况的详细信息。

## API 文档的各部分

- **Actions (操作)** – 有关每个操作及其参数 (包括对长度或内容以及默认值的约束) 的详细信息。它列出了此操作可能发生的错误。每个操作对应于 AWS CLI 中的一个子命令。
- **Data Types (数据类型)** – 有关命令可能需要作为参数或在响应请求时要返回的结构的详细信息。
- **Common Parameters (常用参数)** – 有关由服务的所有操作共享的参数的详细信息。
- **Common Errors (常见错误)** – 有关可能由服务的任意操作返回的错误的详细信息。

每个部分的名称和可用性可能根据具体服务而不同。

### 特定于服务的 CLI

与以前创建单个 AWS CLI 以处理所有服务不同，有些服务还有单独的 CLI。这些特定于服务的 CLI 具有单独的文档，该服务的文档页面包含指向该文档的链接。特定于服务的 CLI 的文档不适用于 AWS CLI。

## 纠正错误

如需帮助诊断和修复 AWS CLI 错误，请参阅 [排查错误](#)。

## 其他帮助

有关 AWS CLI 问题的其他帮助，请访问 GitHub 上的 [AWS CLI 社区](#)。

## AWS CLI 中的命令结构

本主题介绍如何构造 AWS Command Line Interface (AWS CLI) 命令，以及如何使用 wait 命令。

### 主题

- [命令结构](#)
- [Wait 命令](#)

## 命令结构

AWS CLI 在命令行上使用多部分结构，各部分必须按如下顺序指定：

1. 对 aws 计划的基本调用。
2. 顶级命令，这通常对应于 AWS CLI 支持的 AWS 服务。
3. 用于指定要执行的操作的子命令。
4. 常规 AWS CLI 选项或操作所需的参数。您可以按任意顺序指定这些项，只要它们跟在前三个部分之后。如果多次指定某个排他参数，则仅应用最后一个值。

```
$ aws <command> <subcommand> [options and parameters]
```

参数可采用各种类型的输入值，如数字、字符串、列表、映射和 JSON 结构。支持的内容取决于您指定的命令和子命令。

### 示例

#### Amazon S3

以下示例列出您的所有 Amazon S3 存储桶。

```
$ aws s3 ls
2018-12-11 17:08:50 my-bucket
2018-12-14 14:55:44 my-bucket2
```

有关 Amazon S3 命令的更多信息，请参阅 AWS CLI 命令参考 中的 [aws s3](#)。

#### AWS CloudFormation

以下 [create-change-set](#) 命令示例将 cloudformation 堆栈名称更改为 *my-change-set*。

```
$ aws cloudformation create-change-set --stack-name my-stack --change-set-name my-change-set
```

有关 AWS CloudFormation 命令的更多信息，请参阅 AWS CLI 命令参考 中的 [aws cloudformation](#)。

## Wait 命令

一些 AWS 服务提供 `wait` 命令。使用 `aws wait` 的任何命令通常都会等到命令完成后再进入下一步。这对于多部分命令或脚本编写来说特别有用，因为当命令失败时，您可以使用 `wait` 命令阻止进入后续步骤。

AWS CLI 在命令行上对 `wait` 命令使用多部分结构，各部分必须按如下顺序指定：

1. 对 `aws` 计划的基本调用。
2. 顶级命令，这通常对应于 AWS CLI 支持的 AWS 服务。
3. `wait` 命令。
4. 用于指定要执行的操作的子命令。
5. 常规 CLI 选项或操作所需的参数。您可以按任意顺序指定这些项，只要它们跟在前三个部分之后。如果多次指定某个排他参数，则仅应用最后一个值。

```
$ aws <command> wait <subcommand> [options and parameters]
```

参数可采用各种类型的输入值，如数字、字符串、列表、映射和 JSON 结构。支持的内容取决于您指定的命令和子命令。

### Note

并不是所有 AWS 服务都支持 `wait` 命令。请参阅 [AWS CLI 版本 2 参考指南](#)，了解您的服务是否支持 `wait` 命令。

### 示例

#### AWS CloudFormation

在下面的 `wait change-set-create-complete` 命令示例中，命令将暂停运行，并且仅在可确认 `my-stack` 堆栈中的 `my-change-set` 更改集已准备好运行后才恢复运行。

```
$ aws cloudformation wait change-set-create-complete --stack-name my-stack --change-set-name my-change-set
```

有关 AWS CloudFormation `wait` 命令的更多信息，请参阅《AWS CLI 命令参考》中的 [wait](#)。

## AWS CodeDeploy

在下面的 [wait deployment-successful](#) 命令示例中，命令将暂停运行，直到 *d-A1B2C3111* 部署成功完成。

```
$ aws deploy wait deployment-successful --deployment-id d-A1B2C3111
```

有关 AWS CodeDeploy wait 命令的更多信息，请参阅《AWS CLI 命令参考》中的 [wait](#)。

## 为 AWS CLI 指定参数值

AWS Command Line Interface (AWS CLI) 中使用的很多参数都是简单的字符串或数值，如下面示例中的 my-key-pair 密钥对名称。

```
$ aws ec2 create-key-pair --key-name my-key-pair
```

终端之间的格式可能会有所不同。例如，大多数终端区分大小写，但 Powershell 不区分大小写。这意味着，以下两个命令示例对于区分大小写的终端会产生不同的结果，因为它们将 MyFile\*.txt 和 myfile\*.txt 视为不同的参数。

但是，PowerShell 会将 MyFile\*.txt 和 myfile\*.txt 视为相同的参数来处理这些请求。

```
$ aws s3 cp . s3://my-bucket/path --include "MyFile*.txt"  
$ aws s3 cp . s3://my-bucket/path --include "myfile*.txt"
```

有关 PowerShell 的不区分大小写的更多信息，请参阅 PowerShell 文档中的 [about\\_Case-Sensitivity](#)

有时，您需要在包含特殊字符或空格字符的字符串周围使用引号或文字。有关此格式的规则也可能因终端而异。有关在复杂参数周围使用引号的更多信息，请参阅 [AWS CLI 中的引号及字符串](#)。

### 参数主题

- [通用 AWS CLI 参数类型](#)
- [AWS CLI 中的引号及字符串](#)
- [从文件加载 AWS CLI 参数](#)
- [AWS CLI 框架和输入文件](#)
- [将速记语法与 AWS CLI 结合使用](#)

## 通用 AWS CLI 参数类型

本节介绍一些通用参数类型以及典型的所需格式。

如果您不知道如何设置特定命令的参数格式，请在命令名称后输入 **help** 来查看帮助。每个子命令的帮助均包括一个选项的名称和描述。该选项的参数类型在括号中列出。有关查看帮助的更多信息，请参阅 [the section called “获取帮助”](#)。

参数类型包括：

- [String](#)
- [Timestamp](#)
- [列表](#)
- [布尔值](#)
- [整数](#)
- [二进制/Blob \( 二进制大型对象 \) 和流式传输 Blob](#)
- [映射](#)
- [文档](#)

### String

字符串参数可以包含 [ASCII](#) 字符集中的字母数字字符、符号和空格。包含空格的字符串必须用引号引起来。建议您不要使用标准空格字符以外的符号或空格，并遵循终端的[引用规则](#)，以防止出现意外结果。

一些字符串参数可接受来自文件的二进制数据。有关示例，请参阅[二进制文件](#)。

### Timestamp

时间戳根据 [ISO 8601](#) 标准设置格式。这些通常称为“DateTime”或“Date”参数。

```
$ aws ec2 describe-spot-price-history --start-time 2014-10-13T19:00:00Z
```

可接受的格式包括：

- *YYYY-MM-DDThh:mm:ss.sssTZD (UTC)*，例如，2014-10-01T20:30:00.000Z

- `YYYY-MM-DDThh:mm:ss.sssTZD#####`，例如，`2014-10-01T12:30:00.000-08:00`
- `YYYY-MM-DD`，例如，`2014-10-01`
- 以秒为单位的 Unix 时间，如 `1412195400`。这有时称为 [Unix 纪元时间](#)，表示自 1970 年 1 月 1 日午夜 (UTC) 以来经历的秒数。

预设情况下，AWS CLI 版本 2 会将所有响应 DateTime 值转换为 ISO 8601 格式。

您可以使用 [cli\\_timestamp\\_format](#) 文件设置来设置时间戳格式。

## 列表

以空格分隔的一个或多个字符串。如果任何字符串项目包含空格，则必须用引号括起该项目。遵循您终端的[引号规则](#)以防止出现意外结果。

```
$ aws ec2 describe-spot-price-history --instance-types m1.xlarge m1.medium
```

## 布尔值

打开或关闭某一选项的二进制标志。例如，`ec2 describe-spot-price-history` 有一个布尔 `--dry-run` 参数，如果指定该参数，则针对服务验证查询而不实际运行查询。

```
$ aws ec2 describe-spot-price-history --dry-run
```

输出指示命令格式是否正确。此命令还包含一个 `--no-dry-run` 参数版本，可以用来显式指示命令应正常运行。不过不是必须包含此参数，因为这是默认行为。

## 整数

无符号整数。

```
$ aws ec2 describe-spot-price-history --max-items 5
```

## 二进制/Blob (二进制大型对象) 和流式传输 Blob

在 AWS CLI 中，您可以将二进制值作为字符串直接在命令行上传递。共有两种类型的 blob：

- [Blob](#)

- [流式 blob](#)

## Blob

要将值传递给类型为 blob 的参数，必须使用 `fileb://` 前缀指定包含二进制数据的本地文件的路径。使用 `fileb://` 前缀引用的文件始终被作为原始未编码二进制文件进行处理。指定的路径被解释为相对于当前工作目录。例如，适用于 `aws kms encrypt` 的 `--plaintext` 参数是一个 blob。

```
$ aws kms encrypt \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --plaintext fileb://ExamplePlaintextFile \  
  --output text \  
  --query CiphertextBlob | base64 \  
  --decode > ExampleEncryptedFile
```

### Note

为了向后兼容，您可以使用 `file://` 前缀。根据文件设置 [cli\\_binary\\_format](#) 或 `--cli-binary-format` 命令行选项，可以使用两种格式：

- AWS CLI 版本 2 的默认值。如果设置的值为 `base64`，则使用 `file://` 前缀引用的文件将被作为 `base64` 编码文本进行处理。
- AWS CLI 版本 1 的默认值。如果设置的值为 `raw-in-base64-out`，则使用 `file://` 前缀引用的文件会作为文本进行读取，然后 AWS CLI 尝试将其编码为二进制。

有关更多信息，请参阅文件设置 [cli\\_binary\\_format](#) 或 `--cli-binary-format` 命令行选项。

## 流式 Blob

`aws cloudsearchdomain upload-documents` 等流式 Blob 不使用前缀。相反，流式传输 blob 参数使用直接文件路径进行格式化。以下示例对于 `aws cloudsearchdomain upload-documents` 命令使用直接文件路径 `document-batch.json`：

```
$ aws cloudsearchdomain upload-documents \  
  --endpoint-url https://doc-my-domain.us-west-1.cloudsearch.amazonaws.com \  
  --content-type application/json \  
  document-batch.json
```

```
--documents document-batch.json
```

## 映射

使用 JSON 或 CLI 的[速记语法](#)指定的一系列密钥值对。以下 JSON 示例使用 map 参数 `--key` 从名为 `my-table` 的 Amazon DynamoDB 表中读取项目。此参数在嵌套的 JSON 结构中指定名为 `id` 且数值为 `1` 的主键。

要在命令行中使用更高级的 JSON，请考虑使用 `jq` 等命令行 JSON 处理器来创建 JSON 字符串。有关 `jq` 的更多信息，请参阅 GitHub 上的[jq 存储库](#)。

```
$ aws dynamodb get-item --table-name my-table --key '{"id": {"N": "1"}}'

{
  "Item": {
    "name": {
      "S": "John"
    },
    "id": {
      "N": "1"
    }
  }
}
```

## 文档

### Note

[速记语法](#)与文档类型不兼容。

文档类型用于发送数据，无需在字符串中嵌入 JSON。文档类型使服务能够提供任意架构，以便您使用更灵活的数据类型。

这使得无需对值转义，即可发送 JSON 数据。例如，不使用以下转义的 JSON 输入：

```
{"document": "{\"key\":true}"}
```

您可以使用以下文档类型：



```
{"document": {"key": true}}
```

## 文档类型的有效值

由于文档类型本身多种多样，因此存在多个有效值类型。有效值包括：

### 字符串

```
--option "value"
```

### 数字

```
--option 123  
--option 123.456
```

### 布尔值

```
--option true
```

### Null

```
--option null
```

### 数组

```
--option ["value1", "value2", "value3"]  
--option ["value", 1, true, null, ["key1", 2.34], {"key2": "value2"}]
```

### 对象

```
--option {"key": "value"}  
--option {"key1": "value1", "key2": 123, "key3": true, "key4": null, "key5":  
["value3", "value4"], "key6": {"key5": "value6"}}
```

## AWS CLI 中的引号及字符串

在 AWS CLI 中使用单引号和双引号主要有两种方式。

- [在包含空格的字符串周围使用引号](#)
- [在字符串内使用引号](#)

## 在包含空格的字符串周围使用引号

参数名称及其值由命令行中的空格分隔。如果字符串值包含嵌入式空格，则必须用引号将整个字符串括起来，以防止 AWS CLI 将空格误解为值与下一个参数名称之间的分隔符。您使用的引号类型取决于您运行 AWS CLI 的操作系统。

### Linux and macOS

使用单引号 ' '。

```
$ aws ec2 create-key-pair --key-name 'my key pair'
```

有关使用引号的更多信息，请参阅[首选 Shell 的用户文档](#)。

### PowerShell

单引号 (推荐)

单引号 ' ' 称为 verbatim 字符串。字符串将完全按照您键入的方式传递给命令，这意味着 PowerShell 变量将不会传递。

```
PS C:\> aws ec2 create-key-pair --key-name 'my key pair'
```

双引号

双引号 " " 称为 expandable 字符串。变量可以在可展开的字符串中传递。

```
PS C:\> aws ec2 create-key-pair --key-name "my key pair"
```

有关使用引号的更多信息，请参阅 Microsoft PowerShell 文档中的[关于引号规则](#)。

### Windows command prompt

使用双引号 " "。

```
C:\> aws ec2 create-key-pair --key-name "my key pair"
```

( 可选 ) 您可以用等号 = 而不是空格将参数名称和值分隔开。这通常仅在参数的值以连字符开头时有必要。

```
$ aws ec2 delete-key-pair --key-name=-mykey
```

## 在字符串内使用引号

字符串可能包含引号，并且您的 Shell 可能需要对引号进行转义才能让其正常发挥作用。常见的参数值类型之一是 JSON 字符串。这很复杂，因为它在 JSON 结构中的每个元素名称和值周围都包含空格和双引号 " "。在命令行中输入 JSON 格式参数的方式因操作系统而异。

要在命令行中使用更高级的 JSON，请考虑使用 jq 等命令行 JSON 处理器来创建 JSON 字符串。有关 jq 的更多信息，请参阅 GitHub 上的 [jq 存储库](#)。

## Linux and macOS

为了让 Linux 和 macOS 按字面含义解释字符串，请使用单引号 ' ' 将 JSON 数据结构括住，如下示例所示。您不需要对嵌入在 JSON 字符串中的双引号进行转义，因为会按字面含义对它们进行处理。由于 JSON 用单引号括住，因此字符串中的任何单引号都需要进行转义，这通常通过在单引号 \ ' 前面使用反斜杠来实现。

```
$ aws ec2 run-instances \  
  --image-id ami-12345678 \  
  --block-device-mappings '[{"DeviceName":"/dev/sdb","Ebs":  
{"VolumeSize":20,"DeleteOnTermination":false,"VolumeType":"standard"}]'
```

有关使用引号的更多信息，请参阅首选 Shell 的用户文档。

## PowerShell

使用单引号 ' ' 或双引号 " "。

### 单引号 ( 推荐 )

单引号 ' ' 称为 verbatim 字符串。字符串将完全按照您键入的方式传递给命令，这意味着 PowerShell 变量将不会传递。

由于 JSON 数据结构包含双引号，因此我们建议使用单引号 ' ' 将其括起来。如果使用单引号，则不需要对嵌入在 JSON 字符串中的双引号进行转义。但是，您需要在 JSON 结构中使用反撇号 ` 对每个单引号进行转义。

```
PS C:\> aws ec2 run-instances `
```

```
--image-id ami-12345678 `
--block-device-mappings '[{"DeviceName":"/dev/sdb","Ebs":
{"VolumeSize":20,"DeleteOnTermination":false,"VolumeType":"standard"}]`
```

## 双引号

双引号 " " 称为 expandable 字符串。变量可以在可展开的字符串中传递。

如果您使用双引号，则不需要对嵌入在 JSON 字符串中的单引号进行转义。但是，您需要在 JSON 结构中使用反撇号 ` 对每个双引号进行转义，如以下示例所示。

```
PS C:\> aws ec2 run-instances `
--image-id ami-12345678 `
--block-device-mappings "[{"DeviceName`":`"/dev/sdb`",`"Ebs`":
{"VolumeSize`":20,`"DeleteOnTermination`":false,`"VolumeType`":`"standard`"}]"
```

有关使用引号的更多信息，请参阅 Microsoft PowerShell 文档中的[关于引号规则](#)。

### Warning

PowerShell 在向 AWS CLI 发送命令之前，它会确定是使用典型 PowerShell 还是 CommandLineToArgvW 引用规则来解释您的命令。PowerShell 使用 CommandLineToArgvW 进行处理时，必须用反斜杠 \ 对字符进行转义。有关 PowerShell 中 CommandLineToArgvW 的更多信息，请参阅 Microsoft DevBlogs 上的 [What's up with the strange treatment of quotation marks and backslashes by CommandLineToArgvW](#) ( CommandLineToArgvW 对引号和反斜杠的奇怪处理是怎么回事 )、Microsoft Docs Blog 中的 [Everyone quotes command line arguments the wrong way](#) ( 每个人引用命令行参数的方法都错了 ) 以及 Microsoft Docs 上的 [CommandLineToArgvW function](#) ( CommandLineToArgvW 函数 )。

### 单引号

单引号 ' ' 称为 verbatim 字符串。字符串将完全按照您键入的方式传递给命令，这意味着 PowerShell 变量将不会传递。使用反斜杠 \ 对字符进行转义。

```
PS C:\> aws ec2 run-instances `
--image-id ami-12345678 `
--block-device-mappings '[{"DeviceName\`":\`"/dev/sdb\`",\`"Ebs\`":
{\`"VolumeSize\`":20,\`"DeleteOnTermination\`":false,\`"VolumeType\`":\`"standard\`"}]`
```

### 双引号

双引号 " " 称为 expandable 字符串。变量可以在 expandable 字符串中传递。对于双引号字符串，必须对每个引号使用 ` ` 进行两次转义，而不是仅使用反引号。反引号对反斜杠进行转义，然后将反斜杠用作 CommandLineToArgvW 流程的转义字符。

```
PS C:\> aws ec2 run-instances `
  --image-id ami-12345678 `
  --block-device-mappings "[{ `\"DeviceName `\": `\"/dev/sdb `\", `\"Ebs `\":
  { `\"VolumeSize `\":20, `\"DeleteOnTermination `\":false, `\"VolumeType `\": `
  `\"standard `\"}}]"
```

### Blob ( 推荐 )

要绕过 JSON 数据输入的 PowerShell 引用规则，请使用 Blob 将 JSON 数据直接传递到 AWS CLI。有关 Blob 的更多信息，请参阅[Blob](#)。

## Windows command prompt

Windows 命令提示符要求使用双引号 " " 括住 JSON 数据结构。此外，为了防止命令处理器误解 JSON 中嵌入的双引号，还必须对 JSON 数据结构本身中的每个双引号 ` ` 进行转义（在前面加一个反斜杠 ` ` 字符），如以下示例所示。

```
C:\> aws ec2 run-instances ^
  --image-id ami-12345678 ^
  --block-device-mappings "[{ `\"DeviceName `\": `\"/dev/sdb `\", `\"Ebs `\":
  { `\"VolumeSize `\":20, `\"DeleteOnTermination `\":false, `\"VolumeType `\": `\"standard `\"}}]"
```

只有最外层双引号不进行转义。

## 从文件加载 AWS CLI 参数

有些参数需要文件名作为变量，AWS CLI 将从这些变量中加载数据。其他参数允许您将参数值指定为在命令行上键入的文本或从文件中读取的文本。无论文件是必需的还是可选的，您都必须对文件进行正确编码，以便 AWS CLI 能够理解它。该文件的编码必须与读取系统的默认区域设置相匹配。您可以通过使用 Python `locale.getpreferredencoding()` 方法来确定这一点。

**Note**

默认情况下，Windows PowerShell 将文本输出为 UTF-16，这与 JSON 文件和许多 Linux 系统使用的 UTF-8 编码冲突。建议您将 `-Encoding ascii` 与 PowerShell `Out-File` 命令一起使用，以确保 AWS CLI 可以读取生成的文件。

**主题**

- [如何从文件加载参数](#)
- [二进制文件](#)

**如何从文件加载参数**

有时，从文件加载参数值（而不是尝试将其全部键入为命令行参数值）很方便，例如当参数为复杂的 JSON 字符串时。要指定包含该值的文件，请按以下格式指定文件 URL。

```
file://complete/path/to/file
```

- 前两个斜杠“/”字符是规范的一部分。如果所需的路径以“/”开头，结果为三个斜杠字符：`file:///folder/file`。
- URL 提供包含实际参数内容的文件的路径。
- 使用带空格或特殊字符的文件时，请遵循终端的[引用和转义规则](#)。

以下示例中的文件路径被解读为相对于当前工作目录。

**Linux or macOS**

```
// Read from a file in the current directory
$ aws ec2 describe-instances --filters file://filter.json

// Read from a file in /tmp
$ aws ec2 describe-instances --filters file:///tmp/filter.json

// Read from a file with a filename with whitespaces
$ aws ec2 describe-instances --filters 'file://filter content.json'
```

## Windows command prompt

```
// Read from a file in C:\temp
C:\> aws ec2 describe-instances --filters file://C:\temp\filter.json

// Read from a file with a filename with whitespaces
C:\> aws ec2 describe-instances --filters "file://C:\temp\filter content.json"
```

`file://` 前缀选项支持包含“~/”、“./”和“../”的 Unix 式扩展。在 Windows 上，“~/”表达式将展开到您的用户目录（存储在 `%USERPROFILE%` 环境变量中）。例如，在 Windows 10 上，您通常在 `C:\Users\UserName\` 下有一个用户目录。

仍必须对作为另一个 JSON 文档的值嵌入的 JSON 文档进行转义。

```
$ aws sqs create-queue --queue-name my-queue --attributes file://attributes.json
```

### attributes.json

```
{
  "RedrivePolicy": "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-west-2:0123456789012:deadletter\", \"maxReceiveCount\":\"5\"}"
}
```

## 二进制文件

对于将二进制数据用作参数的命令，请使用 `fileb://` 前缀指定该数据为二进制内容。接受二进制数据的命令包括：

- **aws ec2 run-instances:** `--user-data` 参数。
- **aws s3api put-object:** `--sse-customer-key` 参数。
- **aws kms decrypt:** `--ciphertext-blob` 参数。

以下示例使用 Linux 命令行工具生成一个二进制 256 位 AES 密钥，然后将该密钥提供给 Amazon S3 以对上传的文件服务器端进行加密。

```
$ dd if=/dev/urandom bs=1 count=32 > sse.key
32+0 records in
32+0 records out
```

```
32 bytes (32 B) copied, 0.000164441 s, 195 kB/s
$ aws s3api put-object \
  --bucket my-bucket \
  --key test.txt \
  --body test.txt \
  --sse-customer-key fileb://sse.key \
  --sse-customer-algorithm AES256
{
  "SSECustomerKeyMD5": "iVg8oWa8sy714+FjtesrJg==",
  "SSECustomerAlgorithm": "AES256",
  "ETag": "\"a6118e84b76cf98bf04bbe14b6045c6c\""
}
```

有关引用包含 JSON 格式参数的文件的更多示例，请参阅[将 IAM 托管策略附加到用户](#)。

## AWS CLI 框架和输入文件

大多数 AWS CLI 命令接受来自文件的所有参数输入。可以使用 `generate-cli-skeleton` 选项生成这些模板。

### 主题

- [关于 AWS CLI 框架和输入文件](#)
- [生成命令框架](#)

## 关于 AWS CLI 框架和输入文件

大多数 AWS Command Line Interface (AWS CLI) 命令支持使用 `--cli-input-json` 和 `--cli-input-yaml` 参数接受文件中的所有参数输入的功能。

这些相同的命令帮助提供 `--generate-cli-skeleton` 参数，以使用您可以编辑和填写的所有参数以 JSON 或 YAML 格式生成文件。然后，您可以带有相关 `--cli-input-json` 或 `--cli-input-yaml` 参数运行命令并指向填充的文件。

### Important

多个 AWS CLI 命令不直接映射到各个 AWS API 操作，例如 [aws s3 命令](#)。此类命令不支持本主题中介绍的 `--generate-cli-skeleton` 或 `--cli-input-json` 和 `--cli-input-yaml` 参数。如果您不知道特定命令是否支持这些参数，请运行以下命令，将 `service` 和 `command` 名称替换为您感兴趣的名称。



```
$ aws service command help
```

输出包含 Synopsis 部分，其中显示了指定的命令支持的参数。

```
$ aws iam list-users help
...
SYNOPSIS
    list-users
    ...
    [--cli-input-json | --cli-input-yaml]
    ...
    [--generate-cli-skeleton <value>]
...
```

`--generate-cli-skeleton` 参数将导致命令无法运行，而是生成和显示您可以自定义的参数模板并用作以后命令的输入。生成的模板包含命令支持的所有参数。

`--generate-cli-skeleton` 参数接受以下值之一：

- `input` – 生成的模板包括格式化为 JSON 的所有输入参数。这是默认值。
- `yaml-input` – 生成的模板包括格式化为 YAML 的所有输入参数。
- `output` – 生成的模板包括格式化为 JSON 的所有输出参数。您当前无法请求输出参数作为 YAML。

由于 AWS CLI 本质上是围绕服务 API 的“包装程序”，骨架文件预计您会通过他们底层的 API 参数名称引用所有参数。该参数名称可能与 AWS CLI 参数名称不同。例如，一个名为 AWS CLI 的 `username` 参数可能映射到名为 AWS 的 `UserName` 服务 API 参数（注意，更改后的大写字母和缺失的破折号）。我们建议您使用 `--generate-cli-skeleton` 选项，用“正确”的参数名称生成模板，以避免错误。您也可以引用服务的 API 参考指南，查看预期的参数名称。您可以从模板中删除不需要的和不想为其提供值的任何参数。

例如，如果您运行以下命令，它会为 Amazon Elastic Compute Cloud (Amazon EC2) 命令 `run-instances` 生成参数模板。

## JSON

以下示例演示如何使用 `input` 参数的默认值 (`--generate-cli-skeleton`) 生成格式化为 JSON 的模板。

```
$ aws ec2 run-instances --generate-cli-skeleton
```

```
{
  "DryRun": true,
  "ImageId": "",
  "MinCount": 0,
  "MaxCount": 0,
  "KeyName": "",
  "SecurityGroups": [
    ""
  ],
  "SecurityGroupIds": [
    ""
  ],
  "UserData": "",
  "InstanceType": "",
  "Placement": {
    "AvailabilityZone": "",
    "GroupName": "",
    "Tenancy": ""
  },
  "KernelId": "",
  "RamdiskId": "",
  "BlockDeviceMappings": [
    {
      "VirtualName": "",
      "DeviceName": "",
      "Ebs": {
        "SnapshotId": "",
        "VolumeSize": 0,
        "DeleteOnTermination": true,
        "VolumeType": "",
        "Iops": 0,
        "Encrypted": true
      },
      "NoDevice": ""
    }
  ],
  "Monitoring": {
    "Enabled": true
  },
  "SubnetId": "",
  "DisableApiTermination": true,
```

```

"InstanceInitiatedShutdownBehavior": "",
"PrivateIpAddress": "",
"ClientToken": "",
"AdditionalInfo": "",
"NetworkInterfaces": [
  {
    "NetworkInterfaceId": "",
    "DeviceIndex": 0,
    "SubnetId": "",
    "Description": "",
    "PrivateIpAddress": "",
    "Groups": [
      ""
    ],
    "DeleteOnTermination": true,
    "PrivateIpAddresses": [
      {
        "PrivateIpAddress": "",
        "Primary": true
      }
    ],
    "SecondaryPrivateIpAddressCount": 0,
    "AssociatePublicIpAddress": true
  }
],
"IamInstanceProfile": {
  "Arn": "",
  "Name": ""
},
"EbsOptimized": true
}

```

## YAML

以下示例演示如何使用 `--generate-cli-skeleton` 参数的值 `yaml-input` 生成格式化为 YAML 的模板。

```
$ aws ec2 run-instances --generate-cli-skeleton yaml-input
```

```

BlockDeviceMappings: # The block device mapping entries.
- DeviceName: '' # The device name (for example, /dev/sdh or xvdh).
  VirtualName: '' # The virtual device name (ephemeralN).

```

Ebs: # Parameters used to automatically set up Amazon EBS volumes when the instance is launched.

DeleteOnTermination: true # Indicates whether the EBS volume is deleted on instance termination.

Iops: 0 # The number of I/O operations per second (IOPS) that the volume supports.

SnapshotId: '' # The ID of the snapshot.

VolumeSize: 0 # The size of the volume, in GiB.

VolumeType: st1 # The volume type. Valid values are: standard, io1, gp2, sc1, st1.

Encrypted: true # Indicates whether the encryption state of an EBS volume is changed while being restored from a backing snapshot.

KmsKeyId: '' # Identifier (key ID, key alias, ID ARN, or alias ARN) for a customer managed KMS key under which the EBS volume is encrypted.

NoDevice: '' # Suppresses the specified device included in the block device mapping of the AMI.

ImageId: '' # The ID of the AMI.

InstanceType: c4.4xlarge # The instance type. Valid values are: t1.micro, t2.nano, t2.micro, t2.small, t2.medium, t2.large, t2.xlarge, t2.2xlarge, t3.nano, t3.micro, t3.small, t3.medium, t3.large, t3.xlarge, t3.2xlarge, t3a.nano, t3a.micro, t3a.small, t3a.medium, t3a.large, t3a.xlarge, t3a.2xlarge, m1.small, m1.medium, m1.large, m1.xlarge, m3.medium, m3.large, m3.xlarge, m3.2xlarge, m4.large, m4.xlarge, m4.2xlarge, m4.4xlarge, m4.10xlarge, m4.16xlarge, m2.xlarge, m2.2xlarge, m2.4xlarge, cr1.8xlarge, r3.large, r3.xlarge, r3.2xlarge, r3.4xlarge, r3.8xlarge, r4.large, r4.xlarge, r4.2xlarge, r4.4xlarge, r4.8xlarge, r4.16xlarge, r5.large, r5.xlarge, r5.2xlarge, r5.4xlarge, r5.8xlarge, r5.12xlarge, r5.16xlarge, r5.24xlarge, r5.metal, r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12xlarge, r5a.16xlarge, r5a.24xlarge, r5d.large, r5d.xlarge, r5d.2xlarge, r5d.4xlarge, r5d.8xlarge, r5d.12xlarge, r5d.16xlarge, r5d.24xlarge, r5d.metal, r5ad.large, r5ad.xlarge, r5ad.2xlarge, r5ad.4xlarge, r5ad.8xlarge, r5ad.12xlarge, r5ad.16xlarge, r5ad.24xlarge, x1.16xlarge, x1.32xlarge, x1e.xlarge, x1e.2xlarge, x1e.4xlarge, x1e.8xlarge, x1e.16xlarge, x1e.32xlarge, i2.xlarge, i2.2xlarge, i2.4xlarge, i2.8xlarge, i3.large, i3.xlarge, i3.2xlarge, i3.4xlarge, i3.8xlarge, i3.16xlarge, i3.metal, i3en.large, i3en.xlarge, i3en.2xlarge, i3en.3xlarge, i3en.6xlarge, i3en.12xlarge, i3en.24xlarge, i3en.metal, hi1.4xlarge, hs1.8xlarge, c1.medium, c1.xlarge, c3.large, c3.xlarge, c3.2xlarge, c3.4xlarge, c3.8xlarge, c4.large, c4.xlarge, c4.2xlarge, c4.4xlarge, c4.8xlarge, c5.large, c5.xlarge, c5.2xlarge, c5.4xlarge, c5.9xlarge, c5.12xlarge, c5.18xlarge, c5.24xlarge, c5.metal, c5d.large, c5d.xlarge, c5d.2xlarge, c5d.4xlarge, c5d.9xlarge, c5d.18xlarge, c5n.large, c5n.xlarge, c5n.2xlarge, c5n.4xlarge, c5n.9xlarge, c5n.18xlarge, cc1.4xlarge, cc2.8xlarge, g2.2xlarge, g2.8xlarge, g3.4xlarge, g3.8xlarge, g3.16xlarge, g3s.xlarge, g4dn.xlarge, g4dn.2xlarge, g4dn.4xlarge, g4dn.8xlarge, g4dn.12xlarge, g4dn.16xlarge, cg1.4xlarge, p2.xlarge, p2.8xlarge, p2.16xlarge, p3.2xlarge, p3.8xlarge, p3.16xlarge, p3dn.24xlarge,

d2.xlarge, d2.2xlarge, d2.4xlarge, d2.8xlarge, f1.2xlarge, f1.4xlarge, f1.16xlarge, m5.large, m5.xlarge, m5.2xlarge, m5.4xlarge, m5.8xlarge, m5.12xlarge, m5.16xlarge, m5.24xlarge, m5.metal, m5a.large, m5a.xlarge, m5a.2xlarge, m5a.4xlarge, m5a.8xlarge, m5a.12xlarge, m5a.16xlarge, m5a.24xlarge, m5d.large, m5d.xlarge, m5d.2xlarge, m5d.4xlarge, m5d.8xlarge, m5d.12xlarge, m5d.16xlarge, m5d.24xlarge, m5d.metal, m5ad.large, m5ad.xlarge, m5ad.2xlarge, m5ad.4xlarge, m5ad.8xlarge, m5ad.12xlarge, m5ad.16xlarge, m5ad.24xlarge, h1.2xlarge, h1.4xlarge, h1.8xlarge, h1.16xlarge, z1d.large, z1d.xlarge, z1d.2xlarge, z1d.3xlarge, z1d.6xlarge, z1d.12xlarge, z1d.metal, u-6tb1.metal, u-9tb1.metal, u-12tb1.metal, u-18tb1.metal, u-24tb1.metal, a1.medium, a1.large, a1.xlarge, a1.2xlarge, a1.4xlarge, a1.metal, m5dn.large, m5dn.xlarge, m5dn.2xlarge, m5dn.4xlarge, m5dn.8xlarge, m5dn.12xlarge, m5dn.16xlarge, m5dn.24xlarge, m5n.large, m5n.xlarge, m5n.2xlarge, m5n.4xlarge, m5n.8xlarge, m5n.12xlarge, m5n.16xlarge, m5n.24xlarge, r5dn.large, r5dn.xlarge, r5dn.2xlarge, r5dn.4xlarge, r5dn.8xlarge, r5dn.12xlarge, r5dn.16xlarge, r5dn.24xlarge, r5n.large, r5n.xlarge, r5n.2xlarge, r5n.4xlarge, r5n.8xlarge, r5n.12xlarge, r5n.16xlarge, r5n.24xlarge.

Ipv6AddressCount: 0 # [EC2-VPC] The number of IPv6 addresses to associate with the primary network interface.

Ipv6Addresses: # [EC2-VPC] The IPv6 addresses from the range of the subnet to associate with the primary network interface.

- Ipv6Address: '' # The IPv6 address.

KernelId: '' # The ID of the kernel.

KeyName: '' # The name of the key pair.

MaxCount: 0 # [REQUIRED] The maximum number of instances to launch.

MinCount: 0 # [REQUIRED] The minimum number of instances to launch.

Monitoring: # Specifies whether detailed monitoring is enabled for the instance.

Enabled: true # [REQUIRED] Indicates whether detailed monitoring is enabled.

Placement: # The placement for the instance.

AvailabilityZone: '' # The Availability Zone of the instance.

Affinity: '' # The affinity setting for the instance on the Dedicated Host.

GroupName: '' # The name of the placement group the instance is in.

PartitionNumber: 0 # The number of the partition the instance is in.

HostId: '' # The ID of the Dedicated Host on which the instance resides.

Tenancy: dedicated # The tenancy of the instance (if the instance is running in a VPC). Valid values are: default, dedicated, host.

SpreadDomain: '' # Reserved for future use.

RamdiskId: '' # The ID of the RAM disk to select.

SecurityGroupIds: # The IDs of the security groups.

- ''

SecurityGroups: # [default VPC] The names of the security groups.

- ''

SubnetId: '' # [EC2-VPC] The ID of the subnet to launch the instance into.

UserData: '' # The user data to make available to the instance.

AdditionalInfo: '' # Reserved.

```
ClientToken: '' # Unique, case-sensitive identifier you provide to ensure the
idempotency of the request.
DisableApiTermination: true # If you set this parameter to true, you can't terminate
the instance using the Amazon EC2 console, CLI, or API; otherwise, you can.
DryRun: true # Checks whether you have the required permissions for the action,
without actually making the request, and provides an error response.
EbsOptimized: true # Indicates whether the instance is optimized for Amazon EBS I/O.
IamInstanceProfile: # The IAM instance profile.
  Arn: '' # The Amazon Resource Name (ARN) of the instance profile.
  Name: '' # The name of the instance profile.
InstanceInitiatedShutdownBehavior: stop # Indicates whether an instance stops or
terminates when you initiate shutdown from the instance (using the operating system
command for system shutdown). Valid values are: stop, terminate.
NetworkInterfaces: # The network interfaces to associate with the instance.
- AssociatePublicIpAddress: true # Indicates whether to assign a public IPv4
address to an instance you launch in a VPC.
  DeleteOnTermination: true # If set to true, the interface is deleted when the
instance is terminated.
  Description: '' # The description of the network interface.
  DeviceIndex: 0 # The position of the network interface in the attachment order.
  Groups: # The IDs of the security groups for the network interface.
  - ''
  Ipv6AddressCount: 0 # A number of IPv6 addresses to assign to the network
interface.
  Ipv6Addresses: # One or more IPv6 addresses to assign to the network interface.
  - Ipv6Address: '' # The IPv6 address.
  NetworkInterfaceId: '' # The ID of the network interface.
  PrivateIpAddress: '' # The private IPv4 address of the network interface.
  PrivateIpAddresses: # One or more private IPv4 addresses to assign to the network
interface.
  - Primary: true # Indicates whether the private IPv4 address is the primary
private IPv4 address.
    PrivateIpAddress: '' # The private IPv4 addresses.
  SecondaryPrivateIpAddressCount: 0 # The number of secondary private IPv4
addresses.
  SubnetId: '' # The ID of the subnet associated with the network interface.
  InterfaceType: '' # The type of network interface.
PrivateIpAddress: '' # [EC2-VPC] The primary IPv4 address.
ElasticGpuSpecification: # An elastic GPU to associate with the instance.
- Type: '' # [REQUIRED] The type of Elastic Graphics accelerator.
ElasticInferenceAccelerators: # An elastic inference accelerator to associate with
the instance.
- Type: '' # [REQUIRED] The type of elastic inference accelerator.
TagSpecifications: # The tags to apply to the resources during launch.
```

```
- ResourceType: network-interface # The type of resource to tag. Valid values
are: client-vpn-endpoint, customer-gateway, dedicated-host, dhcp-options, elastic-
ip, fleet, fpga-image, host-reservation, image, instance, internet-gateway,
launch-template, natgateway, network-acl, network-interface, reserved-instances,
route-table, security-group, snapshot, spot-instances-request, subnet, traffic-
mirror-filter, traffic-mirror-session, traffic-mirror-target, transit-gateway,
transit-gateway-attachment, transit-gateway-route-table, volume, vpc, vpc-peering-
connection, vpn-connection, vpn-gateway.
  Tags: # The tags to apply to the resource.
    - Key: '' # The key of the tag.
      Value: '' # The value of the tag.
LaunchTemplate: # The launch template to use to launch the instances.
  LaunchTemplateId: '' # The ID of the launch template.
  LaunchTemplateName: '' # The name of the launch template.
  Version: '' # The version number of the launch template.
InstanceMarketOptions: # The market (purchasing) option for the instances.
  MarketType: spot # The market type. Valid values are: spot.
  SpotOptions: # The options for Spot Instances.
    MaxPrice: '' # The maximum hourly price you're willing to pay for the Spot
Instances.
    SpotInstanceType: one-time # The Spot Instance request type. Valid values are:
one-time, persistent.
    BlockDurationMinutes: 0 # The required duration for the Spot Instances (also
known as Spot blocks), in minutes.
    ValidUntil: 1970-01-01 00:00:00 # The end date of the request.
    InstanceInterruptionBehavior: terminate # The behavior when a Spot Instance is
interrupted. Valid values are: hibernate, stop, terminate.
CreditSpecification: # The credit option for CPU usage of the T2 or T3 instance.
  CpuCredits: '' # [REQUIRED] The credit option for CPU usage of a T2 or T3
instance.
CpuOptions: # The CPU options for the instance.
  CoreCount: 0 # The number of CPU cores for the instance.
  ThreadsPerCore: 0 # The number of threads per CPU core.
CapacityReservationSpecification: # Information about the Capacity Reservation
targeting option.
  CapacityReservationPreference: none # Indicates the instance's Capacity
Reservation preferences. Valid values are: open, none.
  CapacityReservationTarget: # Information about the target Capacity Reservation.
    CapacityReservationId: '' # The ID of the Capacity Reservation.
HibernationOptions: # Indicates whether an instance is enabled for hibernation.
  Configured: true # If you set this parameter to true, your instance is enabled
for hibernation.
LicenseSpecifications: # The license configurations.
```

```
- LicenseConfigurationArn: '' # The Amazon Resource Name (ARN) of the license configuration.
```

## 生成命令框架

### 生成并使用参数框架文件

1. 使用 `--generate-cli-skeleton` 参数运行命令以生成 JSON 或 YAML，并将输出定向到某个文件以保存它。

#### JSON

```
$ aws ec2 run-instances --generate-cli-skeleton input > ec2runinst.json
```

#### YAML

```
$ aws ec2 run-instances --generate-cli-skeleton yaml-input > ec2runinst.yaml
```

2. 在文本编辑器中打开参数骨架文件，并删除任何不需要的参数。例如，您可以将模板缩减到以下内容。请确保在删除不需要的元素后，文件仍是有效的 JSON 或 YAML。

#### JSON

```
{
  "DryRun": true,
  "ImageId": "",
  "KeyName": "",
  "SecurityGroups": [
    ""
  ],
  "InstanceType": "",
  "Monitoring": {
    "Enabled": true
  }
}
```

#### YAML

```
DryRun: true
ImageId: ''
```



```
KeyName: ''
SecurityGroups:
- ''
InstanceType:
Monitoring:
  Enabled: true
```

在此示例中，我们将 `DryRun` 参数设置为 `true` 以使用 Amazon EC2 空运行功能。通过此功能，您可以安全地测试命令，而无需实际创建或修改任何资源。

3. 使用适合您的场景的值填入剩余的值。在本例中，我们提供要使用的 Amazon 系统映像 (AMI) 的实例类型、密钥名称、安全组和标识符。此示例假定默认的 AWS 区域。AMI `ami-dfc39aef` 是 `us-west-2` 区域中托管的 64 位 Amazon Linux 映像。如果您使用不同的区域，您必须[查找正确 AMI ID 来使用](#)。

## JSON

```
{
  "DryRun": true,
  "ImageId": "ami-dfc39aef",
  "KeyName": "mykey",
  "SecurityGroups": [
    "my-sg"
  ],
  "InstanceType": "t2.micro",
  "Monitoring": {
    "Enabled": true
  }
}
```

## YAML

```
DryRun: true
ImageId: 'ami-dfc39aef'
KeyName: 'mykey'
SecurityGroups:
- 'my-sg'
InstanceType: 't2.micro'
Monitoring:
  Enabled: true
```

4. 通过使用 `file://` 前缀将完成的模板文件传递到 `--cli-input-json` 或 `--cli-input-yaml` 参数，使用填写的参数运行命令。AWS CLI 将路径解释为相对于当前工作目录，因此，在以下仅显示文件名而不带路径的示例中，它将直接在当前工作目录中查找此文件。

## JSON

```
$ aws ec2 run-instances --cli-input-json file://ec2runinst.json
```

```
A client error (DryRunOperation) occurred when calling the RunInstances operation: Request would have succeeded, but DryRun flag is set.
```

## YAML

```
$ aws ec2 run-instances --cli-input-yaml file://ec2runinst.yaml
```

```
A client error (DryRunOperation) occurred when calling the RunInstances operation: Request would have succeeded, but DryRun flag is set.
```

空运行错误表明，JSON 或 YAML 格式正确且参数值有效。如果输出中报告了其他问题，请解决这些问题并重复上一步，直到显示“Request would have succeeded”消息。

5. 现在，您可以将 `DryRun` 参数设置为 `false` 以禁用空运行。

## JSON

```
{
  "DryRun": false,
  "ImageId": "ami-dfc39aef",
  "KeyName": "mykey",
  "SecurityGroups": [
    "my-sg"
  ],
  "InstanceType": "t2.micro",
  "Monitoring": {
    "Enabled": true
  }
}
```

## YAML

```
DryRun: false
ImageId: 'ami-dfc39aef'
KeyName: 'mykey'
SecurityGroups:
- 'my-sg'
InstanceType: 't2.micro'
Monitoring:
  Enabled: true
```

6. 运行此命令，`run-instances` 实际上会启动 Amazon EC2 实例并显示成功启动所生成的详细信息。输出格式由 `--output` 参数控制，与输入参数模板的格式分开。

## JSON

```
$ aws ec2 run-instances --cli-input-json file://ec2runinst.json --output json
```

```
{
  "OwnerId": "123456789012",
  "ReservationId": "r-d94a2b1",
  "Groups": [],
  "Instances": [
  ...
```

## YAML

```
$ aws ec2 run-instances --cli-input-yaml file://ec2runinst.yaml --output yaml
```

```
OwnerId: '123456789012'
ReservationId: 'r-d94a2b1',
Groups:
- ''
Instances:
...
```

## 将速记语法与 AWS CLI 结合使用

AWS Command Line Interface (AWS CLI) 可以接受 JSON 格式的许多选项。但是，在命令行上输入较大的 JSON 列表或结构会比较繁琐。为了简化此过程，AWS CLI 还支持一种速记语法，允许采用比完整 JSON 格式更简单的方式表示选项参数。

### 主题

- [结构参数](#)
- [将速记语法与 AWS Command Line Interface 结合使用](#)

### 结构参数

通过 AWS CLI 中的速记语法，用户更容易输入平面（非嵌套结构）参数。格式采用以逗号分隔的键值对列表。请务必使用适用于终端的[引用](#)以及转义规则，因为速记语法是字符串。

#### Linux or macOS

```
--option key1=value1,key2=value2,key3=value3
```

#### PowerShell

```
--option "key1=value1,key2=value2,key3=value3"
```

这两者均等同于以下采用 JSON 格式的示例。

```
--option '{"key1":"value1","key2":"value2","key3":"value3"}'
```

各逗号分隔的键值对之间不能有空格。下面的示例 Amazon DynamoDB update-table 命令包含采用速记语法指定的 --provisioned-throughput 选项。

```
$ aws dynamodb update-table \  
  --provisioned-throughput ReadCapacityUnits=15,WriteCapacityUnits=10 \  
  --table-name MyDDBTable
```

该示例等同于以下 JSON 格式的示例。

```
$ aws dynamodb update-table \  
  --provisioned-throughput '{"ReadCapacityUnits":15,"WriteCapacityUnits":10}' \  
  --table-name MyDDBTable
```

```
--table-name MyDDBTable
```

## 将速记语法与 AWS Command Line Interface 结合使用

您可以使用两种方法以列表形式指定输入参数：JSON 或速记。使用 AWS CLI 速记语法，可更方便地传入含有数字、字符串或非嵌套结构的列表。

下面显示了基本格式，列表中的值用单个空格分隔。

```
--option value1 value2 value3
```

该示例等同于以下 JSON 格式的示例。

```
--option '[value1,value2,value3]'
```

如前所述，您可以用速记语法指定数字列表、字符串列表或非嵌套结构的列表。以下是用于 Amazon Elastic Compute Cloud (Amazon EC2) 的 `stop-instances` 命令示例，其中，`--instance-ids` 选项的输入参数（字符串列表）采用速记语法指定。

```
$ aws ec2 stop-instances \  
  --instance-ids i-1486157a i-1286157c i-ec3a7e87
```

该示例等同于以下 JSON 格式的示例。

```
$ aws ec2 stop-instances \  
  --instance-ids ["i-1486157a","i-1286157c","i-ec3a7e87"]'
```

下面的示例显示 Amazon EC2 `create-tags` 命令，该命令针对 `--tags` 选项使用非嵌套结构的列表。`--resources` 选项指定要添加标签的实例的 ID。

```
$ aws ec2 create-tags \  
  --resources i-1286157c \  
  --tags Key=My1stTag,Value=Value1 Key=My2ndTag,Value=Value2  
        Key=My3rdTag,Value=Value3
```

该示例等同于以下 JSON 格式的示例。JSON 参数分多行编写以便于阅读。

```
$ aws ec2 create-tags \  
  --resources i-1286157c \  
  --tags '['
```

```
{"Key": "My1stTag", "Value": "Value1"},  
{"Key": "My2ndTag", "Value": "Value2"},  
{"Key": "My3rdTag", "Value": "Value3"}  
]
```

## 让 AWS CLI 提示您输入命令

运行 `aws` 命令时，您可以让 AWS CLI 版本 2 提示输入命令、参数和资源。

### 主题

- [工作原理](#)
- [自动提示功能](#)
- [自动提示模式](#)
- [配置自动提示](#)

## 工作原理

如果启用，自动提示可让您使用 Enter 键完成部分输入的命令。按 Enter 键后，根据您继续键入的内容，建议命令、参数和资源。建议在左侧列出了命令、参数或资源的名称，在右侧列出了该命令、参数或资源的说明。要选择并使用建议，请使用箭头键突出显示某个行，然后按空格键。输入完命令后，按 Enter 键以使用该命令。以下示例演示了自动提示中的建议列表的外观。

```
$ aws  
> aws a  
    accessanalyzer      Access Analyzer  
    acm                  AWS Certificate Manager  
    acm-pca              AWS Certificate Manager Private Certificate  
    Authority  
    alexaforbusiness    Alexa For Business  
    amplify              AWS Amplify
```

## 自动提示功能

自动提示包含以下有用功能：

### 文档面板

提供当前命令的帮助文档。要打开文档，请按 F3 键。

## 命令完成

建议要使用的 `aws` 命令。要查看列表，请部分输入命令。以下示例正在搜索以字母 `a` 开头的服务。

```
$ aws
> aws a
    accessanalyzer           Access Analyzer
    acm                      AWS Certificate Manager
    acm-pca                  AWS Certificate Manager Private Certificate
    Authority
    alexaforbusiness        Alexa For Business
    amplify                 AWS Amplify
```

## 参数完成

键入命令后，自动提示功能会开始建议参数。参数的描述包括值类型以及对参数定义的描述。首先列出必要的参数，并将它们标记为必需。以下示例显示了 `aws dynamodb describe-table` 的自动提示参数列表。

```
$ aws dynamodb describe-table
> aws dynamodb describe-table
    --table-name (required) [string] The name of the
table to describe.
    --cli-input-json [string] Reads arguments
from the JSON string provided. The JSON string follows the format provide...
    --cli-input-yaml [string] Reads arguments
from the YAML string provided. The YAML string follows the format provide...
    --generate-cli-skeleton [string] Prints a JSON
skeleton to standard output without sending an API request. If provided wit...
```

## 资源完成

自动提示会使用可用的 AWS 资源属性调用 AWS API 来建议资源值。这样就能够输入参数时自动提示以建议您可能拥有的资源。在以下示例中，在填写 `aws dynamodb describe-table` 命令的 `--table-name` 参数时，自动提示会列出表名称。

```
$ aws dynamodb describe-table
> aws dynamodb describe-table --table-name
    Table1
    Table2
```

Table 3

## 速记完成

对于使用速记语法的参数，自动提示会建议要使用的值。在以下示例中，自动提示列出了 `aws ec2 run-instances` 命令中 `--placement` 参数的速记语法值。

```
$ aws ec2 run-instances
> aws ec2 run-instances --placement
AvailabilityZone=      [string] The Availability Zone of the instance. If not
specified, an Availability Zone wil...
Affinity=              [string] The affinity setting for the instance on the
Dedicated Host. This parameter is no...
GroupName=            [string] The name of the placement group the instance is in.
PartitionNumber=      [integer] The number of the partition the instance is in.
Valid only if the placement grou...
```

## 文件完成

在 `aws` 命令中填写参数时，自动完成会在使用前缀 `file://` 或 `fileb://` 后建议本地文件名。在以下示例中，在为 `aws ec2 run-instances` 命令输入 `--item file://` 后，自动提示会建议使用本地文件。

```
$ aws ec2 run-instances
> aws ec2 run-instances --item file://
    item1.txt
    file1.json
    file2.json
```

## 区域完成

使用全局参数 `--region` 时，自动提示会列出可供选择的可能区域。在以下示例中，在为 `aws dynamodb list-tables` 命令输入 `--region` 后，自动提示会按字母顺序建议区域。

```
$ aws dynamodb list-tables
> aws dynamodb list-tables --region
    af-south-1
    ap-east-1
    ap-northeast-1
    ap-northeast-2
```



## 配置文件完成

使用全局参数 `--profile` 时，自动提示会列出您的配置文件。在以下示例中，在为 `aws dynamodb list-tables` 命令输入 `--profile` 后，自动提示会建议您的配置文件。

```
$ aws dynamodb list-tables
> aws dynamodb list-tables --profile
                                profile1
                                profile2
                                profile3
```

## 模糊搜索

包含一组特定字符的完整命令和值。在以下示例中，在为 `aws dynamodb list-tables` 命令输入 `--region eu` 后，自动提示会建议包含 `eu` 的区域。

```
$ aws dynamodb list-tables
> aws dynamodb list-tables --region west
                                eu-west-1
                                eu-west-2
                                eu-west-3
                                us-west-1
```

## 历史记录

要在自动提示模式下查看和运行先前使用的命令，请按 `Ctrl + R`。历史记录列出了以前的命令，您可以使用箭头键进行选择。在以下示例中，将显示自动提示模式历史记录。

```
$ aws
> aws
    dynamodb list-tables
    s3 ls
```

## 自动提示模式

AWS CLI 版本 2 的自动提示有 2 种可以配置的模式：

- 完全模式：无论是使用 `--cli-auto-prompt` 参数手动调用 `aws` 命令还是永久启用该命令，每次尝试运行该命令时都会使用自动提示。这包括在完整或不完整的命令之后按 `Enter` 键。

- 部分模式：如果命令不完整或由于客户端验证错误而无法运行，则使用自动提示。如果您已有现有脚本、运行手册，或者如果您希望只对于不熟悉的命令收到自动提示，而不是每个命令都收到提示，则此模式将非常有帮助。

## 配置自动提示

要配置自动提示，您可以按优先顺序使用以下方法：

- 命令行选项启用或禁用单个命令的自动提示。使用 [--cli-auto-prompt](#) 调用自动提示，使用 [--no-cli-auto-prompt](#) 禁用自动提示。
- 环境变量使用 [aws\\_cli\\_auto\\_prompt](#) 变量。
- 共享的配置文件使用 [cli\\_auto\\_prompt](#) 设置。

## 控制来自的命令输出 AWS CLI

本部分介绍控制 AWS Command Line Interface (AWS CLI) 的输出的不同方式。在终端中自定义 AWS CLI 输出可以提高可读性，简化脚本自动化，并在更大的数据集中更轻松地导航。

AWS CLI 支持多种[输出格式](#)，包括[json](#)、[yaml](#)、和[table](#)。有些服务为其数据提供了服务器端[分页](#)功能，并 AWS CLI 提供了自己的客户端功能以提供其他分页选项。

最后，AWS CLI 具有[服务器端和客户端筛选功能](#)，您可以单独或一起使用它们来 AWS CLI 筛选输出。

### 主题

- [灵敏输出](#)
- [服务器端与客户端输出选项](#)
- [设置 AWS CLI 输出格式](#)
- [使用 AWS CLI 分页选项](#)
- [过滤器 AWS CLI 输出](#)

## 灵敏输出

的某些操作 AWS CLI 可能会返回可能被视为敏感的信息，包括来自环境变量的信息。在某些情况下，这些信息的泄露可能构成安全风险；例如，这些信息可能包含在持续集成和持续部署 (CI/CD) 日志中。因此，请务必查看何时将此类输出作为日志的一部分，并在不需要时隐藏该输出。

有关保护敏感数据的其他信息，请参见[the section called “数据保护”](#)。

考虑下面的最佳实践：

- 可以考虑以编程方式从密钥库中检索您的密钥，例如。AWS Secrets Manager
- 查看构建日志的内容，确保其中不包含敏感信息。考虑诸如管道传送/dev/null或将输出捕获为 bash 或 PowerShell 变量之类的方法来抑制命令输出。

以下是将输出（但不是错误）重定向到 bash 示例：/dev/null

```
$ aws s3 ls > /dev/null
```

有关抑制终端输出的详细信息，请参阅所用终端的用户文档。

- 考虑日志的访问权限，并根据您的用例适当确定访问范围。

## 服务器端与客户端输出选项

同时 AWS CLI 具有[服务器端和客户端筛选功能](#)，您可以单独或一起使用它们来 AWS CLI 筛选输出。首先处理服务器端筛选，然后返回输出以进行客户端筛选。服务器端筛选由服务 API 提供支持。使用 `--query` 参数的客户端支持 AWS CLI 客户端筛选。

服务器端输出选项是 AWS 服务 API 直接支持的功能。任何经过筛选或分页的数据都不会发送到客户端，这可以缩短 HTTP 响应时间，并为较大的数据集提高带宽。

客户端输出选项是由 AWS CLI 创建的功能。所有数据都发送到客户端，然后 AWS CLI 过滤器或页面显示内容。对于较大的数据集，客户端操作不会加快速度或节省带宽。

当服务器端和客户端选项同时使用时，服务器端操作会首先完成，然后发送到客户端进行客户端操作。这利用了服务器端选项可以加快速度和节省带宽的特点，同时使用其他 AWS CLI 功能来获得所需的输出。

## 设置 AWS CLI 输出格式

本主题介绍了 AWS Command Line Interface (AWS CLI) 的不同输出格式。AWS CLI 支持以下输出格式：

- **json** – 输出采用 [JSON](#) 字符串的格式。
- **yaml** – 输出采用 [YAML](#) 字符串的格式。

- **yaml-stream** – 输出被流式处理并采用 [YAML](#) 字符串的格式。串流支持更快地处理大型数据类型。
- **text** – 输出采用多个制表符分隔字符串值行的格式。这对于将输出传递到文本处理器（如 `grep`、`sed` 或 `awk`）很有用。
- **table** – 输出采用表格形式，使用字符 `+|-` 以形成单元格边框。它通常以“人性化”格式呈现信息，这种格式比其他格式更容易阅读，但从编程方面来讲不是那么有用。

## 如何选择输出格式

正如[配置](#)主题所述，输出格式可通过三种不同方式指定：

- 在 **config** 文件的命名配置文件中 使用 **output** 选项 – 以下示例将默认输出格式设置为 `text`。

```
[default]
output=text
```

- 使用 **AWS\_DEFAULT\_OUTPUT** 环境变量 – 对于此命令行会话中的命令，以下输出将格式设置为 `table`，直到更改此变量或会话结束。使用此环境变量将覆盖在 `config` 文件中设置的任何值。

```
$ export AWS_DEFAULT_OUTPUT="table"
```

- 在命令行上使用 **--output** 选项 – 以下示例仅将这一个命令的输出设置为 `json`。对此命令使用此选项将覆盖任何当前设置的环境变量或 `config` 文件中的值。

```
$ aws swf list-domains --registration-status REGISTERED --output json
```

### Important

指定的输出类型更改 `--query` 选项的运行方式：

- 如果您指定 `--output text`，则在应用 `--query` 筛选条件之前，输出采用分页方式，并且 AWS CLI 会一次性地在输出的每个页面上运行查询。因此，查询在每个页面上包括第一个匹配元素，这可能会导致意外的额外输出。要进一步筛选输出，您可以使用其他命令行工具，例如 `head` 或 `tail`。
- 如果您指定 `--output json`、`--output yaml` 或 `--output yaml-stream`，则在应用 `--query` 筛选条件之前，输出会完全处理为单个本机结构。AWS CLI 仅针对整个结构运行查询一次，同时生成筛选的结果，然后将结果输出。

## JSON 输出格式

[JSON](#) 是 AWS CLI 的默认输出格式。大多数编程语言可以使用内置函数或公开提供的库轻松解码 JSON 字符串。您可以通过强有力的方式将 JSON 输出与 [--query 选项](#) 结合使用，以筛选和格式化 AWS CLI JSON 格式的输出。

对于您可能无法使用 `--query` 执行的更高级的筛选，可以考虑使用 `jq`，这是一个命令行 JSON 处理器。您可以在以下网址下载它并找到正式的教程：<http://stedolan.github.io/jq/>。

以下是 JSON 输出的示例。

```
$ aws iam list-users --output json
```

```
{
  "Users": [
    {
      "Path": "/",
      "UserName": "Admin",
      "UserId": "AIDA111111111111EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:user/Admin",
      "CreateDate": "2014-10-16T16:03:09+00:00",
      "PasswordLastUsed": "2016-06-03T18:37:29+00:00"
    },
    {
      "Path": "/backup/",
      "UserName": "backup-user",
      "UserId": "AIDA222222222222EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:user/backup/backup-user",
      "CreateDate": "2019-09-17T19:30:40+00:00"
    },
    {
      "Path": "/",
      "UserName": "cli-user",
      "UserId": "AIDA333333333333EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:user/cli-user",
      "CreateDate": "2019-09-17T19:11:39+00:00"
    }
  ]
}
```

## YAML 输出格式

对于使用某些服务和工具（这些工具发出或使用 [YAML](#) 格式的字符串）以编程方式处理输出的情况，如 AWS CloudFormation 及其对 [YAML 格式的模板](#) 的支持，[YAML](#) 是一个很好的选择。

对于您可能无法使用 `--query` 执行的更高级的筛选，可以考虑使用 `yq`，这是一个命令行 YAML 处理器。您可以从 GitHub 上的 [yq 存储库](#) 下载 `yq`。

以下是 YAML 输出的示例。

```
$ aws iam list-users --output yaml
```

```
Users:
- Arn: arn:aws:iam::123456789012:user/Admin
  CreateDate: '2014-10-16T16:03:09+00:00'
  PasswordLastUsed: '2016-06-03T18:37:29+00:00'
  Path: /
  UserId: AIDA111111111111EXAMPLE
  UserName: Admin
- Arn: arn:aws:iam::123456789012:user/backup/backup-user
  CreateDate: '2019-09-17T19:30:40+00:00'
  Path: /backup/
  UserId: AIDA222222222222EXAMPLE
  UserName: arq-45EFD6D1-CE56-459B-B39F-F9C1F78FBE19
- Arn: arn:aws:iam::123456789012:user/cli-user
  CreateDate: '2019-09-17T19:30:40+00:00'
  Path: /
  UserId: AIDA333333333333EXAMPLE
  UserName: cli-user
```

## YAML 流输出格式

`yaml-stream` 格式利用 [YAML](#) 格式，同时通过流式传输数据给您，使您在查看大型数据集时的响应能力更高/速度更快。您可以在整个查询下载之前开始查看和使用 YAML 数据。

对于您可能无法使用 `--query` 执行的更高级的筛选，可以考虑使用 `yq`，这是一个命令行 YAML 处理器。您可以从 GitHub 上的 [yq 存储库](#) 下载 `yq`。

下面是 `yaml-stream` 输出的一个示例。

```
$ aws iam list-users --output yaml-stream
```

```

- IsTruncated: false
  Users:
  - Arn: arn:aws:iam::123456789012:user/Admin
    CreateDate: '2014-10-16T16:03:09+00:00'
    PasswordLastUsed: '2016-06-03T18:37:29+00:00'
    Path: /
    UserId: AIDA111111111111EXAMPLE
    UserName: Admin
  - Arn: arn:aws:iam::123456789012:user/backup/backup-user
    CreateDate: '2019-09-17T19:30:40+00:00'
    Path: /backup/
    UserId: AIDA222222222222EXAMPLE
    UserName: arq-45EFD6D1-CE56-459B-B39F-F9C1F78FBE19
  - Arn: arn:aws:iam::123456789012:user/cli-user
    CreateDate: '2019-09-17T19:30:40+00:00'
    Path: /
    UserId: AIDA333333333333EXAMPLE
    UserName: cli-user

```

下面是一个 `yaml-stream` 输出示例，并且使用 `--page-size` 参数对流式 YAML 内容进行分页。

```
$ aws iam list-users --output yaml-stream --page-size 2
```

```

- IsTruncated: true
  Marker: ab1234cdef5ghi67jk8lmo9p/
q012rs3t445uv6789w0x1y2z/345a6b78c9d00/1efgh234ij56klmno78pqrstu90vwxyx
  Users:
  - Arn: arn:aws:iam::123456789012:user/Admin
    CreateDate: '2014-10-16T16:03:09+00:00'
    PasswordLastUsed: '2016-06-03T18:37:29+00:00'
    Path: /
    UserId: AIDA111111111111EXAMPLE
    UserName: Admin
  - Arn: arn:aws:iam::123456789012:user/backup/backup-user
    CreateDate: '2019-09-17T19:30:40+00:00'
    Path: /backup/
    UserId: AIDA222222222222EXAMPLE
    UserName: arq-45EFD6D1-CE56-459B-B39F-F9C1F78FBE19
- IsTruncated: false
  Users:
  - Arn: arn:aws:iam::123456789012:user/cli-user
    CreateDate: '2019-09-17T19:30:40+00:00'

```

```
Path: /
UserId: AIDA333333333333EXAMPLE
UserName: cli-user
```

## 文本输出格式

`text` 格式将 AWS CLI 输出组织为制表符分隔的行。此格式适合在传统 Unix 文本工具 ( 如 `grep`、`sed` 和 `awk` ) 和由 PowerShell 执行的文本处理中使用。

`text` 输出格式遵循以下所示的基本结构。这些列根据底层 JSON 对象相应的键名称按字母顺序排序。

```
IDENTIFIER sorted-column1 sorted-column2
IDENTIFIER2 sorted-column1 sorted-column2
```

下面是 `text` 输出的一个示例。每个字段都用标签与其他字段分开，并带有一个额外的标签，其中有一个空字段。

```
$ aws iam list-users --output text
```

```
USERS arn:aws:iam::123456789012:user/Admin 2014-10-16T16:03:09+00:00
2016-06-03T18:37:29+00:00 / AIDA111111111111EXAMPLE Admin
USERS arn:aws:iam::123456789012:user/backup/backup-user 2019-09-17T19:30:40+00:00
/backup/ AIDA222222222222EXAMPLE backup-user
USERS arn:aws:iam::123456789012:user/cli-user 2019-09-17T19:11:39+00:00
/ AIDA333333333333EXAMPLE cli-user
```

第四列是 `PasswordLastUsed` 字段，它对于最后两个条目为空，因为这些用户从未登录过 AWS Management Console。

### Important

我们强烈建议，如果您指定 `text` 输出，则也始终使用 `--query` 选项以确保行为一致。这是因为文本格式按由 AWS 服务返回的基础 JSON 对象的键名称以字母顺序对输出列进行排序，而类似的资源可能没有相同的键名称。例如，基于 Linux 的 Amazon EC2 实例的 JSON 表示形式中的元素在基于 Windows 的实例的 JSON 表示形式中可能不存在，反之亦然。此外，资源可能在未来的更新中添加或移除键/值元素，从而修改列的顺序。因此，可以使用 `--query` 补充 `text` 输出的功能，以提供对输出格式的完全控制。

在以下示例中，命令指定要显示的元素，并使用列表表示法 `[key1, key2, ...]` 来定义列的顺序。这可让您十分放心：正确的键值始终显示在预期的列中。最后请注意，对于不存在的键，AWS CLI 将输出 `None` 作为键值。



```
$ aws iam list-users --output text --query 'Users[*].
[UserName,Arn,CreateDate,PasswordLastUsed,UserId]'
```

```
Admin          arn:aws:iam::123456789012:user/Admin
2014-10-16T16:03:09+00:00  2016-06-03T18:37:29+00:00  AIDA111111111111EXAMPLE
backup-user    arn:aws:iam::123456789012:user/backup-user
2019-09-17T19:30:40+00:00  None                        AIDA222222222222EXAMPLE
cli-user       arn:aws:iam::123456789012:user/cli-backup
2019-09-17T19:11:39+00:00  None                        AIDA333333333333EXAMPLE
```

以下示例显示如何将 `grep` 和 `awk` 与来自 `text` 命令的 `aws ec2 describe-instances` 输出结合使用。第一个命令在 `text` 输出中显示每个实例的可用区、当前状态和实例 ID。第二个命令处理该输出，以仅输出在 `us-west-2a` 可用区中运行的所有实例的实例 ID。

```
$ aws ec2 describe-instances --query 'Reservations[*].Instances[*].
[Placement.AvailabilityZone, State.Name, InstanceId]' --output text
```

```
us-west-2a    running i-4b41a37c
us-west-2a    stopped i-a071c394
us-west-2b    stopped i-97a217a0
us-west-2a    running i-3045b007
us-west-2a    running i-6fc67758
```

```
$ aws ec2 describe-instances --query 'Reservations[*].Instances[*].
[Placement.AvailabilityZone, State.Name, InstanceId]' --output text | grep us-west-2a |
grep running | awk '{print $3}'
```

```
i-4b41a37c
i-3045b007
i-6fc67758
```

以下示例更进一步，不仅说明如何筛选输出，还介绍如何使用该输出自动更改每个已停止实例的实例类型。

```
$ aws ec2 describe-instances --query 'Reservations[*].Instances[*].[State.Name,
InstanceId]' --output text |
```

```
> grep stopped |
> awk '{print $2}' |
> while read line;
> do aws ec2 modify-instance-attribute --instance-id $line --instance-type '{"Value":
"m1.medium"}';
> done
```

text 输出也适用于 PowerShell。因为 text 输出中的列使用制表符分隔，所以可以轻松地使用 PowerShell 的 `t` 分隔符将输出拆分为数组。以下命令在第一列 (InstanceId) 与字符串 AvailabilityZone 匹配的情况下显示第三列 (us-west-2a) 的值。

```
PS C:\>aws ec2 describe-instances --query 'Reservations[*].Instances[*].
[Placement.AvailabilityZone, State.Name, InstanceId]' --output text |
%{if ($_.split("`t")[0] -match "us-west-2a") { $_.split("`t")[2]; } }
```

```
-4b41a37c
i-a071c394
i-3045b007
i-6fc67758
```

注意，尽管上一个示例的确显示了如何使用 `--query` 参数解析底层 JSON 对象和提取所需的列，但是 PowerShell 有自己处理 JSON 的功能（如果跨平台兼容性不是问题）。不像大多数命令 shell 需要的那样将输出处理为文本，PowerShell 让您可以使用 `ConvertFrom-Json` cmdlet 生成具有层次结构的对象。然后，您可以直接从该对象访问所需的成员。

```
(aws ec2 describe-instances --output json | ConvertFrom-
Json).Reservations.Instances.InstanceId
```

### Tip

如果使用 `--query` 参数输出文本并将输出筛选到单个字段，则输出是单行制表符分隔值。要将每个值放到单独的行上，可以将输出字段放在括号中，如以下示例所示。

制表符分隔的单行输出：

```
$ aws iam list-groups-for-user --user-name susan --output text --query
"Groups[].GroupName"
```

```
HRDepartment    Developers    SpreadsheetUsers    LocalAdmins
```



您可以将 `--query` 选项与 `table` 格式结合使用，以显示从原始输出中预先选择的一系列元素。请注意字典和列表表示法之间的输出区别：在第一个示例中，列名按字母顺序排序；在第二个示例中，未命名的列按用户指定的顺序排序。有关 `--query` 选项的更多信息，请参阅 [过滤器 AWS CLI 输出](#)。

```
$ aws ec2 describe-volumes --query 'Volumes[*].
{ID:VolumeId,InstanceId:Attachments[0].InstanceId,AZ:AvailabilityZone,Size:Size}' --
output table
```

```
-----
|                               DescribeVolumes                               |
+-----+-----+-----+-----+
|   AZ   |   ID   | InstanceId | Size |
+-----+-----+-----+-----+
| us-west-2a | vol-e11a5288 | i-a071c394 | 30 |
| us-west-2a | vol-2e410a47 | i-4b41a37c | 8  |
+-----+-----+-----+-----+
```

```
$ aws ec2 describe-volumes --query 'Volumes[*].
[VolumeId,Attachments[0].InstanceId,AvailabilityZone,Size]' --output table
```

```
-----
|                               DescribeVolumes                               |
+-----+-----+-----+-----+
| vol-e11a5288 | i-a071c394 | us-west-2a | 30 |
| vol-2e410a47 | i-4b41a37c | us-west-2a | 8  |
+-----+-----+-----+-----+
```

## 使用 AWS CLI 分页选项

本主题介绍分页 AWS CLI 的输出的不同方式。

主要有两种方法可以从 AWS CLI 控制分页。

- [使用服务器端分页参数。](#)
- [使用默认输出客户端分页程序。](#)

服务器端分页参数会先进行处理，然后系统会将所有输出发送到客户端分页程序进行处理。

## 服务器端分页

对于可返回项目大型列表的命令，AWS Command Line Interface (AWS CLI) 提供了多个选项。当 AWS CLI 调用服务 API 以填充此列表时，您可以使用这些选项控制输出中包括的项目数。

这些选项包含以下内容：

- [如何使用 `--no-paginate` 参数](#)
- [如何使用 `--page-size` 参数](#)
- [如何使用 `--max-items` 参数](#)
- [如何使用 `--starting-token` 参数](#)

默认情况下，AWS CLI 使用由单个服务确定的页面大小并检索所有可用项目。例如，Amazon S3 的原定设置页面大小为 1000。如果您在包含 3500 个对象的 Amazon S3 存储桶上运行 `aws s3api list-objects`，则 AWS CLI 将自动对 Amazon S3 发出四次调用，以在后台处理服务特定分页逻辑并在最终输出中返回所有 3500 个对象。

### 如何使用 `--no-paginate` 参数

`--no-paginate` 选项在客户端禁用以下分页标记。使用命令时，默认情况下 AWS CLI 会自动执行多次调用，返回所有可能的结果来创建分页。每页显示一次调用的结果。禁用分页时，AWS CLI 只执行一次调用，显示命令结果的第一页。

例如，如果您在包含 3500 个对象的 Amazon S3 存储桶上运行 `aws s3api list-objects`，则 AWS CLI 只会执行对 Amazon S3 的第一个调用，并在最终输出中仅返回前 1000 个对象。

```
$ aws s3api list-objects \  
  --bucket my-bucket \  
  --no-paginate  
{  
  "Contents": [  
  ...
```

### 如何使用 `--page-size` 参数

如果您在对大量资源运行列表命令时发现问题，则表明原定设置页面大小可能过大。这可能会导致对 AWS 服务的调用超过允许的最大时间并生成“超时”错误。您可以使用 `--page-size` 选项来指定 AWS CLI 从对 AWS 服务的每个调用请求数量较少的项目。AWS CLI 仍将检索完整列表，但会在后台执行大量服务 API 调用，并减少每次调用时检索的项目数。这样，各个调用成功的可能性会更高且不会发生超时。更改页面大小不会影响输出；它只影响生成输出所需进行的 API 调用数。

```
$ aws s3api list-objects \  
  --bucket my-bucket \  
  --page-size 100  
{  
  "Contents": [  
  ...
```

### 如何使用 --max-items 参数

要在 AWS CLI 输出中一次包括更少的项目，请使用 `--max-items` 选项。AWS CLI 仍会按之前所述使用该服务处理分页，但只打印您指定的一次检索的项目数。

```
$ aws s3api list-objects \  
  --bucket my-bucket \  
  --max-items 100  
{  
  "NextToken": "eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ==",  
  "Contents": [  
  ...
```

### 如何使用 --starting-token 参数

如果项目输出的数量 (`--max-items`) 少于基础 API 调用所返回的项目总数，则输出将包含您可传递到后续命令的 `NextToken` 以检索下一组项目。以下示例显示如何使用上一示例返回的 `NextToken` 值，并使您能够检索接下来的 100 个项目。

#### Note

参数 `--starting-token` 不能为空。如果上一个命令未返回 `NextToken` 值，则没有更多项目可返回，您不需要再次调用该命令。

```
$ aws s3api list-objects \  
  --bucket my-bucket \  
  --max-items 100 \  
  --starting-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ==  
{  
  "Contents": [  
  ...
```

每次调用指定的 AWS 服务时返回项目的顺序可能不同。如果您为 `--page-size` 和 `--max-items` 指定不同的值，您可能会获得意外结果（项目缺失或重复）。为防止出现这种情况，请对 `--page-size` 和 `--max-items` 使用相同的数字，以同步 AWS CLI 的分页与基础服务的分页。您还可以检索整个列表并在本地执行任何必需的分页操作。

## 客户端分页程序

AWS CLI 版本 2 支持使用客户端分页程序对输出进行处理。默认情况下，此功能会通过操作系统的默认分页程序返回所有输出。

您可以按照优先级顺序，通过以下方式指定输出分页程序：

- 使用 `default` 或所指定配置文件的 `config` 文件中的 `cli_pager` 设置。
- 使用 `AWS_PAGER` 环境变量。
- 使用 `PAGER` 环境变量。

您可以按照优先级顺序，通过以下方式完全禁止对外部分页程序的使用：

- 使用 `--no-cli-pager` 命令行选项可禁止在单个命令中使用分页程序。
- 将 `cli_pager` 设置或 `AWS_PAGER` 变量设置为空字符串。

客户端分页主题：

- [如何使用 `cli\_pager` 设置](#)
- [如何使用 `AWS\_PAGER` 环境变量](#)
- [如何使用 `--no-cli-pager` 选项](#)
- [如何使用分页程序标志](#)

## 如何使用 `cli_pager` 设置

您可以将常用的配置设置和凭证保存在由 AWS CLI 维护的文件中。所指定配置文件中的设置优先于 `default` 配置文件中的设置。有关配置设置的更多信息，请参阅[配置和凭证文件设置](#)。

以下示例将默认输出分页程序设置为 `less` 程序。

```
[default]
cli_pager=less
```

以下示例将默认值设置为禁用分页程序。

```
[default]
cli_pager=
```

如何使用 AWS\_PAGER 环境变量

以下示例将默认输出分页程序设置为 less 程序。有关环境变量的更多信息，请参阅[用于配置的环境变量 AWS CLI](#)。

Linux and macOS

```
$ export AWS_PAGER="less"
```

Windows

```
C:\> setx AWS_PAGER "less"
```

以下示例禁用分页程序。

Linux and macOS

```
$ export AWS_PAGER=""
```

Windows

```
C:\> setx AWS_PAGER ""
```

如何使用 --no-cli-pager 选项

要对单个命令禁止使用分页程序，请使用 --no-cli-pager 选项。有关命令行选项的更多信息，请参阅[命令行选项](#)。

```
$ aws s3api list-objects \
  --bucket my-bucket \
  --no-cli-pager
{
  "Contents": [
```



```
...
```

## 如何使用分页程序标志

您可以指定要对分页程序自动使用的标志。标志取决于您使用的分页程序。以下示例适用于 `less` 和 `more` 的典型默认值。

### Linux and macOS

如果不另行指定，则 AWS CLI 版本 2 在预设情况下使用的分页程序为 `less`。如果您没有设置 `LESS` 环境变量，则 AWS CLI 版本 2 使用 `FRX` 标志。您可以通过在设置 AWS CLI 分页程序时指定标志，来结合使用各个标志。

以下示例使用 `S` 标志。然后，此标志与默认 `FRX` 标志结合使用，创建最终 `FRXS` 标志。

```
$ export AWS_PAGER="less -S"
```

如果您不希望使用任何 `FRX` 标志，您可以对它们取否定值。以下示例对 `F` 标志取否定值，创建最终 `RX` 标志。

```
$ export AWS_PAGER="less -+F"
```

有关 `less` 标志的更多信息，请参阅 [manpages.org](http://manpages.org) 上的 [less](http://manpages.org/less)。

### Windows

如果不另行指定，则 AWS CLI 版本 2 默认使用的分页程序为不带附加标志的 `more`。

以下示例使用 `/c` 参数。

```
C:\> setx AWS_PAGER "more /c"
```

有关 `more` 标志的更多信息，请参阅 Microsoft 文档上的 [更多](#)。

## 过滤器 AWS CLI 输出

AWS Command Line Interface (AWS CLI) 同时具有服务器端和客户端筛选功能，您可以单独或一起使用它们来 AWS CLI 筛选输出。首先处理服务器端筛选，然后返回输出以进行客户端筛选。

- 服务器端筛选由 API 提供支持，通常使用 `--filter` 参数来实现筛选。该服务仅返回匹配结果，这可以加快大型数据集的 HTTP 响应时间。

- 使用 `--query` 参数的客户端支持 AWS CLI 客户端筛选。此参数具有服务器端筛选可能没有的功能。

## 主题

- [服务器端筛选](#)
- [客户端筛选](#)
- [结合服务器端和客户端筛选](#)
- [其他资源](#)

## 服务器端筛选

中的服务器端筛选由 AWS 服务 API 提供。AWS CLI 该 AWS 服务仅返回 HTTP 响应中与筛选条件匹配的记录，这可以加快大型数据集的 HTTP 响应时间。由于服务器端筛选由服务 API 定义，因此参数名称和函数因服务而异。用于筛选的一些常见参数名称包括：

- `--filter` 例如 [ses](#) 和 [ce](#)。
- `--filters` 例如 [ec2](#)、[autoscaling](#) 和 [rds](#)。
- 以单词 `filter` 开头的名称，例如 `--filter-expression` 用于 [aws dynamodb scan](#) 命令。

有关特定命令是否具有服务器端筛选和筛选规则的信息，请参阅[版本 2 参考指南](#)。

## 客户端筛选

通过参数 AWS CLI 提供了基于 JSON 的内置客户端筛选功能。`--query` 该 `--query` 参数是一个功能强大的工具，可用来自定义输出的内容和样式。该 `--query` 参数可获取从服务器返回的 HTTP 响应，并在显示结果之前对结果进行筛选。由于在筛选之前将整个 HTTP 响应发送到客户端，因此客户端筛选速度可能比大型数据集的服务器端筛选更慢。

查询使用 [JMESPath 语法](#) 来创建用于筛选输出的表达式。要了解 JMESPath 语法，请参阅 JMESPath 网站上的[教程](#)。

### Important

指定的输出类型更改 `--query` 选项的运行方式：

- 如果您指定 `--output text`，则会在应用 `--query` 过滤器之前对输出进行分页，并且会在输出的每一页上 AWS CLI 运行一次查询。因此，查询在每个页面上包括第一个匹配元

素，这可能会导致意外的额外输出。要进一步筛选输出，您可以使用其他命令行工具，例如 `head` 或 `tail`。

- 如果您指定 `--output json`、`--output yaml` 或 `--output yaml-stream`，则在应用 `--query` 筛选条件之前，输出会完全处理为单个本机结构。只对整个结构 AWS CLI 运行一次查询，生成过滤后的结果然后输出。

## 客户端筛选主题

- [在您开始之前](#)
- [标识符](#)
- [从列表中选择](#)
- [筛选嵌套数据](#)
- [展平结果](#)
- [筛选特定值](#)
- [传输表达式](#)
- [筛选多个标识符值](#)
- [将标签添加到标识符值](#)
- [函数](#)
- [高级 `--query` 示例](#)

## 在您开始之前

使用这些示例中使用的筛选条件表达式时，请务必为终端 shell 使用正确的引用规则。有关更多信息，请参阅 [the section called “含字符串的引号”](#)。

以下 JSON 输出显示了 `--query` 参数可以生成的示例。此输出描述了附加到单独 Amazon EC2 实例的三个 Amazon EBS 卷。

## 示例输出

```
$ aws ec2 describe-volumes
{
  "Volumes": [
    {
      "AvailabilityZone": "us-west-2a",
      "Attachments": [
```

```
{
  "AttachTime": "2013-09-17T00:55:03.000Z",
  "InstanceId": "i-a071c394",
  "VolumeId": "vol-e11a5288",
  "State": "attached",
  "DeleteOnTermination": true,
  "Device": "/dev/sda1"
}
],
"VolumeType": "standard",
"VolumeId": "vol-e11a5288",
"State": "in-use",
"SnapshotId": "snap-f23ec1c8",
"CreateTime": "2013-09-17T00:55:03.000Z",
"Size": 30
},
{
  "AvailabilityZone": "us-west-2a",
  "Attachments": [
    {
      "AttachTime": "2013-09-18T20:26:16.000Z",
      "InstanceId": "i-4b41a37c",
      "VolumeId": "vol-2e410a47",
      "State": "attached",
      "DeleteOnTermination": true,
      "Device": "/dev/sda1"
    }
  ],
  "VolumeType": "standard",
  "VolumeId": "vol-2e410a47",
  "State": "in-use",
  "SnapshotId": "snap-708e8348",
  "CreateTime": "2013-09-18T20:26:15.000Z",
  "Size": 8
},
{
  "AvailabilityZone": "us-west-2a",
  "Attachments": [
    {
      "AttachTime": "2020-11-20T19:54:06.000Z",
      "InstanceId": "i-1jd73kv8",
      "VolumeId": "vol-a1b3c7nd",
      "State": "attached",
      "DeleteOnTermination": true,
```

```
    "Device": "/dev/sda1"
  }
],
"VolumeType": "standard",
"VolumeId": "vol-a1b3c7nd",
"State": "in-use",
"SnapshotId": "snap-234087fb",
"CreateTime": "2020-11-20T19:54:05.000Z",
"Size": 15
}
]
}
```

## 标识符

标识符是输出值的标签。创建筛选条件时，您可以使用标识符缩小查询结果的范围。在以下输出示例中，所有标识符（如 `Volumes`、`AvailabilityZone` 和 `AttachTime`）都将突出显示。

```
$ aws ec2 describe-volumes
{
  "Volumes": [
    {
      "AvailabilityZone": "us-west-2a",
      "Attachments": [
        {
          "AttachTime": "2013-09-17T00:55:03.000Z",
          "InstanceId": "i-a071c394",
          "VolumeId": "vol-e11a5288",
          "State": "attached",
          "DeleteOnTermination": true,
          "Device": "/dev/sda1"
        }
      ],
      "VolumeType": "standard",
      "VolumeId": "vol-e11a5288",
      "State": "in-use",
      "SnapshotId": "snap-f23ec1c8",
      "CreateTime": "2013-09-17T00:55:03.000Z",
      "Size": 30
    },
    {
      "AvailabilityZone": "us-west-2a",
      "Attachments": [
```

```

    {
      "AttachTime": "2013-09-18T20:26:16.000Z",
      "InstanceId": "i-4b41a37c",
      "VolumeId": "vol-2e410a47",
      "State": "attached",
      "DeleteOnTermination": true,
      "Device": "/dev/sda1"
    }
  ],
  "VolumeType": "standard",
  "VolumeId": "vol-2e410a47",
  "State": "in-use",
  "SnapshotId": "snap-708e8348",
  "CreateTime": "2013-09-18T20:26:15.000Z",
  "Size": 8
},
{
  "AvailabilityZone": "us-west-2a",
  "Attachments": [
    {
      "AttachTime": "2020-11-20T19:54:06.000Z",
      "InstanceId": "i-1jd73kv8",
      "VolumeId": "vol-a1b3c7nd",
      "State": "attached",
      "DeleteOnTermination": true,
      "Device": "/dev/sda1"
    }
  ],
  "VolumeType": "standard",
  "VolumeId": "vol-a1b3c7nd",
  "State": "in-use",
  "SnapshotId": "snap-234087fb",
  "CreateTime": "2020-11-20T19:54:05.000Z",
  "Size": 15
}
]
}

```

有关更多信息，请参阅 JMESPath 网站上的[标识符](#)。

从列表中选择

列表或数组是后跟方括号“[]”的标识符，例如 Volumes 中的 Attachments 和 [the section called “在您开始之前”](#)。

## 语法

```
<listName>[ ]
```

要筛选数组的所有输出，可以使用通配符表示法。[通配符](#)表达式是用于使用 \* 表示法返回元素的表达式。

以下示例查询所有 Volumes 内容。

```
$ aws ec2 describe-volumes \
  --query 'Volumes[*]'
[
  {
    "AvailabilityZone": "us-west-2a",
    "Attachments": [
      {
        "AttachTime": "2013-09-17T00:55:03.000Z",
        "InstanceId": "i-a071c394",
        "VolumeId": "vol-e11a5288",
        "State": "attached",
        "DeleteOnTermination": true,
        "Device": "/dev/sda1"
      }
    ],
    "VolumeType": "standard",
    "VolumeId": "vol-e11a5288",
    "State": "in-use",
    "SnapshotId": "snap-f23ec1c8",
    "CreateTime": "2013-09-17T00:55:03.000Z",
    "Size": 30
  },
  {
    "AvailabilityZone": "us-west-2a",
    "Attachments": [
      {
        "AttachTime": "2020-11-20T19:54:06.000Z",
        "InstanceId": "i-1jd73kv8",
        "VolumeId": "vol-a1b3c7nd",
        "State": "attached",
        "DeleteOnTermination": true,
        "Device": "/dev/sda1"
      }
    ],
  },
]
```

```

    "VolumeType": "standard",
    "VolumeId": "vol-a1b3c7nd",
    "State": "in-use",
    "SnapshotId": "snap-234087fb",
    "CreateTime": "2020-11-20T19:54:05.000Z",
    "Size": 15
  }
]

```

要按索引查看数组中的特定卷，请调用数组索引。例如，Volumes 数组中的第一个项目的索引为 0，返回 Volumes[0] 查询。有关数组索引的更多信息，请参阅 JMESPath 网站上的[索引表达式](#)。

```

$ aws ec2 describe-volumes \
  --query 'Volumes[0]'
{
  "AvailabilityZone": "us-west-2a",
  "Attachments": [
    {
      "AttachTime": "2013-09-17T00:55:03.000Z",
      "InstanceId": "i-a071c394",
      "VolumeId": "vol-e11a5288",
      "State": "attached",
      "DeleteOnTermination": true,
      "Device": "/dev/sda1"
    }
  ],
  "VolumeType": "standard",
  "VolumeId": "vol-e11a5288",
  "State": "in-use",
  "SnapshotId": "snap-f23ec1c8",
  "CreateTime": "2013-09-17T00:55:03.000Z",
  "Size": 30
}

```

要按索引查看特定范围的卷，请使用 slice 与以下语法，其中 start 是起始数组索引，stop 是筛选条件停止处理的索引，step 是跳过间隔。

## 语法

```
<arrayName>[<start>:<stop>:<step>]
```

如果 Slice 表达式中忽略了其中任何一个，它们将使用以下默认值：



- Start – 列表中的第一个索引，0。
- Stop – 列表中的最后一个索引。
- Step – 没有跳过步骤，其中值为 1。

要仅返回前两个卷，请使用起始值 0、停止值 2 和步长值 1，如以下示例所示。

```
$ aws ec2 describe-volumes \
  --query 'Volumes[0:2:1]'
[
  {
    "AvailabilityZone": "us-west-2a",
    "Attachments": [
      {
        "AttachTime": "2013-09-17T00:55:03.000Z",
        "InstanceId": "i-a071c394",
        "VolumeId": "vol-e11a5288",
        "State": "attached",
        "DeleteOnTermination": true,
        "Device": "/dev/sda1"
      }
    ],
    "VolumeType": "standard",
    "VolumeId": "vol-e11a5288",
    "State": "in-use",
    "SnapshotId": "snap-f23ec1c8",
    "CreateTime": "2013-09-17T00:55:03.000Z",
    "Size": 30
  },
  {
    "AvailabilityZone": "us-west-2a",
    "Attachments": [
      {
        "AttachTime": "2013-09-18T20:26:16.000Z",
        "InstanceId": "i-4b41a37c",
        "VolumeId": "vol-2e410a47",
        "State": "attached",
        "DeleteOnTermination": true,
        "Device": "/dev/sda1"
      }
    ],
    "VolumeType": "standard",
    "VolumeId": "vol-2e410a47",
```

```
"State": "in-use",
"SnapshotId": "snap-708e8348",
"CreateTime": "2013-09-18T20:26:15.000Z",
"Size": 8
}
]
```

由于此示例包含默认值，因此您可以将 Slice 从 `Volumes[0:2:1]` 缩短到 `Volumes[:2]`。

以下示例省略了默认值，并返回整个数组中的每两个卷。

```
$ aws ec2 describe-volumes \
  --query 'Volumes[:2]'
[
  {
    "AvailabilityZone": "us-west-2a",
    "Attachments": [
      {
        "AttachTime": "2013-09-17T00:55:03.000Z",
        "InstanceId": "i-a071c394",
        "VolumeId": "vol-e11a5288",
        "State": "attached",
        "DeleteOnTermination": true,
        "Device": "/dev/sda1"
      }
    ],
    "VolumeType": "standard",
    "VolumeId": "vol-e11a5288",
    "State": "in-use",
    "SnapshotId": "snap-f23ec1c8",
    "CreateTime": "2013-09-17T00:55:03.000Z",
    "Size": 30
  },
  {
    "AvailabilityZone": "us-west-2a",
    "Attachments": [
      {
        "AttachTime": "2020-11-20T19:54:06.000Z",
        "InstanceId": "i-1jd73kv8",
        "VolumeId": "vol-a1b3c7nd",
        "State": "attached",
        "DeleteOnTermination": true,
        "Device": "/dev/sda1"
      }
    ]
  }
]
```

```

    ],
    "VolumeType": "standard",
    "VolumeId": "vol-a1b3c7nd",
    "State": "in-use",
    "SnapshotId": "snap-234087fb",
    "CreateTime": "2020-11-20T19:54:05.000Z",
    "Size": 15
  }
]

```

Steps 还可以使用负数按数组的相反顺序进行筛选，如以下示例所示。

```

$ aws ec2 describe-volumes \
  --query 'Volumes[::-2]'
[
  {
    "AvailabilityZone": "us-west-2a",
    "Attachments": [
      {
        "AttachTime": "2020-11-20T19:54:06.000Z",
        "InstanceId": "i-1jd73kv8",
        "VolumeId": "vol-a1b3c7nd",
        "State": "attached",
        "DeleteOnTermination": true,
        "Device": "/dev/sda1"
      }
    ],
    "VolumeType": "standard",
    "VolumeId": "vol-a1b3c7nd",
    "State": "in-use",
    "SnapshotId": "snap-234087fb",
    "CreateTime": "2020-11-20T19:54:05.000Z",
    "Size": 15
  },
  {
    "AvailabilityZone": "us-west-2a",
    "Attachments": [
      {
        "AttachTime": "2013-09-17T00:55:03.000Z",
        "InstanceId": "i-a071c394",
        "VolumeId": "vol-e11a5288",
        "State": "attached",
        "DeleteOnTermination": true,

```

```

    "Device": "/dev/sda1"
  }
],
"VolumeType": "standard",
"VolumeId": "vol-e11a5288",
"State": "in-use",
"SnapshotId": "snap-f23ec1c8",
"CreateTime": "2013-09-17T00:55:03.000Z",
"Size": 30
}
]

```

有关更多信息，请参阅 JMESPath 网站上的 [Slices](#)。

## 筛选嵌套数据

要缩小嵌套值 `Volumes[*]` 的筛选范围，您可以通过附加句点和筛选条件来使用子表达式。

## 语法

```
<expression>.<expression>
```

以下示例显示了所有卷的所有 Attachments 信息。

```

$ aws ec2 describe-volumes \
  --query 'Volumes[*].Attachments'
[
  [
    {
      "AttachTime": "2013-09-17T00:55:03.000Z",
      "InstanceId": "i-a071c394",
      "VolumeId": "vol-e11a5288",
      "State": "attached",
      "DeleteOnTermination": true,
      "Device": "/dev/sda1"
    }
  ],
  [
    {
      "AttachTime": "2013-09-18T20:26:16.000Z",
      "InstanceId": "i-4b41a37c",
      "VolumeId": "vol-2e410a47",
      "State": "attached",

```

```

    "DeleteOnTermination": true,
    "Device": "/dev/sda1"
  }
],
[
  {
    "AttachTime": "2020-11-20T19:54:06.000Z",
    "InstanceId": "i-1jd73kv8",
    "VolumeId": "vol-a1b3c7nd",
    "State": "attached",
    "DeleteOnTermination": true,
    "Device": "/dev/sda1"
  }
]
]
```

要进一步筛选嵌套值，请为每个嵌套标识符附加表达式。以下示例列出了所有 State 的 Volumes。

```

$ aws ec2 describe-volumes \
  --query 'Volumes[*].Attachments[*].State'
[
  [
    "attached"
  ],
  [
    "attached"
  ],
  [
    "attached"
  ]
]
```

### 展平结果

有关更多信息，请参阅 [SubExpressions](#) JMESPath 网站。

您可以通过删除导致 `Volumes[*].Attachments[*].State` 查询的通配符表示法来展平 `Volumes[*].Attachments[].State` 的结果。展平操作通常有助于提高结果的可读性。

```

$ aws ec2 describe-volumes \
  --query 'Volumes[*].Attachments[].State'
[
  "attached",
```

```
"attached",
"attached"
]
```

有关更多信息，请参阅 JMESPath 网站上的[展平](#)。

## 筛选特定值

要筛选列表中的特定值，可以使用筛选条件表达式，如以下语法所示。

## 语法

```
? <expression> <comparator> <expression>]
```

表达式比较器包括 ==、!=、<、<=、> 和 >=。以下示例为 VolumeIdsVolumes 中的所有 Attached 筛选 State。

```
$ aws ec2 describe-volumes \
  --query 'Volumes[*].Attachments[?State==`attached`].VolumeId'
[
  [
    "vol-e11a5288"
  ],
  [
    "vol-2e410a47"
  ],
  [
    "vol-a1b3c7nd"
  ]
]
```

然后可以将其展平，从而生成以下示例。

```
$ aws ec2 describe-volumes \
  --query 'Volumes[*].Attachments[?State==`attached`].VolumeId[]'
[
  "vol-e11a5288",
  "vol-2e410a47",
  "vol-a1b3c7nd"
]
```

以下示例筛选了尺寸小于 20 的所有 VolumeIds 的 Volumes。

```
$ aws ec2 describe-volumes \
  --query 'Volumes[?Size < `20`].VolumeId'
[
  "vol-2e410a47",
  "vol-a1b3c7nd"
]
```

有关更多信息，请参阅 JMESPath 网站上的[筛选表达式](#)。

## 传输表达式

您可以将筛选器的结果传输到新列表中，然后通过以下语法使用另一个表达式筛选结果：

## 语法

```
<expression> | <expression>]
```

以下示例获取 `Volumes[*].Attachments[].InstanceId` 表达式的筛选结果并在数组中输出第一个结果。

```
$ aws ec2 describe-volumes \
  --query 'Volumes[*].Attachments[].InstanceId | [0]'
"i-a071c394"
```

本示例首先通过以下表达式创建数组来实现此目的。

```
$ aws ec2 describe-volumes \
  --query 'Volumes[*].Attachments[].InstanceId'
"i-a071c394",
"i-4b41a37c",
"i-1jd73kv8"
```

然后返回该数组中的第一个元素。

```
"i-a071c394"
```

有关更多信息，请参阅 JMESPath 网站上的[传输表达式](#)。

## 筛选多个标识符值

要筛选多个标识符，您可以使用以下语法使用多选列表：

## 语法

```
<listName>[].[<expression>, <expression>]
```

在以下示例中，VolumeId 和 VolumeType 在 Volumes 列表中进行筛选，生成以下表达式。

```
$ aws ec2 describe-volumes \  
  --query 'Volumes[][VolumeId, VolumeType]'  
[  
  [  
    "vol-e11a5288",  
    "standard"  
  ],  
  [  
    "vol-2e410a47",  
    "standard"  
  ],  
  [  
    "vol-a1b3c7nd",  
    "standard"  
  ]  
]
```

要将嵌套数据添加到列表中，请添加另一个多选列表。下面的示例通过在嵌套 InstanceId 列表中筛选 State 和 Attachments 在上一个示例中进行了扩展。这将产生以下表达式。

```
$ aws ec2 describe-volumes \  
  --query 'Volumes[][VolumeId, VolumeType, Attachments[][InstanceId, State]]'  
[  
  [  
    "vol-e11a5288",  
    "standard",  
    [  
      [  
        "i-a071c394",  
        "attached"  
      ]  
    ]  
  ],  
  [  
    "vol-2e410a47",  
    "standard",  
    [  
      [  
        "i-a071c394",  
        "attached"  
      ]  
    ]  
  ]  
]
```



```
[
  [
    "i-4b41a37c",
    "attached"
  ]
],
[
  "vol-a1b3c7nd",
  "standard",
  [
    [
      "i-1jd73kv8",
      "attached"
    ]
  ]
]
```

为了更具可读性，请按以下示例所示展开表达式。

```
$ aws ec2 describe-volumes \
  --query 'Volumes[][VolumeId, VolumeType, Attachments[][InstanceId, State]][[]]'
```

```
[
  "vol-e11a5288",
  "standard",
  [
    "i-a071c394",
    "attached"
  ],
  "vol-2e410a47",
  "standard",
  [
    "i-4b41a37c",
    "attached"
  ],
  "vol-a1b3c7nd",
  "standard",
  [
    "i-1jd73kv8",
    "attached"
  ]
]
```

有关更多信息，请参阅 JMESPath 网站上的[多选列表](#)。

将标签添加到标识符值

为了使此输出更易于读取，请使用具有以下语法的多选哈希。

语法

```
<listName>[].{<label>: <expression>, <label>: <expression>}
```

您的标识符标签不必与标识符的名称相同。以下示例为 VolumeType 值使用标签 VolumeType。

```
$ aws ec2 describe-volumes \
  --query 'Volumes[].{VolumeType: VolumeType}'
[
  {
    "VolumeType": "standard",
  },
  {
    "VolumeType": "standard",
  },
  {
    "VolumeType": "standard",
  }
]
```

为简单起见，以下示例保留每个标签的标识符名称，并显示所有卷的 VolumeId、VolumeType、InstanceId 和 State：

```
$ aws ec2 describe-volumes \
  --query 'Volumes[].{VolumeId: VolumeId, VolumeType: VolumeType, InstanceId:
  Attachments[0].InstanceId, State: Attachments[0].State}'
[
  {
    "VolumeId": "vol-e11a5288",
    "VolumeType": "standard",
    "InstanceId": "i-a071c394",
    "State": "attached"
  },
  {
    "VolumeId": "vol-2e410a47",
    "VolumeType": "standard",
    "InstanceId": "i-4b41a37c",
  }
]
```

```

    "State": "attached"
  },
  {
    "VolumeId": "vol-a1b3c7nd",
    "VolumeType": "standard",
    "InstanceId": "i-1jd73kv8",
    "State": "attached"
  }
]

```

有关更多信息，请参阅 JMESPath 网站上的[多选哈希](#)。

## 函数

JMESPath 语法包含许多可用于查询的函数。有关 JMESPath 函数的信息，请参阅 JMESPath 网站上的[内置函数](#)。

为了演示如何将函数合并到查询中，以下示例使用了 `sort_by` 函数。该 `sort_by` 函数使用以下语法将表达式作为排序键对数组进行排序：

## 语法

```
sort_by(<listName>, <sort expression>)[].<expression>
```

以下示例使用先前的[多选哈希示例](#)并按照 VolumeId 对输出进行排序。

```

$ aws ec2 describe-volumes \
  --query 'sort_by(Volumes, &VolumeId)[].{VolumeId: VolumeId, VolumeType: VolumeType,
InstanceId: Attachments[0].InstanceId, State: Attachments[0].State}'
[
  {
    "VolumeId": "vol-2e410a47",
    "VolumeType": "standard",
    "InstanceId": "i-4b41a37c",
    "State": "attached"
  },
  {
    "VolumeId": "vol-a1b3c7nd",
    "VolumeType": "standard",
    "InstanceId": "i-1jd73kv8",
    "State": "attached"
  },
  {

```

```
"VolumeId": "vol-e11a5288",
"VolumeType": "standard",
"InstanceId": "i-a071c394",
"State": "attached"
}
]
```

有关更多信息，请参阅 JMESPath 网站上的[排序依据](#)。

## 高级 `--query` 示例

### 从特定项目中提取信息

以下示例使用 `--query` 参数在列表中查找特定的项目，然后提取该项目的信息。此示例列出了与指定的服务终端节点相关联的所有 `AvailabilityZones`。它从 `ServiceDetails` 列表中提取具有指定 `ServiceName` 的项目，然后输出该选定项目的 `AvailabilityZones` 字段。

```
$ aws --region us-east-1 ec2 describe-vpc-endpoint-services \
  --query 'ServiceDetails[?ServiceName==`com.amazonaws.us-
east-1.ecs`].AvailabilityZones'
[
  [
    "us-east-1a",
    "us-east-1b",
    "us-east-1c",
    "us-east-1d",
    "us-east-1e",
    "us-east-1f"
  ]
]
```

### 在指定创建日期之后显示快照

以下示例显示如何列出在指定日期之后创建的所有快照，从而在输出中仅包括几个可用字段。

```
$ aws ec2 describe-snapshots --owner self \
  --output json \
  --query 'Snapshots[?StartTime>=`2018-02-07`].
{Id:SnapshotId,VId:VolumeId,Size:VolumeSize}'
[
  {
    "id": "snap-0effb42b7a1b2c3d4",
    "vid": "vol-0be9bb0bf12345678",
```

```
    "Size": 8
  }
]
```

## 显示最新 AMI

以下示例列出了您创建五个最新 Amazon 系统映像 (AMI)，从最新到最旧排序。

```
$ aws ec2 describe-images \
  --owners self \
  --query 'reverse(sort_by(Images,&CreationDate))[:5].{id:ImageId,date:CreationDate}'
[
  {
    "id": "ami-0a1b2c3d4e5f60001",
    "date": "2018-11-28T17:16:38.000Z"
  },
  {
    "id": "ami-0a1b2c3d4e5f60002",
    "date": "2018-09-15T13:51:22.000Z"
  },
  {
    "id": "ami-0a1b2c3d4e5f60003",
    "date": "2018-08-19T10:22:45.000Z"
  },
  {
    "id": "ami-0a1b2c3d4e5f60004",
    "date": "2018-05-03T12:04:02.000Z"
  },
  {
    "id": "ami-0a1b2c3d4e5f60005",
    "date": "2017-12-13T17:16:38.000Z"
  }
]
```

## 显示运行状况不佳的 Auto Scaling 实例

以下示例仅显示指定 Auto Scaling 组中任何运行状况不佳的实例的 InstanceId。

```
$ aws autoscaling describe-auto-scaling-groups \
  --auto-scaling-group-name My-AutoScaling-Group-Name \
  --output text \
  --query 'AutoScalingGroups[*].Instances[?HealthStatus==`Unhealthy`].InstanceId'
```

## 包括带指定标签的卷

以下示例描述了所有带 `test` 标签的实例。只要附加的卷旁边 `test` 还有另一个标签，那么结果中仍会返回该卷。

下面的表达式在数组中返回带 `test` 标签的所有标签。任何不是 `test` 标签的标签都包含 `null` 值。

```
$ aws ec2 describe-volumes \  
  --query 'Volumes[*].Tags[?Value == `test`]'
```

## 排除具有指定标签的卷

以下示例描述了所有不带 `test` 标签的实例。使用简单 `?Value != `test`` 表达式不适用于排除卷，因为卷可能有多个标签。只要附加的卷旁边 `test` 还有另一个标签，那么结果中仍会返回该卷。

要排除带有 `test` 标签的所有卷，请从下面的表达式开始返回数组中带有该 `test` 标签的所有标签。任何不是 `test` 标签的标签都包含 `null` 值。

```
$ aws ec2 describe-volumes \  
  --query 'Volumes[*].Tags[?Value == `test`]'
```

然后使用 `test` 函数筛选掉所有正数 `not_null` 结果。

```
$ aws ec2 describe-volumes \  
  --query 'Volumes[!not_null(Tags[?Value == `test`].Value)]'
```

传输结果以展开会导致以下查询的结果。

```
$ aws ec2 describe-volumes \  
  --query 'Volumes[!not_null(Tags[?Value == `test`].Value)] | []'
```

## 结合服务器端和客户端筛选

您可以同时使用服务器端和客户端筛选。首先完成服务器端筛选，将数据发送到客户端，然后由 `--query` 参数进行筛选。如果您使用的是大型数据集，则首先使用服务器端筛选可以减少每次 AWS CLI 调用发送给客户端的数据量，同时仍然保留客户端筛选提供的强大自定义功能。

以下示例列出了使用服务器端和客户端筛选的 Amazon EC2 卷。该服务在 `us-west-2a` 可用区中筛选所有附加的卷的列表。`--query` 参数进一步将输出限制为只有 `Size` 值大于 50 的卷，并且仅显示具有用户定义名称的指定字段。

```
$ aws ec2 describe-volumes \
  --filters "Name=availability-zone,Values=us-west-2a" "Name=status,Values=attached" \
  --query 'Volumes[?Size > `50`].{Id:VolumeId,Size:Size,Type:VolumeType}'
[
  {
    "Id": "vol-0be9bb0bf12345678",
    "Size": 80,
    "VolumeType": "gp2"
  }
]
```

以下示例检索满足多个条件的镜像的列表。然后，它使用 `--query` 参数按 `CreationDate` 对输出进行排序，从而仅选择最新的。最终，它显示这一个镜像的 `ImageId`。

```
$ aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn*gp2" "Name=virtualization-type,Values=hvm" \
  "Name=root-device-type,Values=ebs" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text
ami-00ced3122871a4921
```

以下示例通过使用 `length` 计算列表中的数量，以显示超过 1000 IOPS 的可用卷数。

```
$ aws ec2 describe-volumes \
  --filters "Name=status,Values=available" \
  --query 'length(Volumes[?Iops > `1000`])'
3
```

## 其他资源

### AWS CLI 自动提示

开始使用筛选表达式时，您可以使用 AWS CLI 版本 2 中的自动提示功能。按 F5 键时，自动提示功能提供预览。有关更多信息，请参阅 [the section called “自动提示”](#)。

### JMESPath 终端

JMESPath 终端是一个交互式终端命令，以试验用于客户端筛选的 JMESPath 表达式。使用 `jpterm` 命令，终端会在您键入时显示即时查询结果。您可以直接通过管道将 AWS CLI 输出传送到终端，从而启用高级查询实验。

以下示例将 `aws ec2 describe-volumes` 输出直接传输到 JMESPath 终端。

```
$ aws ec2 describe-volumes | jpterm
```

有关 JMESPath 终端和安装说明的更多信息，请参阅上的 [JMESP ath 终端](#)。GitHub

## jq 实用工具

该 jq 实用工具为您提供了一种将客户端输出转换为所需输出格式的方法。有关详细jq信息和安装说明，请参阅 [jq on](#)。GitHub

## 从 AWS CLI 返回代码

返回代码通常是运行 AWS Command Line Interface (AWS CLI) 命令后发送的隐藏代码，该命令可描述命令的状态。您可以使用 `echo` 命令显示从上一个 AWS CLI 命令发送的代码，并使用这些代码确定命令是成功还是失败，以及命令可能出错的原因。除了返回代码之外，您还可以运行带有 `--debug` 开关的命令，查看有关故障的更多详细信息。此开关将生成一个详细报告，描述 AWS CLI 用于处理命令的步骤以及每个步骤的结果。

要确定 AWS CLI 命令的返回代码，请在运行 CLI 命令后立即运行以下命令之一。

### Linux and macOS

```
$ echo $?  
0
```

### Windows PowerShell

```
PS> echo $lastexitcode  
0
```

### Windows Command Prompt

```
C:\> echo %errorlevel%  
0
```

以下是运行 AWS Command Line Interface (AWS CLI) 命令结束时可能返回的返回代码值。



代码	意义
0	该服务使用 HTTP 响应状态代码 200 进行响应，表示请求发送到的 AWS CLI 和 AWS 服务未生成错误。
1	一个或多个 Amazon S3 传输操作失败。仅限 S3 命令。
2	该返回代码的含义取决于命令： <ul style="list-style-type: none"> <li>• 适用于所有 AWS CLI 命令 – 无法解析输入的命令。解析失败的原因可能是（但不限于）缺少必需的子命令或参数，或使用了未知的命令或参数。</li> <li>• 限制为 S3 命令 – 在传输过程中，跳过了标记为要进行传输的一个或多个文件。但是，标记为要进行传输的所有其他文件都已成功传输。传输过程中跳过的文件包括：不存在的文件，作为字符特殊设备、块特殊设备、FIFO 队列或套接字的文件，以及用户没有读取权限的文件。</li> </ul>
130	命令已被 SIGINT 中断。这是您通过 Ctrl+C 发送的信号，用于取消某个命令。
252	命令语法无效，提供了未知参数，或者参数值不正确并且阻止了命令运行。
253	系统环境或配置无效。尽管提供的命令在语法上可能是有效的，但缺少配置或凭证，导致该命令无法运行。
254	该命令已成功解析，并已对指定服务发出了请求，但该服务返回了错误。这通常表示 API 使用不正确或出现其他特定于服务的问题。
255	命令失败。AWS CLI 或将请求发送到的 AWS 服务生成了错误。

## 使用 AWS CLI 向导的交互式命令

AWS Command Line Interface (AWS CLI) 提供了使用向导执行某些命令的功能。要贡献或查看可用 AWS CLI 向导的完整列表，请查看上的 [wi AWS CLI zards 文件夹](#)。GitHub

### 工作方式

与 AWS 控制台类似，AWS CLI 具有可指导您管理 AWS 资源的用户界面向导。要使用向导，请在命令中的服务名称后调用 wizard 子命令和向导名称。命令结构如下：

语法：

```
$ aws <command> wizard <wizardName>
```

以下示例调用向导来创建新的 dynamodb 表。

```
$ aws dynamodb wizard new-table
```

aws configure 是唯一没有向导名称的向导。运行向导时，如以下示例所示，运行 aws configure wizard 命令：

```
$ aws configure wizard
```

调用向导后，将显示 Shell 中的表单。对于每个参数，都会为您提供一个可供选择的选项列表，或者提示您在字符串中输入。要从列表中进行选择，请使用向上和向下箭头键，然后按 ENTER 键。要查看选项的详细信息，请按向右箭头键。填写完参数后，按 ENTER 键。

```
$ aws configure wizard
What would you like to configure
> Static Credentials
  Assume Role
  Process Provider
  Additional CLI configuration
Enter the name of the profile:
Enter your Access Key Id:
Enter your Secret Access Key:
```

要编辑先前的提示，请使用 SHIFT + TAB。对于某些向导，在填写完所有提示后，您可以预览包含您的信息的 AWS CloudFormation 模板或 AWS CLI 命令。此预览模式对于学习 AWS CLI、服务 API 和为脚本创建模板非常有用。

预览后，按 ENTER 键或最后一个提示符运行最终命令。

```
$ aws configure wizard
What would you like to configure
Enter the name of the profile: testWizard
Enter your Access Key Id: AB1C2D3EF4GH5I678J90K
Enter your Secret Access Key: ab1c2def34gh5i67j8k90l1mnop2qr3s45tu678v90
<ENTER>
```

# 创建和使用名为别名的 AWS CLI 命令快捷方式

别名是您可以在 AWS Command Line Interface (AWS CLI) 中创建的快捷方式，用于缩短经常使用的命令或脚本。您可以在配置文件夹中的 `alias` 文件中创建别名。

## 主题

- [先决条件](#)
- [步骤 1：创建别名文件](#)
- [步骤 2：创建别名](#)
- [步骤 3：调用别名](#)
- [别名存储库示例](#)
- [资源](#)

## 先决条件

要使用别名命令，您需要完成以下操作：

- 安装和配置 AWS CLI。有关更多信息，请参阅 [the section called “安装/更新”](#) 和 [身份验证和访问凭证](#)。
- 请使用最低 AWS CLI 版本 1.11.24 或 2.0.0。
- ( 可选 ) 要使用 AWS CLI 别名 bash 脚本，必须使用与 bash 兼容的终端。

## 步骤 1：创建别名文件

要创建 `alias` 文件，您可以使用文件导航和文本编辑器，也可以通过 step-by-step 以下过程使用首选终端。要快速创建别名文件，请使用以下命令块。

### Linux and macOS

```
$ mkdir -p ~/.aws/cli
$ echo '[toplevel]' > ~/.aws/cli/alias
```

### Windows

```
C:\> md %USERPROFILE%\aws\cli
```

```
C:\> echo [toplevel] > %USERPROFILE%\.aws/cli/alias
```

## 创建别名文件

1. `cli` 在您的 AWS CLI 配置文件夹中创建一个名为 `cli` 的文件夹。默认情况下，配置文件夹为 `~/.aws/` (Linux 或 macOS) 和 `%USERPROFILE%\.aws\` (Windows)。您可以通过文件导航或使用以下命令进行创建。

### Linux and macOS

```
$ mkdir -p ~/.aws/cli
```

### Windows

```
C:\> md %USERPROFILE%\.aws\cli
```

生成的 `cli` 文件夹默认路径为 `~/.aws/cli/` (Linux 或 macOS) 和 `%USERPROFILE%\.aws\cli` (Windows)。

2. 在 `cli` 文件夹中，创建不带扩展名的名为 `alias` 的文本文件，然后将 `[toplevel]` 添加到第一行。您可以通过首选的文本编辑器或使用以下命令创建此文件。

### Linux and macOS

```
$ echo '[toplevel]' > ~/.aws/cli/alias
```

### Windows

```
C:\> echo [toplevel] > %USERPROFILE%\.aws/cli/alias
```

## 步骤 2：创建别名

您可以使用基本命令或 `bash` 脚本创建别名。

### 创建基本命令别名

您可以在上一步中创建的 `alias` 文件中使用以下语法来添加命令，从而来创建别名。

## 语法

```
aliasname = command [--options]
```

**aliasname** 即您所称的别名。**command** 是您想要调用的命令，它可以包括其他别名。您可以在别名中包含选项或参数，也可以在调用别名时添加选项或参数。

以下示例使用 [aws sts get-caller-identity](#) 命令创建别名 `aws whoami`。由于此别名调用了现有 AWS CLI 命令，因此您可以编写不带 `aws` 前缀的命令。

```
whoami = sts get-caller-identity
```

以下示例利用了上一个 `whoami` 示例并添加了 `Account` 筛选条件和文本 `output` 选项。

```
whoami2 = sts get-caller-identity --query Account --output text
```

## 创建子命令别名

### Note

子命令别名功能要求的最低 AWS CLI 版本为 1.11.24 或 2.0.0

您可以在上一步中创建的 `alias` 文件中使用以下语法来添加命令，从而为子命令创建别名。

## 语法

```
[command commandGroup]  
aliasname = command [--options]
```

**commandGroup** 是命令命名空间，例如，命令 `aws ec2 describe-regions` 位于 `ec2` 命令组下。**aliasname** 即您所称的别名。**command** 是您想要调用的命令，它可以包括其他别名。您可以在别名中包含选项或参数，也可以在调用别名时添加选项或参数。

以下示例使用 [aws ec2 describe-regions](#) 命令创建别名 `aws ec2 regions`。由于此别名调用了 `ec2` 命令命名空间下的现有 AWS CLI 命令，因此您可以编写不带 `aws ec2` 前缀的命令。

```
[command ec2]  
regions = describe-regions --query Regions[].RegionName
```

要使用命令命名空间之外的命令创建别名，请在完整命令前面加上感叹号前缀。以下示例使用 [aws iam list-instance-profiles](#) 命令创建别名 `aws ec2 instance-profiles`。

```
[command ec2]
instance-profiles = !aws iam list-instance-profiles
```

### Note

别名仅使用现有命令命名空间，您不能创建新的命名空间。例如，您无法使用 `[command johnsmith]` 部分创建别名，因为 `johnsmith` 命令命名空间尚不存在。

## 创建 bash 脚本别名

### Warning

要使用 AWS CLI 别名 bash 脚本，必须使用与 bash 兼容的终端

您可以使用以下语法为更高级的流程使用 bash 脚本创建别名。

### 语法

```
aliasname =
    !f() {
        script content
    }; f
```

*aliasname* 即您所称的别名，*script content* 是您调用别名时要运行的脚本。

以下示例使用 `opendns` 输出您当前的 IP 地址。由于您可以在其他别名中使用别名，因此以下 `myip` 别名可用于允许或撤消从其他别名访问 IP 地址的权限。

```
myip =
    !f() {
        dig +short myip.opendns.com @resolver1.opendns.com
    }; f
```

以下脚本示例调用了之前的 `aws myip` 别名，以授权 Amazon EC2 安全组入口的 IP 地址。

```
authorize-my-ip =  
  !f() {  
    ip=$(aws myip)  
    aws ec2 authorize-security-group-ingress --group-id ${1} --cidr $ip/32 --protocol  
    tcp --port 22  
  }; f
```

当您调用使用 bash 脚本的别名时，变量将始终按照您输入的顺序进行传递。在 bash 脚本中，不考虑变量名称，仅考虑它们出现的顺序。在以下 `textalert` 别名示例中，`--message` 选项的变量是第一个，`--phone-number` 选项是第二个。

```
textalert =  
  !f() {  
    aws sns publish --message "${1}" --phone-number ${2}  
  }; f
```

### 步骤 3：调用别名

要运行在 `alias` 文件中创建的别名，请使用以下语法。您可以在调用别名时添加其他选项。

语法

```
$ aws aliasname
```

以下示例使用 `aws whoami` 命令别名。

```
$ aws  
whoami  
{  
  "UserId": "A12BCD34E5FGHI6JKLM",  
  "Account": "1234567890987",  
  "Arn": "arn:aws:iam::1234567890987:user/userName"  
}
```

以下示例使用了带有其他选项的 `aws whoami` 别名，仅返回 `Account` 输出中的 `text` 数字。

```
$ aws whoami --query Account --output  
text  
1234567890987
```

以下示例使用 `aws ec2 regions` [子命令别名](#)。

```
$ aws ec2
  regions
[
  "ap-south-1",
  "eu-north-1",
  "eu-west-3",
  "eu-west-2",
  ...
```

## 使用 bash 脚本变量调用别名

调用使用 bash 脚本的别名时，变量将按照输入的顺序进行传递。在 bash 脚本中，不考虑变量的名称，仅考虑它们出现的顺序。例如，在以下 `textalert` 别名中，选项 `--message` 的变量是第一个，`--phone-number` 是第二个。

```
textalert =
!f() {
  aws sns publish --message "${1}" --phone-number ${2}
}; f
```

调用 `textalert` 别名时，您需要按照变量在别名中运行的顺序进行传递。在以下示例中，我们使用变量 `$message` 和 `$phone`。`$message` 变量将作为 `${1}` 选项的 `--message` 传递，`$phone` 变量将作为 `${2}` 选项的 `--phone-number` 传递。这会成功调用 `textalert` 别名来发送消息。

```
$ aws textalert $message
  $phone
{
  "MessageId": "1ab2cd3e4-fg56-7h89-i01j-2klmn34567"
}
```

在以下示例中，将别名调用至 `$phone` 和 `$message` 时，将切换顺序。`$phone` 变量将作为 `${1}` 选项的 `--message` 传递，`$message` 变量将作为 `${2}` 选项的 `--phone-number` 传递。由于变量顺序混乱，因此别名错误地传递了变量。这会导致发生错误，因为 `$message` 的内容与 `--phone-number` 选项的电话号码格式要求不匹配。

```
$ aws textalert $phone
  $message
usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
```



To see help text, you can run:

```
aws help
aws <command> help
aws <command> <subcommand> help
```

Unknown options: text

## 别名存储库示例

上的[AWS CLI 别名存储库GitHub](#)包含 AWS CLI 开发团队和社区创建的 AWS CLI 别名示例。您可以使用整个 alias 文件示例，也可以自己使用单个别名。

### Warning

运行本节中的命令会删除现有 alias 文件。为避免覆盖现有别名文件，请更改下载位置。

### 使用存储库中的别名

1. 安装 Git。有关安装说明，请参阅 Git 文档中的[入门 - 安装 Git](#)。
2. 安装 jp 命令。jp 命令是在 tostring 别名中使用的。有关安装说明，请参阅上的[jmesPath \(jp\) README.md](#)。GitHub
3. 安装 jq 命令。jq 命令是在 tostring-with-jq 别名中使用的。有关安装说明，请参阅上的[JSON 处理器 \(jq\)](#)。GitHub
4. 通过执行以下操作之一下载 alias 文件：
  - 运行以下命令，它是从存储库下载的并将 alias 文件复制到配置文件夹。

Linux and macOS

```
$ git clone https://github.com/awslabs/awscli-aliases.git
$ mkdir -p ~/.aws/cli
$ cp awscli-aliases/alias ~/.aws/cli/alias
```

Windows

```
C:\> git clone https://github.com/awslabs/awscli-aliases.git
C:\> md %USERPROFILE%\aws\cli
C:\> copy awscli-aliases\alias %USERPROFILE%\aws\cli
```

- 直接从存储库下载并保存到 AWS CLI 配置cli文件夹中的文件夹。默认情况下，配置文件夹为 `~/.aws/` (Linux 或 macOS) 和 `%USERPROFILE%\aws\` (Windows)。

5. 要验证别名是否有效，请运行以下别名。

```
$ aws whoami
```

这将显示与 `aws sts get-caller-identity` 命令相同的响应：

```
{
  "Account": "012345678901",
  "UserId": "AIUAINBADX2VEG2TC6HD6",
  "Arn": "arn:aws:iam::012345678901:user/myuser"
}
```

## 资源

- 上的 [AWS CLI 别名存储库GitHub](#) 包含 AWS CLI 开发团队创建的 AWS CLI 别名示例和 AWS CLI 社区的贡献。
- 来自 re [AWS : Invent 2016 : The Effective User](#) 的别名功能公告。AWS CLI YouTube
- [aws sts get-caller-identity](#)
- [aws ec2 describe-instances](#)
- [aws sns publish](#)

# 代码示例

本章提供了一系列示例，向您展示如何将 AWS Command Line Interface (AWS CLI) 与一起使用 AWS 服务。

本 AWS CLI 指南中有以下类型的示例：

- [指导命令示例](#)-《AWS CLI 用户指南》的指导命令示例，介绍如何 AWS CLI 与某些人一起使用 AWS 服务。这些示例通常比参考指南[第 2 AWS CLI 版AWS CLI](#)中的示例更为详细。
- [AWS CLI 命令示例](#)-开源命令示例，也可在参考指南[第 2 AWS CLI 版AWS CLI](#)中找到。命令示例托管在上的[AWS CLI](#)存储库中GitHub。
- [AWS CLI 使用 Bash 脚本代码示例](#)-开源 bash 脚本示例。Bash 脚本示例托管在上的[AWS 代码示例存储库中](#)。GitHub

## 反馈示例

找不到所需的内容？使用本页底部的提供反馈链接或中的相关命令页面请求命令示例。

想要提交示例？在上的“[AWS 代码示例存储库](#)”中贡献 [AWS CLI 命令示例GitHub](#)。有关贡献的更多信息，请参阅GitHub页面上的[AWS CLI 代码示例贡献快速步骤](#)。

# 引导式 AWS CLI 命令示例

本节提供说明如何使用 AWS Command Line Interface (AWS CLI) 访问各种 AWS 服务的示例。

## Note

有关对每种服务的所有可用命令的完整参考，请参阅 [AWS CLI 版本 2 参考指南](#)或使用内置的命令行帮助。有关更多信息，请参阅 [获取有关 AWS CLI 的帮助](#)。

## 服务

- [将 Amazon DynamoDB 与 AWS CLI 结合使用](#)
- [将 Amazon EC2 与 AWS CLI 结合使用](#)
- [将 Amazon S3 Glacier 与 AWS CLI 结合使用](#)

- [从 AWS CLI 使用 AWS Identity and Access Management](#)
- [将 Amazon S3 与 AWS CLI 结合使用](#)
- [将 Amazon SNS 与 AWS CLI 结合使用](#)

## 将 Amazon DynamoDB 与 AWS CLI 结合使用

### Amazon DynamoDB 简介

#### [什么是 Amazon DynamoDB ?](#)

AWS Command Line Interface (AWS CLI) 为所有AWS数据库服务（包括 Amazon DynamoDB）提供支持。您可以使用 AWS CLI 进行临时操作，如创建表。您还可以使用它在实用工具脚本中嵌入 DynamoDB 操作。

有关将 AWS CLI 与 DynamoDB 结合使用的更多信息，请参阅《AWS CLI 命令参考》中的 [dynamodb](#)。

要列出适用于 DynamoDB 的 AWS CLI 命令，请使用以下命令。

```
$ aws dynamodb help
```

#### 主题

- [先决条件](#)
- [创建和使用 DynamoDB 表](#)
- [使用 DynamoDB Local](#)
- [资源](#)

#### 先决条件

要运行 dynamodb 命令，您需要：

- 安装和配置 AWS CLI。有关更多信息，请参阅 [the section called “安装/更新”](#) 和 [身份验证和访问凭证](#)：

## 创建和使用 DynamoDB 表

命令行格式为 DynamoDB 命令名称后接该命令的参数。AWS CLI 对于参数值支持 CLI [速记语法](#)和完全 JSON。

下面的示例创建了一个名为 MusicCollection 的表。

```
$ aws dynamodb create-table \  
  --table-name MusicCollection \  
  --attribute-definitions AttributeName=Artist,AttributeType=S  
  AttributeName=SongTitle,AttributeType=S \  
  --key-schema AttributeName=Artist,KeyType=HASH  
  AttributeName=SongTitle,KeyType=RANGE \  
  --provisioned-throughput ReadCapacityUnits=1,WriteCapacityUnits=1
```

您可以使用类似下面的命令向表中添加新行，如以下示例所示。这些示例使用速记语法和 JSON 的组合。

```
$ aws dynamodb put-item \  
  --table-name MusicCollection \  
  --item '{  
    "Artist": {"S": "No One You Know"},  
    "SongTitle": {"S": "Call Me Today"} ,  
    "AlbumTitle": {"S": "Somewhat Famous"}  
  }' \  
  --return-consumed-capacity TOTAL  
{  
  "ConsumedCapacity": {  
    "CapacityUnits": 1.0,  
    "TableName": "MusicCollection"  
  }  
}
```

```
$ aws dynamodb put-item \  
  --table-name MusicCollection \  
  --item '{  
    "Artist": {"S": "Acme Band"},  
    "SongTitle": {"S": "Happy Day"} ,  
    "AlbumTitle": {"S": "Songs About Life"}  
  }' \  
  --return-consumed-capacity TOTAL
```

```
{
  "ConsumedCapacity": {
    "CapacityUnits": 1.0,
    "TableName": "MusicCollection"
  }
}
```

可能难以在单行命令中编写有效的 JSON。为了使之更简单，AWS CLI 可以读取 JSON 文件。例如，请考虑以下 JSON 代码段，它存储在一个名为 `expression-attributes.json` 的文件中。

```
{
  ":v1": {"S": "No One You Know"},
  ":v2": {"S": "Call Me Today"}
}
```

您可以使用此文件通过 `query` 发出 AWS CLI 请求。在下面的示例中，`expression-attributes.json` 文件的内容用作 `--expression-attribute-values` 参数的值。

```
$ aws dynamodb query --table-name MusicCollection \
  --key-condition-expression "Artist = :v1 AND SongTitle = :v2" \
  --expression-attribute-values file://expression-attributes.json
{
  "Count": 1,
  "Items": [
    {
      "AlbumTitle": {
        "S": "Somewhat Famous"
      },
      "SongTitle": {
        "S": "Call Me Today"
      },
      "Artist": {
        "S": "No One You Know"
      }
    }
  ],
  "ScannedCount": 1,
  "ConsumedCapacity": null
}
```

## 使用 DynamoDB Local

除了 DynamoDB 之外，您还可以将 AWS CLI 与 DynamoDB Local 结合使用。DynamoDB Local 是模拟 DynamoDB 服务的小客户端数据库和服务器。通过 DynamoDB Local 可编写使用 DynamoDB API 的应用程序，而无需操作 DynamoDB Web 服务中的任何表或数据。而所有 API 操作均重新路由到本地数据库。这样可节省预配置吞吐量、数据存储和数据传输费用。

有关 DynamoDB Local 以及如何将其与 AWS CLI 结合使用的更多信息，请参阅 [Amazon DynamoDB 开发人员指南](#) 中的以下部分：

- [DynamoDB Local](#)
- [将 AWS CLI 与 DynamoDB Local 结合使用](#)

## 资源

AWS CLI 参考：

- [aws dynamodb](#)
- [aws dynamodb create-table](#)
- [aws dynamodb put-item](#)
- [aws dynamodb query](#)

服务参考：

- Amazon DynamoDB 开发人员指南中的 [DynamoDB Local](#)
- Amazon DynamoDB 开发人员指南中的 [将 AWS CLI 与 DynamoDB Local 结合使用](#)

## 将 Amazon EC2 与 AWS CLI 结合使用

### Amazon Elastic Compute Cloud 简介

#### [Amazon EC2 简介 – 通过 AWS 实现弹性云服务器和托管](#)

您可以使用 AWS Command Line Interface (AWS CLI) 访问 Amazon Elastic Compute Cloud (Amazon EC2) 的功能。要列出适用于 Amazon EC2 的 AWS CLI 命令，请使用以下命令。

```
aws ec2 help
```

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 [配置 AWS CLI](#)。

本主题展示了执行 Amazon EC2 常见任务的 AWS CLI 命令的短格式示例。

有关 AWS CLI 命令的长格式示例，请参阅 GitHub 上的 [AWS CLI 代码示例存储库](#)。

## 主题

- [创建、显示和删除 Amazon EC2 密钥对](#)
- [创建、配置和删除 Amazon EC2 的安全组](#)
- [启动、列出和终止 Amazon EC2 实例](#)
- [使用 bash 脚本更改 Amazon EC2 实例类型](#)

## 创建、显示和删除 Amazon EC2 密钥对

您可以使用 AWS Command Line Interface (AWS CLI) 为亚马逊弹性计算云 (Amazon EC2) 创建、显示和删除您的密钥对。您可以使用密钥对连接 Amazon EC2 实例。

在创建实例时您必须向 Amazon EC2 提供密钥对，然后在连接到实例时使用该密钥对进行身份验证。

### Note

有关其他命令示例，请参阅 [参考指南](#)。

## 主题

- [先决条件](#)
- [创建密钥对](#)
- [显示您的密钥对](#)
- [删除您的密钥对](#)
- [参考信息](#)

## 先决条件

要运行 ec2 命令，您需要：



- 安装和配置 AWS CLI。有关更多信息，请参阅[the section called “安装/更新”](#) 和 [身份验证和访问凭证](#)。
- 设置您的 IAM 权限以允许访问 Amazon EC2。有关亚马逊 EC2 的 IAM 权限的更多信息，请参阅亚马逊 EC2 用户指南中的亚马逊 EC2 的 [IAM 政策](#)。

## 创建密钥对

要创建密钥对，请使用 `aws ec2 create-key-pair` 命令以及 `--query` 选项和 `--output text` 选项，以通过管道将私有密钥直接传输到文件。

```
$ aws ec2 create-key-pair --key-name MyKeyPair --query 'KeyMaterial' --output text  
> MyKeyPair.pem
```

对于 PowerShell，`> file` 重定向默认为 UTF-8 编码，这不能用于某些 SSH 客户端。因此，您必须通过管道将输出传输到 `out-file` 命令和显式将编码设置为 `ascii` 来转换输出。

```
PS C:\>aws ec2 create-key-pair --key-name MyKeyPair --query 'KeyMaterial' --output text  
| out-file -encoding ascii -filepath MyKeyPair.pem
```

生成的 `MyKeyPair.pem` 文件类似于以下内容。

```
-----BEGIN RSA PRIVATE KEY-----  
EXAMPLEKEYKCAQEAY7WZhaDsrA1W3mRlQtvhwYORRX8gnxgDAfRt/gx42kWXsT4rXE/b5CpSgie/  
vBoU7jLxx92pNHoFnByP+Dc21eyyz6CvjTmWA0JwfWiW5/akH7i05dSrvC7dQkW2duV5QuUdE0QW  
Z/aNxMniGQE6XAgfwlnXVBwrerrQo+ZwQeqiUwwMkuEbLeJFLhMCvYURpUMSC1oehm449ilx9X1F  
G50TCFe0zf18dqqCP6GzbPaIjiU19xX/az0R9V+tpU0zEL+wmXnZt3/nHPQ5xvD20JH67km6SuPW  
oPzev/D8V+x4+bHthfSjR9Y7DvQFjFBVwHXigBdtZcU2/wei8D/HYwIDAQABAoIBAGZ1kaEvnrrqu  
/uler7vgIn5m71N5LKw4hJLAIW6tUT/fzvtcHK0SkbQCQXuriHmQ2MQyJX/0kn2NfjLV/ufGxbl1  
mb5qwMGUnEpJaZD6QSSs3kICLwWUYUiGfc0uiSbmJoap/GTLU0W5Mfcv36PaBUNy5p53V6G7hXb2  
bahyWyJNfjLe4M86yd2YK3V2CmK+X/B0sShnJ36+hjrXPPWmV3N9zEmCdJjA+K15DYmhm/tJWSD9  
81oGk9TopEp7CkIfatEATyyZiVqoRq6k64iuM9JkA30zdXzMQexXVJ1TLZVEH0E7bh1Y9d801ozR  
oQs/FiZNAx2iijCWyv01pjE73+kCgYEA9mZtyhkHkFDpwrSM1APaL8oNAbbjwEy7Z5Mqfql+1Ip1  
YkriL0DbLX1vRAH+yHPRit2hH0jtUNZh4Axv+cpg09qbUI3+43eEy24B7G/Uh+GTfbjsXs0xQx/x  
p9otyVvc7hsQ5TA5PZb+mvkJ50BEKzet9XcKw0NBVELGhnEPe7cCgYEA06Vgov6YHleHui9kHuws  
ayav0elc5zkxjF9nfHFJRry21R1trw2Vdpn+9g481URrpzWV0Eihvm+xTtmaZ1Sp//1kq75XDwnU  
WA8gkn603QE3fq2yN98BURsAKdJfJ5RL1HvGQvTe10HLYYXpJnEkHv+Unl2ajLivWUt5pbBrKbUC  
gYBjb0+0Zk0sCcpZ29sbzYjypIddErySIyRX5gV2uNQwAjLdp9Pfn295yQ+BxMBXiIycWVQiw0bH  
oMo7yykABY70zd5wQewBQ4AdS1WSX4nGDtsiFxiWiI5sKuAAe0CbTosy1s8w8fxoJ5Tz1sdoxNeGs  
Arq6Wv/G16zQuAE9zK9vwwKBgF+09VI/1wJBirsDGz9whVwFFPrTkJNVJZzYt69qezx1sjgFKshy  
WBhd4xHZtmCqpBP1AymEjr/T01bxyARmXMnIOWIANXMGb4KGSy11mzSVAoQ+fqR+cJ3d0dyP11j
```

```
jjb0Ed/NY8fr1NDxAVHE8BSkdsx2f6ELEyBKJSRr9snRAoGAMrTwYneXzvTskF/S5Fyu0i0egLda
NWUH38v/nDCgEpIXD5Hn3qAEcju1IjmbwlvT+nY2jVhv7UGd8MjwUTNGItdb6nsYqM2asrnF3qS
VRkAKKKYeGjKpUfVTiW0YFjXkfcirR/V+QFL50ndHAKJXjW7a4ejJLncTzmZSpYzwApc=
-----END RSA PRIVATE KEY-----
```

您的私钥不存储在 AWS 中，只能在创建后才能检索。以后您无法恢复它。而如果您丢失了私有密钥，则必须创建新的密钥对。

如果您要从 Linux 电脑连接到您的实例，建议您使用以下命令设置您的私有密钥文件的权限，以确保只有您可以读取该文件。

```
$ chmod 400 MyKeyPair.pem
```

### 显示您的密钥对

指纹是从密钥对生成的，您可以使用指纹验证您本地电脑上的私有密钥是否与 AWS 中存储的公有密钥匹配。

指纹是取自私有密钥的 DER 编码副本的 SHA1 哈希。创建密钥对时会捕获此值，并与公钥 AWS 一起存储。您可以在 Amazon EC2 控制台或通过运行 AWS CLI 命令来查看指纹 [aws ec2 describe-key-pairs](#)。

以下示例显示 MyKeyPair 的指纹。

```
$ aws ec2 describe-key-pairs --key-name MyKeyPair
{
  "KeyPairs": [
    {
      "KeyName": "MyKeyPair",
      "KeyFingerprint":
      "1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f"
    }
  ]
}
```

有关密钥和指纹的更多信息，请参阅 [Amazon EC2 用户指南中的 Amazon EC2 密钥对](#)。

### 删除您的密钥对

要删除密钥对，请运行 [aws ec2 delete-key-pair](#) 命令，将 *MyKeyPair* 替换为要删除的密钥对的名称。

```
$ aws ec2 delete-key-pair --key-name MyKeyPair
```

## 参考信息

### AWS CLI 参考：

- [aws ec2](#)
- [aws ec2 create-key-pair](#)
- [aws ec2 delete-key-pair](#)
- [aws ec2 describe-key-pairs](#)

### 其他参考资料：

- [Amazon Elastic Compute Cloud 文档](#)
- 要查看 AWS SDK 和 AWS CLI 代码示例并为其做出贡献，请参阅上的“[AWS 代码示例存储库](#)” GitHub。

## 创建、配置和删除 Amazon EC2 的安全组

您可以为您的 Amazon Elastic Compute Cloud ( Amazon EC2 ) 实例创建安全组，安全组本质上用作一个防火墙，具有可确定哪些网络流量可以进入和离开的规则。

使用 AWS Command Line Interface (AWS CLI) 创建安全组、向现有安全组添加规则以及删除安全组。

### Note

有关其他命令示例，请参阅[参考指南](#)。

## 主题

- [先决条件](#)
- [创建安全组](#)
- [为您的安全组添加规则](#)
- [删除您的安全组](#)

- [参考信息](#)

## 先决条件

要运行 `ec2` 命令，您需要：

- 安装和配置 AWS CLI。有关更多信息，请参阅[the section called “安装/更新”](#) 和 [身份验证和访问凭证](#)。
- 设置您的 IAM 权限以允许访问 Amazon EC2。有关亚马逊 EC2 的 IAM 权限的更多信息，请参阅亚马逊 EC2 用户指南中的亚马逊 EC2 [的 IAM 政策](#)。

## 创建安全组

您可以创建与虚拟私有云 ( VPC ) 关联的安全组。

以下 [aws ec2 create-security-group](#) 示例说明如何为指定的 VPC 创建安全组。

```
$ aws ec2 create-security-group --group-name my-sg --description "My security group" --  
vpc-id vpc-1a2b3c4d  
{  
  "GroupId": "sg-903004f8"  
}
```

要查看安全组的初始信息，请运行 [aws ec2 describe-security-groups](#) 命令。您不能仅通过其 `vpc-id` 而非其名称引用 EC2-VPC 安全组。

```
$ aws ec2 describe-security-groups --group-ids sg-903004f8  
{  
  "SecurityGroups": [  
    {  
      "IpPermissionsEgress": [  
        {  
          "IpProtocol": "-1",  
          "IpRanges": [  
            {  
              "CidrIp": "0.0.0.0/0"  
            }  
          ],  
          "UserIdGroupPairs": []  
        }  
      ],  
    }  
  ],  
}
```

```
    "Description": "My security group"
    "IpPermissions": [],
    "GroupName": "my-sg",
    "VpcId": "vpc-1a2b3c4d",
    "OwnerId": "123456789012",
    "GroupId": "sg-903004f8"
  }
]
```

## 为您的安全组添加规则

运行 Amazon EC2 实例时，您必须启用安全组中的规则，以便为连接到镜像的方法允许入站网络流量。

例如，如果您要启动 Windows 实例，您通常会添加规则以允许 TCP 端口 3389 (RDP) 上的入站流量，从而支持远程桌面协议 (RDP)。如果您要启动 Linux 实例，则通常会添加规则以允许 TCP 端口 22 上的入站流量，从而支持 SSH 连接。

使用 [aws ec2 authorize-security-group-ingress](#) 命令向安全组添加规则。此命令的必需参数是您电脑的公有 IP 地址，或您的电脑连接到的网络（以地址范围形式）（采用 [CIDR](#) 表示法）。

### Note

我们提供以下服务 <https://checkip.amazonaws.com/>，使您可以确定您的公有 IP 地址。要查找可以帮助您识别 IP 地址的其他服务，请使用您的浏览器搜索“what is my IP address”。如果您使用动态 IP 地址通过 ISP 或从防火墙后面进行连接（通过私有网络的 NAT 网关），则您的地址可能会定期更改。在这种情况下，您必须找到客户端电脑使用的 IP 地址的范围。

以下示例说明如何使用 IP 地址将适用于 RDP 的规则（TCP 端口 3389）添加到 ID 为 sg-903004f8 的 EC2-VPC 安全组。

首先，确定您的 IP 地址。

```
$ curl https://checkip.amazonaws.com
x.x.x.x
```

然后，您可以运行 [aws ec2 authorize-security-group-ingress](#) 命令以将 IP 地址添加到安全组。

```
$ aws ec2 authorize-security-group-ingress --group-id sg-903004f8 --protocol tcp --port 3389 --cidr x.x.x.x/x
```

以下命令添加另一个规则以对同一安全组中的实例启用 SSH。

```
$ aws ec2 authorize-security-group-ingress --group-id sg-903004f8 --protocol tcp --port 22 --cidr x.x.x.x/x
```

要查看对安全组所做的更改，请运行 [aws ec2 describe-security-groups](#) 命令。

```
$ aws ec2 describe-security-groups --group-ids sg-903004f8
{
  "SecurityGroups": [
    {
      "IpPermissionsEgress": [
        {
          "IpProtocol": "-1",
          "IpRanges": [
            {
              "CidrIp": "0.0.0.0/0"
            }
          ],
          "UserIdGroupPairs": []
        }
      ],
      "Description": "My security group"
      "IpPermissions": [
        {
          "ToPort": 22,
          "IpProtocol": "tcp",
          "IpRanges": [
            {
              "CidrIp": "x.x.x.x/x"
            }
          ],
          "UserIdGroupPairs": [],
          "FromPort": 22
        }
      ],
      "GroupName": "my-sg",
      "OwnerId": "123456789012",
      "GroupId": "sg-903004f8"
    }
  ]
}
```

```
    }  
  ]  
}
```

## 删除您的安全组

要删除安全组，请运行 [aws ec2 delete-security-group](#) 命令。

### Note

如果安全组当前已附加到环境，则无法删除。

以下命令示例删除 EC2-VPC 安全组。

```
$ aws ec2 delete-security-group --group-id sg-903004f8
```

## 参考信息

AWS CLI 参考：

- [aws ec2](#)
- [aws ec2 authorize-security-group-ingress](#)
- [aws ec2 create-security-group](#)
- [aws ec2 delete-security-group](#)
- [aws ec2 describe-security-groups](#)

其他参考资料：

- [Amazon Elastic Compute Cloud 文档](#)
- 要查看 AWS SDK 和 AWS CLI 代码示例并为其做出贡献，请参阅上的“[AWS 代码示例存储库](#)” GitHub。

## 启动、列出和终止 Amazon EC2 实例

您可以使用 AWS Command Line Interface (AWS CLI) 启动、列出和终止亚马逊弹性计算云 (Amazon EC2) 实例。如果您启动的实例不在 AWS 免费套餐范围内，则即使实例处于闲置状态，也会在启动该实例后向您收费，并按该实例的运行时间收费。

**Note**

有关其他命令示例，请参阅[参考指南](#)。

## 主题

- [先决条件](#)
- [启动实例](#)
- [向实例添加块储存设备](#)
- [向您的实例添加标签](#)
- [连接到您的实例](#)
- [列出您的实例](#)
- [终止实例](#)
- [参考信息](#)

## 先决条件

要运行本主题中的 `ec2` 命令，您需要先完成以下操作：

- 安装和配置 AWS CLI。有关更多信息，请参阅[the section called “安装/更新”](#) 和 [身份验证和访问凭证](#)。
- 设置您的 IAM 权限以允许访问 Amazon EC2。有关亚马逊 EC2 的 IAM 权限的更多信息，请参阅亚马逊 EC2 用户指南中的[亚马逊 EC2 的 IAM 政策](#)。
- 创建[密钥对](#)和[安全组](#)。
- 选择 Amazon Machine Image (AMI) 并记下 AMI ID。有关更多信息，请参阅 Amazon EC2 用户指南中的[寻找合适的 AMI](#)。

## 启动实例

要使用所选的 AMI 启动单个 Amazon EC2 实例，请使用 [aws ec2 run-instances](#) 命令。您可以将实例启动到虚拟私有云 (VPC) 中。

最初，您的实例显示为 `pending` 状态，但在几分钟后将更改为 `running` 状态。

以下示例在 VPC 的指定子网中启动 `t2.micro` 实例。将 `##` 参数值替换为您自己的值。



```
$ aws ec2 run-instances --image-id ami-xxxxxxx --count 1 --instance-type t2.micro --  
key-name MyKeyPair --security-group-ids sg-903004f8 --subnet-id subnet-6e7f829e  
{  
  "OwnerId": "123456789012",  
  "ReservationId": "r-5875ca20",  
  "Groups": [  
    {  
      "GroupName": "my-sg",  
      "GroupId": "sg-903004f8"  
    }  
  ],  
  "Instances": [  
    {  
      "Monitoring": {  
        "State": "disabled"  
      },  
      "PublicDnsName": null,  
      "Platform": "windows",  
      "State": {  
        "Code": 0,  
        "Name": "pending"  
      },  
      "EbsOptimized": false,  
      "LaunchTime": "2013-07-19T02:42:39.000Z",  
      "PrivateIpAddress": "10.0.1.114",  
      "ProductCodes": [],  
      "VpcId": "vpc-1a2b3c4d",  
      "InstanceId": "i-5203422c",  
      "ImageId": "ami-173d747e",  
      "PrivateDnsName": "ip-10-0-1-114.ec2.internal",  
      "KeyName": "MyKeyPair",  
      "SecurityGroups": [  
        {  
          "GroupName": "my-sg",  
          "GroupId": "sg-903004f8"  
        }  
      ],  
      "ClientToken": null,  
      "SubnetId": "subnet-6e7f829e",  
      "InstanceType": "t2.micro",  
      "NetworkInterfaces": [  
        {  
          "Status": "in-use",
```

```
    "SourceDestCheck": true,
    "VpcId": "vpc-1a2b3c4d",
    "Description": "Primary network interface",
    "NetworkInterfaceId": "eni-a7edb1c9",
    "PrivateIpAddresses": [
      {
        "PrivateDnsName": "ip-10-0-1-114.ec2.internal",
        "Primary": true,
        "PrivateIpAddress": "10.0.1.114"
      }
    ],
    "PrivateDnsName": "ip-10-0-1-114.ec2.internal",
    "Attachment": {
      "Status": "attached",
      "DeviceIndex": 0,
      "DeleteOnTermination": true,
      "AttachmentId": "eni-attach-52193138",
      "AttachTime": "2013-07-19T02:42:39.000Z"
    },
    "Groups": [
      {
        "GroupName": "my-sg",
        "GroupId": "sg-903004f8"
      }
    ],
    "SubnetId": "subnet-6e7f829e",
    "OwnerId": "123456789012",
    "PrivateIpAddress": "10.0.1.114"
  }
],
"SourceDestCheck": true,
"Placement": {
  "Tenancy": "default",
  "GroupName": null,
  "AvailabilityZone": "us-west-2b"
},
"Hypervisor": "xen",
"BlockDeviceMappings": [
  {
    "DeviceName": "/dev/sda1",
    "Ebs": {
      "Status": "attached",
      "DeleteOnTermination": true,
      "VolumeId": "vol-877166c8",
```

```

        "AttachTime": "2013-07-19T02:42:39.000Z"
      }
    }
  ],
  "Architecture": "x86_64",
  "StateReason": {
    "Message": "pending",
    "Code": "pending"
  },
  "RootDeviceName": "/dev/sda1",
  "VirtualizationType": "hvm",
  "RootDeviceType": "ebs",
  "Tags": [
    {
      "Value": "MyInstance",
      "Key": "Name"
    }
  ],
  "AmiLaunchIndex": 0
}
]
}

```

## 向实例添加块储存设备

每个启动的实例都具有关联的根设备卷。您可以使用块储存设备映射来指定实例启动时要连接的其他 Amazon Elastic Block Store (Amazon EBS) 卷或实例存储卷。

要向实例添加块储存设备，请在使用 `run-instances` 时指定 `--block-device-mappings` 选项。

以下示例参数配置大小为 20 GB 的标准 Amazon EBS 卷，并使用标识符 `/dev/sdf` 将其映射到您的实例。

```
--block-device-mappings "[{\\"DeviceName\\":\\"/dev/sdf\\",\\"Ebs\\":{\\"VolumeSize\\":20,
\\"DeleteOnTermination\\":false}}]"
```

以下示例基于现有快照添加映射到 `/dev/sdf` Amazon EBS 卷。快照表示加载到卷的镜像。指定快照时，无需指定卷大小；它的大小足够容纳您的镜像。但是，如果您确定指定大小，则大小必须大于或等于快照的大小。

```
--block-device-mappings "[{\\"DeviceName\\":\\"/dev/sdf\\",\\"Ebs\\":{\\"SnapshotId\\":\\"snap-
a1b2c3d4\\"}}]"
```

以下示例向实例添加两个卷。可用于您的实例的卷的数目取决于其实例类型。

```
--block-device-mappings "[{\"DeviceName\":\"/dev/sdf\",\"VirtualName\":\"ephemeral0\"},  
{\"DeviceName\":\"/dev/sdg\",\"VirtualName\":\"ephemeral1\"}]"
```

以下示例创建映射 (/dev/sdj)，但未为实例预配置卷。

```
--block-device-mappings "[{\"DeviceName\":\"/dev/sdj\",\"NoDevice\":\"\"}]"
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[块储存设备映射](#)。

### 向您的实例添加标签

标签是您分配给 AWS 资源的标签。它允许您向您可用于各种目的的资源添加元数据。有关更多信息，请参阅 Amazon EC2 用户指南中的为[资源添加标签](#)。

以下示例显示如何使用 [aws ec2 create-tags](#) 命令，将带有密钥名称“Name”和值“MyInstance”的标签添加到指定的实例。

```
$ aws ec2 create-tags --resources i-5203422c --tags Key=Name,Value=MyInstance
```

### 连接到您的实例

当您的实例运行时，您可以连接到该实例，然后像使用您面前的电脑一样使用该实例。有关更多信息，请参阅 [Amazon EC2 用户指南中的连接到您的亚马逊 EC2 实例](#)。

### 列出您的实例

您可以使用列 AWS CLI 出您的实例并查看有关它们的信息。您可以列出所有实例，或根据您感兴趣的实例对结果进行筛选。

以下示例演示了如何使用 [aws ec2 describe-instances](#) 命令。

以下命令列出您的所有实例。

```
$ aws ec2 describe-instances
```

以下命令将列表筛选到只仅限您的 t2.micro 实例，并仅为每个匹配项输出 InstanceId 值。

```
$ aws ec2 describe-instances --filters "Name=instance-type,Values=t2.micro" --query
"Reservations[].Instances[].InstanceId"
[
  "i-05e998023d9c69f9a"
]
```

以下命令列出具有标签 Name=MyInstance 的任何实例。

```
$ aws ec2 describe-instances --filters "Name=tag:Name,Values=MyInstance"
```

以下命令列出使用以下任何 AMI 启动的实例：ami-x0123456、ami-y0123456 和 ami-z0123456。

```
$ aws ec2 describe-instances --filters "Name=image-id,Values=ami-x0123456,ami-
y0123456,ami-z0123456"
```

## 终止实例

终止实例将删除此实例。在您终止之后，您将无法重新连接到此实例。

一旦实例的状态变为 `shutting-down` 或 `terminated`，您即停止为该实例付费。如果您希望稍后重新连接到实例，请使用 [stop-instances](#)，而不是 `terminate-instances`。有关更多信息，请参阅 Amazon EC2 用户指南中的 [终止您的实例](#)。

要删除实例，请使用命令 [aws ec2 terminate-instances](#) 以将其删除。

```
$ aws ec2 terminate-instances --instance-ids i-5203422c
{
  "TerminatingInstances": [
    {
      "InstanceId": "i-5203422c",
      "CurrentState": {
        "Code": 32,
        "Name": "shutting-down"
      },
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
```

```
}
```

## 参考信息

AWS CLI 参考：

- [aws ec2](#)
- [aws ec2 create-tags](#)
- [aws ec2 describe-instances](#)
- [aws ec2 run-instances](#)
- [aws ec2 terminate-instances](#)

其他参考资料：

- [Amazon Elastic Compute Cloud 文档](#)
- 要查看 AWS SDK 和 AWS CLI 代码示例并为其做出贡献，请参阅上的“[AWS 代码示例存储库](#)” GitHub。

## 使用 bash 脚本更改 Amazon EC2 实例类型

这个 Amazon EC2 的 bash 脚本示例使用 AWS Command Line Interface (AWS CLI) 更改了 Amazon EC2 实例的实例类型。如果实例正在运行，它会停止实例，更改实例类型，然后根据请求重启实例。Shell 脚本是专用于在命令行界面中运行的程序。

### Note

有关其他命令示例，请参阅[参考指南](#)。

## 主题

- [开始之前](#)
- [关于此示例](#)
- [参数](#)
- [文件](#)
- [参考信息](#)

## 开始之前

您需要先满足以下条件，才能运行下文中的任何示例代码。

- 安装和配置 AWS CLI。有关更多信息，请参阅 [the section called “安装/更新”](#) 和 [身份验证和访问凭证](#)。
- 您使用的配置文件必须具有允许示例执行 AWS 操作的权限。
- 在账户中有正在运行的 Amazon EC2 实例，并且您有权停止和修改该实例。如果您运行测试脚本，它会为您启动一个实例，测试更改类型，然后终止该实例。
- AWS 最佳做法是授予此代码最低权限，或者仅授予执行任务所需的权限。有关更多信息，请参阅 AWS Identity and Access Management (IAM) 用户指南中的 [授予最低权限](#)。
- 此代码尚未在所有 AWS 地区进行测试。有些 AWS 服务仅在特定地区可用。有关更多信息，请参阅 AWS 一般参考指南中的 [服务终端节点和配额](#)。
- 运行此代码可能会导致您的 AWS 账户被扣款。您有责任确保在使用完由此脚本创建的任何资源后删除这些资源。

## 关于此示例

此示例是作为 shell 脚本文件 `change_ec2_instance_type.sh` 中的一个函数编写的，您可以通过其他脚本或命令行 `source` 该函数。每个脚本文件都包含了介绍每个函数的注释。一旦该函数进入内存，您就可以通过命令行调用它。例如，以下命令会将指定实例的类型更改为 `t2.nano`：

```
$ source ./change_ec2_instance_type.sh
$ ./change_ec2_instance_type -i *instance-id* -t new-type
```

有关完整示例和可下载的脚本文件，请参阅在的 AWS 代码示例存储库中 [更改 Amazon EC2 实例类型](#)。GitHub

## 参数

`-i` - ( 字符串 ) 指定要修改的实例 ID。

`-t` - ( 字符串 ) 指定要切换到的 Amazon EC2 实例类型。

`-r` - ( 开关 ) 默认情况下，此参数未设置。如果设置了 `-r`，则在类型切换后将会重启实例。

`-f` - ( 开关 ) 默认情况下，脚本在进行切换之前会提示用户确认关闭实例。如果设置了 `-f`，则函数在关闭实例以进行类型切换之前不会提示用户

`-v` - ( 开关 ) 默认情况下，脚本会以静默方式运行，仅在出现错误时才显示输出。如果设置了 `-v`，则函数会在整个运行过程中显示状态。

文件

## `change_ec2_instance_type.sh`

该主脚本文件包含执行以下任务的 `change_ec2_instance_type()` 函数：

- 验证指定的 Amazon EC2 实例是否存在。
- 除非选择了 `-f`，否则函数会在停止实例之前向用户发出警告。
- 更改实例类型
- 如果您设置了 `-r`，请重启实例并确认实例正在运行

查看 [change\\_ec2\\_instance\\_type.sh](#) on 的代码GitHub。

## `test_change_ec2_instance_type.sh`

脚本文件 `test_change_ec2_instance_type.sh` 会测试 `change_ec2_instance_type` 函数的各种代码路径。如果测试脚本中的所有步骤都正确完成，测试脚本将会删除它创建的所有资源。

您可以使用以下参数运行该测试脚本：

- `-v` - ( 开关 ) 每个测试在运行时都会显示通过/失败状态。默认情况下，测试以静默方式运行，输出仅包含最终的总体通过/失败状态。
- `-i` - ( 开关 ) 脚本会在每个测试后暂停，以便您能查看每个步骤的中间结果。让您能够使用 Amazon EC2 控制台检查实例的当前状态。您在系统提示符处按 Enter 后，脚本才会继续执行下一步。

查看 [test\\_change\\_ec2\\_instance\\_type.sh](#) on 的代码GitHub。

## `awsdocs_general.sh`

脚本文件 `awsdocs_general.sh` 中包含了在 AWS CLI 的高级示例中使用的通用函数。

查看 [awsdocs\\_general.sh](#) on 的代码GitHub。

参考信息

AWS CLI 参考：



- [aws ec2](#)
- [aws ec2 describe-instances](#)
- [aws ec2 modify-instance-attribute](#)
- [aws ec2 start-instances](#)
- [aws ec2 stop-instances](#)
- [aws ec2 wait instance-running](#)
- [aws ec2 wait instance-stopped](#)

其他参考资料：

- [Amazon Elastic Compute Cloud 文档](#)
- 要查看 AWS SDK 和 AWS CLI 代码示例并为其做出贡献，请参阅上的“[AWS 代码示例存储库](#)”GitHub。

## 将 Amazon S3 Glacier 与 AWS CLI 结合使用

### Amazon S3 Glacier 简介

#### [Amazon S3 Glacier 简介](#)

本主题显示执行 S3 Glacier 常见任务的 AWS CLI 命令。这些示例演示如何使用 AWS CLI 将大型文件上传到 S3 Glacier，方法是将其拆分为较小的部分并从命令行上传它们。

您可以使用 AWS Command Line Interface (AWS CLI) 访问 Amazon S3 Glacier 功能。要列出 S3 Glacier 的 AWS CLI 命令，请使用以下命令。

```
aws glacier help
```

#### Note

有关命令参考和其他示例，请参阅《AWS CLI 命令参考》中的 [aws glacier](#)。

### 主题

- [先决条件](#)
- [创建 Amazon S3 Glacier 文件库](#)
- [准备要上传的文件](#)
- [启动文件分段上传和上传](#)
- [完成上传](#)
- [资源](#)

## 先决条件

要运行 `glacier` 命令，您需要：

- 安装和配置 AWS CLI。有关更多信息，请参阅 [the section called “安装/更新”](#) 和 [身份验证和访问凭证](#)：
- 本教程使用几个命令行工具，这些工具通常预装在 Unix 类操作系统上，包括 Linux 和 macOS。Windows 用户可以通过安装 [Cygwin](#) 并从 Cygwin 终端运行命令来使用相同的工具。如有可执行相同功能的 Windows 本机命令和实用工具，我们会注明。

## 创建 Amazon S3 Glacier 文件库

使用 `create-vault` 命令创建文件库。

```
$ aws glacier create-vault --account-id - --vault-name myvault
{
  "location": "/123456789012/vaults/myvault"
}
```

### Note

所有 S3 Glacier 命令都需要一个账户 ID 参数。使用连字符 (`--account-id -`) 以使用当前账户。

## 准备要上传的文件

创建一个用于测试上传的文件。以下命令将创建一个正好包含 3 MiB 随机数据的名为 *largefile* 的文件。

## Linux 或 macOS

```
$ dd if=/dev/urandom of=largefile bs=3145728 count=1
1+0 records in
1+0 records out
3145728 bytes (3.1 MB) copied, 0.205813 s, 15.3 MB/s
```

dd 是一个实用工具，该实用工具将大量字节从输入文件复制到输出文件。上一个示例使用系统设备文件 /dev/urandom 作为随机数据的源。fsutil 在 Windows 中执行相似的功能。

## Windows

```
C:\> fsutil file createnew largefile 3145728
File C:\temp\largefile is created
```

接下来，将文件拆分为 1 MiB ( 1,048,576 字节 ) 的区块。

```
$ split -b 1048576 --verbose largefile chunk
creating file `chunkaa'
creating file `chunkab'
creating file `chunkac'
```

### Note

[HJ-Split](#) 是一个免费的文件拆分器，适用于 Windows 和很多其他平台。

## 启动文件分段上传和上传

使用 [initiate-multipart-upload](#) 命令在 Amazon S3 Glacier 中创建分段上传。

```
$ aws glacier initiate-multipart-upload --account-id - --archive-description "multipart upload test" --part-size 1048576 --vault-name myvault
{
  "uploadId": "19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-0ssZtLqyFu7sY1_1R7vgFuJV6NtcV5zpsJ",
  "location": "/123456789012/vaults/myvault/multipart-uploads/19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-0ssZtLqyFu7sY1_1R7vgFuJV6NtcV5zpsJ"
}
```

S3 Glacier 需要每个部分的大小以字节为单位 ( 本例中以 1 MiB 为单位 )、您的文件库名称和一个账户 ID，用来配置分段上传。操作完成时，AWS CLI 会输出一个上传 ID。将上传 ID 保存到 shell 变量以待将来使用。

Linux 或 macOS

```
$ UPLOADID="19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-  
OssZtLqyFu7sY1_1R7vgFuJV6NtcV5zpsJ"
```

Windows

```
C:\> set UPLOADID="19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-  
OssZtLqyFu7sY1_1R7vgFuJV6NtcV5zpsJ"
```

接下来，使用 [upload-multipart-part](#) 命令上传三个部分的每个部分。

```
$ aws glacier upload-multipart-part --upload-id $UPLOADID --body chunkaa --range 'bytes  
0-1048575/*' --account-id - --vault-name myvault  
{  
  "checksum": "e1f2a7cd6e047fa606fe2f0280350f69b9f8cfa602097a9a026360a7edc1f553"  
}  
$ aws glacier upload-multipart-part --upload-id $UPLOADID --body chunkab --range 'bytes  
1048576-2097151/*' --account-id - --vault-name myvault  
{  
  "checksum": "e1f2a7cd6e047fa606fe2f0280350f69b9f8cfa602097a9a026360a7edc1f553"  
}  
$ aws glacier upload-multipart-part --upload-id $UPLOADID --body chunkac --range 'bytes  
2097152-3145727/*' --account-id - --vault-name myvault  
{  
  "checksum": "e1f2a7cd6e047fa606fe2f0280350f69b9f8cfa602097a9a026360a7edc1f553"  
}
```

#### Note

上一个示例使用美元符号 (\$) 在 Linux 上引用 UPLOADID shell 变量的内容。在 Windows 命令行上，在变量名称 ( 例如，%UPLOADID% ) 的任一侧使用百分号 (%)。

在上传各个部分时，您必须指定其字节范围，以便 S3 Glacier 可以按正确的顺序重组它。由于每个部分的大小为 1,048,576 字节，因此第一个部分占用 0-1048575 字节，第二个部分占用 1048576-2097151 字节，第三个部分占用 2097152-3145727 字节。

## 完成上传

Amazon S3 Glacier 需要原始文件的树形哈希，以确认所有上传的部分都已完整地到达AWS。

要计算树形哈希，必须将文件拆分为 1 MiB 的部分并计算每个部分的二进制 SHA-256 哈希。然后，将哈希列表拆分成对，合并每对中的两个二进制哈希，并采用结果的哈希。重复此步骤，直到只剩下一个哈希。如果任一级别出现奇数数量的哈希，请将其提升到下一级别而无需修改。

在使用命令行实用工具时，正确计算树形哈希的关键是以二进制的形式存储每个哈希，并且仅在最后一步将其转换为十六进制。对树中的任何哈希的十六进制版本进行合并或哈希处理将导致错误结果。

### Note

Windows 用户可使用 `type` 命令来代替 `cat`。OpenSSL 适用于 Windows，可在 [OpenSSL.org](https://www.openssl.org) 找到。

## 计算树形哈希

1. 将原始文件拆分为 1 MiB 的部分（如果您还没有这样做）。

```
$ split --bytes=1048576 --verbose largefile chunk
creating file `chunkaa'
creating file `chunkab'
creating file `chunkac'
```

2. 计算并存储每个区块的二进制 SHA-256 哈希。

```
$ openssl dgst -sha256 -binary chunkaa > hash1
$ openssl dgst -sha256 -binary chunkab > hash2
$ openssl dgst -sha256 -binary chunkac > hash3
```

3. 合并前两个哈希，并采用结果的二进制哈希。

```
$ cat hash1 hash2 > hash12
$ openssl dgst -sha256 -binary hash12 > hash12hash
```

4. 将区块 aa 和 ab 的父哈希与区块 ac 的哈希合并并对结果进行哈希处理，此时将输出十六进制。将结果存储在 shell 变量中。

```
$ cat hash12hash hash3 > hash123
```

```
$ openssl dgst -sha256 hash123
SHA256(hash123)= 9628195fcdcbbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67
$ TREEHASH=9628195fcdcbbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67
```

最后，使用 [complete-multipart-upload](#) 命令完成上传。此命令采用原始文件的大小（以字节为单位）、最终树形哈希值（十六进制形式）以及您的账户 ID 和文件库名称。

```
$ aws glacier complete-multipart-upload --checksum $TREEHASH --archive-size 3145728 --
upload-id $UPLOADID --account-id - --vault-name myvault
{
  "archiveId": "d3AbWhE0YE1m6f_fI1jPG82F8xzbMEEZmrALLGAA0NJAzo5QdP-
N83MKqd96Unspoa5H51ItWX-sK8-QS0ZhwsyGiu9-R-
kWUyS1dSBlmgPPWkEbeFfqDSav053rU7FvVLHfRc6hg",
  "checksum": "9628195fcdcbbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67",
  "location": "/123456789012/vaults/myvault/archives/
d3AbWhE0YE1m6f_fI1jPG82F8xzbMEEZmrALLGAA0NJAzo5QdP-N83MKqd96Unspoa5H51ItWX-sK8-
QS0ZhwsyGiu9-R-kWUyS1dSBlmgPPWkEbeFfqDSav053rU7FvVLHfRc6hg"
}
```

您也可以使用 [describe-vault](#) 命令查看文件库的状态。

```
$ aws glacier describe-vault --account-id - --vault-name myvault
{
  "SizeInBytes": 3178496,
  "VaultARN": "arn:aws:glacier:us-west-2:123456789012:vaults/myvault",
  "LastInventoryDate": "2018-12-07T00:26:19.028Z",
  "NumberOfArchives": 1,
  "CreationDate": "2018-12-06T21:23:45.708Z",
  "VaultName": "myvault"
}
```

### Note

基本上每天都会更新一次文件库的状态。有关更多信息，请参阅 [使用文件库](#)。

现在可以安全删除您创建的块和哈希文件。

```
$ rm chunk* hash*
```

有关分段上传的更多信息，请参阅《Amazon S3 Glacier 开发人员指南》中的[分段上传大型档案](#)和[计算校验和](#)。

## 资源

AWS CLI 参考：

- [aws glacier](#)
- [aws glacier complete-multipart-upload](#)
- [aws glacier create-vault](#)
- [aws glacier describe-vault](#)
- [aws glacier initiate-multipart-upload](#)

服务参考：

- [Amazon S3 Glacier 开发人员指南](#)
- 《Amazon S3 Glacier 开发人员指南》中的[分段上传大型档案](#)
- 《Amazon S3 Glacier 开发人员指南》中的[计算校验和](#)
- 《Amazon S3 Glacier 开发人员指南》中的[处理文件库](#)

## 从 AWS CLI 使用 AWS Identity and Access Management

AWS Identity and Access Management 简介

[AWS Identity and Access Management 简介](#)

您可以使用 AWS Command Line Interface (AWS CLI) 访问 AWS Identity and Access Management (IAM) 的功能。要列出 IAM 的 AWS CLI 命令，请使用以下命令。

```
aws iam help
```

本主题显示执行 IAM 常见任务的 AWS CLI 命令。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅[配置 AWS CLI](#)。

有关 IAM 服务的更多信息，请参阅[AWS Identity and Access Management 用户指南](#)。

## 主题

- [创建 IAM 用户和组](#)
- [将 IAM 托管式策略附加到用户](#)
- [为 IAM 用户设置初始密码](#)
- [为 IAM 用户创建访问密钥](#)

## 创建 IAM 用户和组

本主题介绍如何使用 AWS Command Line Interface ( AWS CLI ) 命令创建 AWS Identity and Access Management ( IAM ) 组和新的用户，然后将该用户添加到该组中。有关 IAM 服务的更多信息，请参阅 [AWS Identity and Access Management 用户指南](#)。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅[配置 AWS CLI](#)。

创建组并向其中添加新用户

1. 使用 [create-group](#) 命令创建组。

```
$ aws iam create-group --group-name MyIamGroup
{
  "Group": {
    "GroupName": "MyIamGroup",
    "CreateDate": "2018-12-14T03:03:52.834Z",
    "GroupId": "AGPAJNUJ2W4IJVEXAMPLE",
    "Arn": "arn:aws:iam::123456789012:group/MyIamGroup",
    "Path": "/"
  }
}
```

2. 使用 [create-user](#) 命令创建用户。

```
$ aws iam create-user --user-name MyUser
{
  "User": {
    "UserName": "MyUser",
    "Path": "/",
    "CreateDate": "2018-12-14T03:13:02.581Z",
    "UserId": "AIDAJY2PE5XUZ4EXAMPLE",
    "Arn": "arn:aws:iam::123456789012:user/MyUser"
  }
}
```



```
}
```

3. 使用 [add-user-to-group](#) 命令将用户添加到组。

```
$ aws iam add-user-to-group --user-name MyUser --group-name MyIamGroup
```

4. 要验证 MyIamGroup 组包含 MyUser，请使用 [get-group](#) 命令。

```
$ aws iam get-group --group-name MyIamGroup
{
  "Group": {
    "GroupName": "MyIamGroup",
    "CreateDate": "2018-12-14T03:03:52Z",
    "GroupId": "AGPAJNUJ2W4IJVEXAMPLE",
    "Arn": "arn:aws:iam::123456789012:group/MyIamGroup",
    "Path": "/"
  },
  "Users": [
    {
      "UserName": "MyUser",
      "Path": "/",
      "CreateDate": "2018-12-14T03:13:02Z",
      "UserId": "AIDAJY2PE5XUZ4EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:user/MyUser"
    }
  ],
  "IsTruncated": "false"
}
```

## 将 IAM 托管式策略附加到用户

本主题介绍如何使用 AWS Command Line Interface ( AWS CLI ) 命令将 AWS Identity and Access Management ( IAM ) 策略附加到用户。此示例中的策略为用户提供“高级用户访问”。有关 IAM 服务的更多信息，请参阅 [AWS Identity and Access Management 用户指南](#)。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅[配置 AWS CLI](#)。

## 将 IAM 托管式策略附加到用户

1. 确定要附加的策略的 Amazon 资源名称 (ARN)。以下命令使用 `list-policies` 查找具有名称 `PowerUserAccess` 的策略的 ARN。然后，它会将该 ARN 存储在环境变量中。

```
$ export POLICYARN=$(aws iam list-policies --query 'Policies[?
PolicyName==`PowerUserAccess`].{ARN:Arn}' --output text) ~
$ echo $POLICYARN
arn:aws:iam::aws:policy/PowerUserAccess
```

2. 要附加策略，请使用 [attach-user-policy](#) 命令，并引用存放策略 ARN 的环境变量。

```
$ aws iam attach-user-policy --user-name MyUser --policy-arn $POLICYARN
```

3. 通过运行 [list-attached-user-policies](#) 命令验证策略已附加到此用户。

```
$ aws iam list-attached-user-policies --user-name MyUser
{
  "AttachedPolicies": [
    {
      "PolicyName": "PowerUserAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/PowerUserAccess"
    }
  ]
}
```

有关更多信息，请参阅 [访问管理资源](#)。该主题提供权限和策略概览的链接以及用于访问 Amazon S3、Amazon EC2 和其他服务的示例策略的链接。

## 为 IAM 用户设置初始密码

本主题介绍如何使用 AWS Command Line Interface (AWS CLI) 命令为 AWS Identity and Access Management (IAM) 用户设置初始密码。有关 IAM 服务的更多信息，请参阅 [AWS Identity and Access Management 用户指南](#)。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅[配置 AWS CLI](#)。

以下命令使用 [create-login-profile](#) 为指定用户设置初始密码。当用户首次登录时，用户需要将密码更改为只有用户知道的内容。

```
$ aws iam create-login-profile --user-name MyUser --password My!User1Login8P@ssword --
password-reset-required
{
  "LoginProfile": {
    "UserName": "MyUser",
```

```
    "CreateDate": "2018-12-14T17:27:18Z",
    "PasswordResetRequired": true
  }
}
```

您可以使用 `update-login-profile` 命令更改用户的密码。

```
$ aws iam update-login-profile --user-name MyUser --password My!User1ADifferentP@ssword
```

## 为 IAM 用户创建访问密钥

本主题介绍如何使用 AWS Command Line Interface (AWS CLI) 命令为 AWS Identity and Access Management (IAM) 用户创建一组访问密钥。有关 IAM 服务的更多信息，请参阅 [AWS Identity and Access Management 用户指南](#)。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅[配置 AWS CLI](#)。

您可以使用 [create-access-key](#) 命令为用户创建访问密钥。访问密钥是一组安全凭证，由访问密钥 ID 和私有密钥组成。

用户一次只能创建两个访问密钥。如果您尝试创建第三组，则命令返回 `LimitExceeded` 错误。

```
$ aws iam create-access-key --user-name MyUser
{
  "AccessKey": {
    "UserName": "MyUser",
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "Status": "Active",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY",
    "CreateDate": "2018-12-14T17:34:16Z"
  }
}
```

使用 [delete-access-key](#) 命令为用户删除访问密钥。使用访问密钥 ID 指定要删除的访问密钥。

```
$ aws iam delete-access-key --user-name MyUser --access-key-id AKIAIOSFODNN7EXAMPLE
```

## 将 Amazon S3 与 AWS CLI 结合使用

### Amazon Simple Storage Service (Amazon S3) 简介

#### [Amazon Simple Storage Service \(Amazon S3\) 简介 - AWS 上的云存储](#)

您可以使用 AWS Command Line Interface (AWS CLI) 访问 Amazon Simple Storage Service (Amazon S3) 的功能。AWS CLI 提供两个层级的命令来访问 Amazon S3：

- `s3` – 用于简化常见任务（如创建、操作和删除对象及存储桶）的执行的高级命令。
- `s3api` – 提供对所有 Amazon S3 API 操作的直接访问，使您能够执行高级操作。

本指南中的主题：

- [使用高级别 \(`s3`\) 命令和 AWS CLI](#)
- [通过 AWS CLI 使用 API 级 \(`s3api`\) 命令](#)
- [Amazon S3 存储桶生命周期操作脚本示例](#)

### 使用高级别 (`s3`) 命令和 AWS CLI

本主题介绍一些命令，可用于在 AWS CLI 中通过 [aws s3](#) 命令管理 Amazon S3 存储桶和对象。有关本主题未涵盖的命令和其他命令示例，请参阅《AWS CLI 参考》中的 [aws s3](#) 命令。

高级别 `aws s3` 命令简化了 Amazon S3 对象管理。使用这些命令，您能够在 Amazon S3 自身中管理其内容以及使用本地目录管理这些内容。

主题

- [先决条件](#)
- [开始之前](#)
- [创建存储桶](#)
- [列出存储桶和对象](#)
- [删除存储桶](#)
- [删除对象](#)
- [移动对象](#)

- [复制对象](#)
- [同步对象](#)
- [s3 命令的常用选项](#)
- [资源](#)

## 先决条件

要运行 s3 命令，您需要：

- 安装和配置 AWS CLI。有关更多信息，请参阅 [the section called “安装/更新”](#) 和 [身份验证和访问凭证](#)。
- 您使用的配置文件必须具有允许示例执行 AWS 操作的权限。
- 需了解如下 Amazon S3 术语：
  - 存储桶 – 顶级 Amazon S3 文件夹。
  - 前缀 – 存储桶中的 Amazon S3 文件夹。
  - 对象 – 托管在 Amazon S3 存储桶中的任何项。

## 开始之前

本节介绍在使用 `aws s3` 命令之前需要注意的一些事项。

### 大型对象上载

当您使用 `aws s3` 命令将大型对象上传到 Amazon S3 存储桶时，AWS CLI 会自动执行分段上传。使用这些 `aws s3` 命令时，您无法恢复失败的上传操作。

如果由于超时而导致分段上传失败，或者如果您在中手动取消 AWS CLI，则 AWS CLI 会停止上传并清理所有已创建的文件。此过程可能耗时数分钟。

如果使用 `kill` 命令或者由于系统故障而取消了分段上传或清理过程，则创建的文件将保留在 Amazon S3 存储桶中。[要清理分段上传，请使用 `s3api abort-multipart-upload`](#)

### 分段复制中的文件属性和标签

当您在 `aws s3` 命名空间中使用 AWS CLI 版本 1 版本的命令将文件从一个 Amazon S3 存储桶位置复制到另一个 Amazon S3 存储桶位置，并且该操作使用分[段复制](#)时，源对象中的任何文件属性都不会复制到目标对象。

默认情况下，s3命名空间中执行多部分复制的 AWS CLI 版本 2 命令会将所有标签和以下一组属性从源副本传输到目标副本：content-type、content-language、content-encoding、content-disposition、cache-control、expires 和 metadata。

这可能会导致对 Amazon S3 终端节点进行额外的 AWS API 调用，而这些调用在您使用 AWS CLI 版本 1 时是不会进行的。这些调用可能包括：HeadObject、GetObjectTagging 和 PutObjectTagging。

如果您需要在 AWS CLI 版本 2 命令中更改此默认行为，请使用 --copy-props 参数指定以下选项之一：

- default – 默认值。指定该复制包含附加到源对象的所有标签以及用于非分段复制的 --metadata-directive 参数所包含的属性：content-type、content-language、content-encoding、content-disposition、cache-control、expires 和 metadata。
- metadata-directive – 指定该复制仅包含用于非分段复制的 --metadata-directive 参数所包含的属性。它不会复制任何标签。
- none – 指定该复制不包含源对象中的任何属性。

## 创建存储桶

使用 `s3 mb` 命令创建存储桶。存储桶名称必须是全局唯一的（在所有 Amazon S3 存储桶中都是唯一的），并且应符合 DNS 标准。

存储桶名称可以包含小写字母、数字、连字符和点号。存储桶名称只能以字母或数字开头和结尾，连字符或点号后不能跟点号。

## 语法

```
$ aws s3 mb <target> [--options]
```

## s3 mb 示例

以下示例将创建 `s3://bucket-name` 存储桶。

```
$ aws s3 mb s3://bucket-name
```

## 列出存储桶和对象

要列出存储桶、文件夹或对象，请使用 `s3 ls` 命令。使用不带目标或选项的命令时，将会列出所有存储桶。

## 语法

```
$ aws s3 ls <target> [--options]
```

有关与此命令一起使用的一些常见选项以及相关示例，请参阅 [s3 命令的常用选项](#)。有关可用选项的完整列表，请参阅《AWS CLI 命令参考》中的 [s3 ls](#)。

### s3 ls 示例

以下示例列出您的所有 Amazon S3 存储桶。

```
$ aws s3 ls
2018-12-11 17:08:50 my-bucket
2018-12-14 14:55:44 my-bucket2
```

下面的命令列出一个存储桶中的所有对象和前缀。在本示例输出中，前缀 `example/` 有一个名为 `MyFile1.txt` 的文件。

```
$ aws s3 ls s3://bucket-name
                PRE example/
2018-12-04 19:05:48      3 MyFile1.txt
```

您可以通过在命令中包含特定前缀将输出筛选到此前缀。下面的命令列出 `bucket-name/example/` 中的对象（即 `bucket-name` 中按前缀 `example/` 筛选出的对象）。

```
$ aws s3 ls s3://bucket-name/example/
2018-12-06 18:59:32      3 MyFile1.txt
```

### 删除存储桶

要删除存储桶，请使用 [s3 rb](#) 命令。

## 语法

```
$ aws s3 rb <target> [--options]
```

### s3 rb 示例

以下示例删除 `s3://bucket-name` 存储桶。

```
$ aws s3 rb s3://bucket-name
```

默认情况下，存储桶必须为空，此操作才能成功。要删除不为空的存储桶，您必须包含 `--force` 选项。如果您使用的是受版本控制的存储桶，即其中包含以前删除“但仍保留”的对象，则此命令不允许您删除该存储桶。您必须先删除所有内容。

以下示例命令删除存储桶中的所有对象和前缀，然后删除存储桶本身。

```
$ aws s3 rb s3://bucket-name --force
```

## 删除对象

要删除存储桶或本地目录中的对象，请使用 [s3 rm](#) 命令。

## 语法

```
$ aws s3 rm <target> [--options]
```

有关与此命令一起使用的一些常见选项以及相关示例，请参阅 [s3 命令的常用选项](#)。有关选项的完整列表，请参阅《AWS CLI 命令参考》中的 [s3 rm](#)。

## s3 rm 示例

以下示例将从 `filename.txt` 删除 `s3://bucket-name/example`。

```
$ aws s3 rm s3://bucket-name/example/filename.txt
```

以下示例使用 `s3://bucket-name/example` 选项从 `--recursive` 删除所有对象。

```
$ aws s3 rm s3://bucket-name/example --recursive
```

## 移动对象

使用 [s3 mv](#) 命令可从存储桶或本地目录中移动对象。该 `s3 mv` 命令将源对象或文件复制到指定的目标，然后删除源对象或文件。

## 语法

```
$ aws s3 mv <source> <target> [--options]
```

有关与此命令一起使用的一些常见选项以及相关示例，请参阅 [s3 命令的常用选项](#)。有关可用选项的完整列表，请参阅《AWS CLI 命令参考》中的 [s3 mv](#)。



**⚠ Warning**

如果您在 Amazon S3 源或目标 URI 中使用任何类型的接入点 ARN 或接入点别名，则必须格外小心，确保源和目标 Amazon S3 URI 解析到不同的底层存储桶。如果源存储桶和目标存储桶相同，则可以将源文件或对象移到其自身上，这可能会导致源文件或对象意外删除。要验证源存储桶和目标存储桶是否不同，请使用 `--validate-same-s3-paths` 参数或将环境变量 `AWS_CLI_S3_MV_VALIDATE_SAME_S3_PATHS` 设置为 `true`。

**s3 mv 示例**

以下示例将所有对象从 `s3://bucket-name/example` 移动到 `s3://my-bucket/`。

```
$ aws s3 mv s3://bucket-name/example s3://my-bucket/
```

以下示例使用 `s3 mv` 命令，将本地文件从当前工作目录移动到 Amazon S3 存储桶。

```
$ aws s3 mv filename.txt s3://bucket-name
```

以下示例将文件从 Amazon S3 存储桶移动到当前工作目录，其中 `./` 指定当前的工作目录。

```
$ aws s3 mv s3://bucket-name/filename.txt ./
```

**复制对象**

使用 `s3 cp` 命令可从存储桶或本地目录复制对象。

**语法**

```
$ aws s3 cp <source> <target> [--options]
```

您可以使用短划线参数，将文件流式传输到标准输入 (stdin) 或标准输出 (stdout)。

**⚠ Warning**

如果你使用的是 PowerShell，shell 可能会更改 CRLF 的编码，或者在管道输入或输出或重定向输出中添加 CRLF。

`s3 cp` 命令使用以下语法，将文件流从 stdin 上传到指定的存储桶。

## 语法

```
$ aws s3 cp - <target> [--options]
```

s3 cp 命令使用以下语法下载 stdout 的 Amazon S3 文件流。

## 语法

```
$ aws s3 cp <target> [--options] -
```

有关与此命令一起使用的一些常见选项以及相关示例，请参阅 [s3 命令的常用选项](#)。有关选项的完整列表，请参阅《AWS CLI 命令参考》中的 [s3 cp](#)。

## s3 cp 示例

以下示例将所有对象从 s3://bucket-name/example 复制到 s3://my-bucket/。

```
$ aws s3 cp s3://bucket-name/example s3://my-bucket/
```

以下示例使用 s3 cp 命令，将本地文件从当前工作目录复制到 Amazon S3 存储桶。

```
$ aws s3 cp filename.txt s3://bucket-name
```

以下示例将文件从 Amazon S3 存储桶复制到当前工作目录，其中 ./ 指定当前的工作目录。

```
$ aws s3 cp s3://bucket-name/filename.txt ./
```

以下示例使用 echo 将文本“hello world”流式传输到 s3://bucket-name/filename.txt 文件。

```
$ echo "hello world" | aws s3 cp - s3://bucket-name/filename.txt
```

以下示例将 s3://bucket-name/filename.txt 文件流式传输到 stdout，并将内容输出到控制台。

```
$ aws s3 cp s3://bucket-name/filename.txt -  
hello world
```

以下示例将 s3://bucket-name/pre 的内容流式传输到 stdout，使用 bzip2 命令压缩文件，并将名为 key.bz2 的新压缩文件上传到 s3://bucket-name。

```
$ aws s3 cp s3://bucket-name/pre - | bzip2 --best | aws s3 cp - s3://bucket-name/key.bz2
```

## 同步对象

[s3 sync](#) 命令同步一个存储桶与一个目录中的内容，或者同步两个存储桶中的内容。通常，s3 sync 在源和目标之间复制缺失或过时的文件或对象。不过，您还可以提供 `--delete` 选项来从目标中删除源中不存在的文件或对象。

## 语法

```
$ aws s3 sync <source> <target> [--options]
```

有关与此命令一起使用的一些常见选项以及相关示例，请参阅 [s3 命令的常用选项](#)。有关选项的完整列表，请参阅《AWS CLI 命令参考》中的 [s3 sync](#)。

## s3 sync 示例

下面的示例将存储桶 my-bucket 中名为 path 的 Amazon S3 前缀中的内容与当前工作目录同步。

s3 sync 将更新与目标位置中同名文件的大小或修改时间不同的任何文件。输出显示在同步期间执行的特定操作。请注意，该操作将与 MySubdirectory 递归地同步子目录 s3://my-bucket/path/MySubdirectory 及其内容。

```
$ aws s3 sync . s3://my-bucket/path
upload: MySubdirectory\MyFile3.txt to s3://my-bucket/path/MySubdirectory/MyFile3.txt
upload: MyFile2.txt to s3://my-bucket/path/MyFile2.txt
upload: MyFile1.txt to s3://my-bucket/path/MyFile1.txt
```

下面的示例对上一示例进行了扩展，显示了如何使用 `--delete` 选项。

```
// Delete local file
$ rm ./MyFile1.txt

// Attempt sync without --delete option - nothing happens
$ aws s3 sync . s3://my-bucket/path

// Sync with deletion - object is deleted from bucket
$ aws s3 sync . s3://my-bucket/path --delete
delete: s3://my-bucket/path/MyFile1.txt
```

```
// Delete object from bucket
$ aws s3 rm s3://my-bucket/path/MySubdirectory/MyFile3.txt
delete: s3://my-bucket/path/MySubdirectory/MyFile3.txt

// Sync with deletion - local file is deleted
$ aws s3 sync s3://my-bucket/path . --delete
delete: MySubdirectory\MyFile3.txt

// Sync with Infrequent Access storage class
$ aws s3 sync . s3://my-bucket/path --storage-class STANDARD_IA
```

使用 `--delete` 选项时，`--exclude` 和 `--include` 选项可以筛选要在 `s3 sync` 操作过程中删除的文件或对象。在这种情况下，参数字符串必须指定要在目标目录或存储桶上下文中包含或排除在删除操作中的文件。下面是一个示例。

```
Assume local directory and s3://my-bucket/path currently in sync and each contains 3
files:
MyFile1.txt
MyFile2.rtf
MyFile88.txt
...

// Sync with delete, excluding files that match a pattern. MyFile88.txt is deleted,
while remote MyFile1.txt is not.
$ aws s3 sync . s3://my-bucket/path --delete --exclude "path/MyFile?.txt"
delete: s3://my-bucket/path/MyFile88.txt
...

// Sync with delete, excluding MyFile2.rtf - local file is NOT deleted
$ aws s3 sync s3://my-bucket/path . --delete --exclude "./MyFile2.rtf"
download: s3://my-bucket/path/MyFile1.txt to MyFile1.txt
...

// Sync with delete, local copy of MyFile2.rtf is deleted
$ aws s3 sync s3://my-bucket/path . --delete
delete: MyFile2.rtf
```

## s3 命令的常用选项

以下选项经常用于本主题中所述的命令。有关可在命令上使用的选项的完整列表，请参阅参考指南[第 2 节 AWS CLI 版 AWS CLI 中的特定命令](#)。

## acl

`s3 sync` 和 `s3 cp` 可以使用 `--acl` 选项。这样您能够为复制到 Amazon S3 的文件设置访问权限。`--acl` 选项接受 `private`、`public-read` 和 `public-read-write` 值。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的[标准 ACL](#)。

```
$ aws s3 sync . s3://my-bucket/path --acl public-read
```

## exclude

使用 `s3 cp`、`s3 mv`、`s3 sync` 或 `s3 rm` 命令时，可以使用 `--exclude` 或 `--include` 选项筛选结果。`--exclude` 选项将规则设置为仅从命令中排除对象，并且系统将按照指定的顺序应用这些选项。如下例所示。

```
Local directory contains 3 files:
MyFile1.txt
MyFile2.rtf
MyFile88.txt

// Exclude all .txt files, resulting in only MyFile2.rtf being copied
$ aws s3 cp . s3://my-bucket/path --exclude "*.txt"

// Exclude all .txt files but include all files with the "MyFile*.txt" format,
resulting in, MyFile1.txt, MyFile2.rtf, MyFile88.txt being copied
$ aws s3 cp . s3://my-bucket/path --exclude "*.txt" --include "MyFile*.txt"

// Exclude all .txt files, but include all files with the "MyFile*.txt" format,
but exclude all files with the "MyFile?.txt" format resulting in, MyFile2.rtf and
MyFile88.txt being copied
$ aws s3 cp . s3://my-bucket/path --exclude "*.txt" --include "MyFile*.txt" --
exclude "MyFile?.txt"
```

情况如：

使用 `s3 cp`、`s3 mv`、`s3 sync` 或 `s3 rm` 命令时，可以使用 `--exclude` 或 `--include` 选项筛选结果。`--include` 选项将规则设置为仅包括为命令指定的对象，并且系统将按照指定的顺序应用这些选项。如下例所示。

```
Local directory contains 3 files:
MyFile1.txt
MyFile2.rtf
```

```
MyFile88.txt

// Include all .txt files, resulting in MyFile1.txt and MyFile88.txt being copied
$ aws s3 cp . s3://my-bucket/path --include "*.txt"

// Include all .txt files but exclude all files with the "MyFile*.txt" format,
resulting in no files being copied
$ aws s3 cp . s3://my-bucket/path --include "*.txt" --exclude "MyFile*.txt"

// Include all .txt files, but exclude all files with the "MyFile*.txt" format, but
include all files with the "MyFile?.txt" format resulting in MyFile1.txt being
copied

$ aws s3 cp . s3://my-bucket/path --include "*.txt" --exclude "MyFile*.txt" --
include "MyFile?.txt"
```

## 授予

s3 cp、s3 mv 和 s3 sync 命令包括一个 --grants 选项，用来向指定的用户或组授予对对象的权限。使用以下语法对权限列表设置 --grants 选项。将 Permission、Grantee\_Type 和 Grantee\_ID 替换为您自己的值。

### 语法

```
--grants Permission=Grantee_Type=Grantee_ID
        [Permission=Grantee_Type=Grantee_ID ...]
```

每个值都包含以下元素：

- **##** – 指定授予的权限。可以设置为 read、readacl、writeacl 或 full。
- **Grantee\_Type** – 指定如何标识被授权者。可以设置为 uri、emailaddress 或 id。
- **Grantee\_ID** – 根据 **Grantee\_Type** 指定被授权者。
  - uri – 组 URI。有关更多信息，请参阅 [谁是授权者？](#)
  - emailaddress – 账户的电子邮件地址。
  - id – 账户的规范 ID。

有关 Amazon S3 访问控制的更多信息，请参阅 [访问控制](#)。

下面的示例将一个对象复制到一个存储桶中。它授予所有人对对象的 read 权限，向 full 的关联账户授予 read 权限 (readacl、writeacl 和 user@example.com)。

```
$ aws s3 cp file.txt s3://my-bucket/ --grants read-uri=http://acs.amazonaws.com/groups/global/AllUsers full=emailaddress=user@example.com
```

还可以为上传到 Amazon S3 的对象指定非默认存储类 ( `REDUCED_REDUNDANCY` 或 `STANDARD_IA` )。为此，请使用 `--storage-class` 选项。

```
$ aws s3 cp file.txt s3://my-bucket/ --storage-class REDUCED_REDUNDANCY
```

## recursive

使用此选项时，系统针对所指定目录或前缀下的所有文件或对象执行该命令。以下示例删除 `s3://my-bucket/path` 及其所有内容。

```
$ aws s3 rm s3://my-bucket/path --recursive
```

## 资源

### AWS CLI 参考：

- [aws s3](#)
- [aws s3 cp](#)
- [aws s3 mb](#)
- [aws s3 mv](#)
- [aws s3 ls](#)
- [aws s3 rb](#)
- [aws s3 rm](#)
- [aws s3 sync](#)

### 服务参考：

- Amazon Simple Storage Service 用户指南中的 [使用 Amazon S3 存储桶](#)
- Amazon Simple Storage Service 用户指南中的 [使用 Amazon S3 对象](#)
- Amazon Simple Storage Service 用户指南中的 [使用前缀和分隔符按层次结构列出键](#)
- [使用 Amazon 简单存储服务用户指南中的 AWS SDK for .NET \( 低级 \) 中止分段上传到 S3 存储桶](#)

## 通过 AWS CLI 使用 API 级 ( s3api ) 命令

API 级命令 ( 包含在 s3api 命令集中 ) 提供对 Amazon Simple Storage Service (Amazon S3) API 的直接访问，并且可以执行高级别 s3 命令中未公开的某些操作。这些命令等同于对服务功能提供 API 级访问的其他 AWS 服务的命令。有关 s3 命令的更多信息，请参阅 [使用高级别 \(s3\) 命令和 AWS CLI](#)。

本主题提供了若干示例，以演示如何使用映射到 Amazon S3 API 的低级别命令。此外，您可以在 [AWS CLI 版本 2 参考指南](#) 的 s3api 部分中找到每个 S3 API 命令的示例。

### 主题

- [先决条件](#)
- [应用自定义 ACL](#)
- [配置日志记录策略](#)
- [资源](#)

### 先决条件

要运行 s3api 命令，您需要：

- 安装和配置 AWS CLI。有关更多信息，请参阅 [the section called “安装/更新”](#) 和 [身份验证和访问凭证](#)：
- 您使用的配置文件必须具有允许示例代码执行 AWS 操作的权限。
- 需了解如下 Amazon S3 术语：
  - 存储桶 – 顶级 Amazon S3 文件夹。
  - 前缀 – 存储桶中的 Amazon S3 文件夹。
  - 对象 – 托管在 Amazon S3 存储桶中的任何项。

### 应用自定义 ACL

对于高级别命令，您可以使用 --acl 选项对 Amazon S3 对象应用预定义的访问控制列表 (ACL)。但不能使用该命令设置存储桶范围的 ACL。但是，您可以通过使用 [put-bucket-acl](#) API 级命令来执行此操作。

下面的示例说明如何向两个 AWS 用户 ( user1@example.com 和 user2@example.com ) 授予完全控制权限，并向所有人授予读取权限。“everyone”的标识符来自作为参数传递的特殊 URI。



```
$ aws s3api put-bucket-acl --bucket MyBucket --grant-full-control  
'emailaddress="user1@example.com",emailaddress="user2@example.com"' --grant-read  
'uri="http://acs.amazonaws.com/groups/global/AllUsers"'
```

有关如何构建 ACL 的详细信息，请参阅 Amazon Simple Storage Service API 参考中的 [PUT Bucket acl](#)。CLI 中的 s3api ACL 命令（如 put-bucket-acl）使用相同的 [简化参数表示法](#)。

## 配置日志记录策略

API 命令 put-bucket-logging 配置存储桶日志记录策略。

在下面的示例中，已向 AWS 用户 user@example.com 授予对日志文件的完全控制权限，而向所有用户授予了读取访问权限。请注意，还需要使用 put-bucket-acl 命令向 Amazon S3 的日志传输系统（由 URI 指定）授予必要的权限，以读取和向存储桶写入日志。

```
$ aws s3api put-bucket-acl --bucket MyBucket --grant-read-acp 'URI="http://  
acs.amazonaws.com/groups/s3/LogDelivery"' --grant-write 'URI="http://acs.amazonaws.com/  
groups/s3/LogDelivery"'  
$ aws s3api put-bucket-logging --bucket MyBucket --bucket-logging-status file://  
logging.json
```

上一个命令中的 logging.json 文件具有以下内容。

```
{  
  "LoggingEnabled": {  
    "TargetBucket": "MyBucket",  
    "TargetPrefix": "MyBucketLogs/",  
    "TargetGrants": [  
      {  
        "Grantee": {  
          "Type": "AmazonCustomerByEmail",  
          "EmailAddress": "user@example.com"  
        },  
        "Permission": "FULL_CONTROL"  
      },  
      {  
        "Grantee": {  
          "Type": "Group",  
          "URI": "http://acs.amazonaws.com/groups/global/AllUsers"  
        },  
        "Permission": "READ"  
      }  
    ]  
  }  
}
```

```
    }  
  ]  
}  
}
```

## 资源

### AWS CLI 参考：

- [aws s3api](#)
- [aws s3api put-bucket-acl](#)
- [aws s3api put-bucket-logging](#)

### 服务参考：

- Amazon Simple Storage Service 用户指南中的[使用 Amazon S3 存储桶](#)
- Amazon Simple Storage Service 用户指南中的[使用 Amazon S3 对象](#)
- Amazon Simple Storage Service 用户指南中的[使用前缀和分隔符按层次结构列出键](#)
- Amazon Simple Storage Service 用户指南中的[使用 AWS SDK for .NET \( 低级别 \) 中止到 S3 存储桶的分段上传](#)

## Amazon S3 存储桶生命周期操作脚本示例

本主题介绍使用 AWS Command Line Interface (AWS CLI) 的 Amazon S3 存储桶生命周期操作的 bash 脚本示例。此脚本示例使用 [aws s3api](#) 命令集。Shell 脚本是专用于在命令行界面中运行的程序。

### 主题

- [在您开始之前](#)
- [关于此示例](#)
- [文件](#)
- [参考信息](#)

### 在您开始之前

您需要先满足以下条件，才能运行下文中的任何示例代码。

- 安装和配置 AWS CLI。有关更多信息，请参阅 [the section called “安装/更新”](#) 和 [身份验证和访问凭证](#)：
- 您使用的配置文件必须具有允许示例代码执行AWS操作的权限。
- 作为AWS的最佳实践，请授予此代码最低的权限，或者仅授予它执行任务所需的权限。有关更多信息，请参阅 IAM 用户指南 中的 [授予最低权限](#)。
- 此代码尚未在所有AWS区域中进行测试。有些 AWS 服务仅在特定区域中提供。有关更多信息，请参阅 AWS 一般参考指南中的 [服务终端节点和配额](#)。
- 运行此代码可能会导致您的AWS账户产生相关费用。您有责任确保在使用完由此脚本创建的任何资源后删除这些资源。

Amazon S3 服务使用以下术语：

- 存储桶 – 顶级 Amazon S3 文件夹。
- 前缀 – 存储桶中的 Amazon S3 文件夹。
- 对象 – Amazon S3 存储桶中托管的任何项。

关于此示例

此示例说明如何使用 shell 脚本文件中的一组函数与一些基本的 Amazon S3 操作进行交互。这些函数包含在名为 `bucket-operations.sh` 的 shell 脚本文件中。您可以在另一个文件中调用这些函数。每个脚本文件都包含了介绍每个函数的注释。

要查看每个步骤的中间结果，请使用 `-i` 参数运行脚本。您可以使用 Amazon S3 控制台查看存储桶或其内容的当前状态。仅当您在系统提示符处按 Enter 后，脚本才会继续执行下一步。

有关完整示例和可下载的脚本文件，请参阅 GitHub 上的 AWS 代码示例存储库 中的 [Amazon S3 存储桶生命周期操作](#)。

文件

此示例包含以下文件：

`bucket-operations.sh`

此主脚本文件可以从另一个文件中调取。它包含了执行以下任务的函数：

- 创建存储桶并验证它是否存在

- 将文件从本地电脑复制到存储桶
- 将文件从一个存储桶位置复制到另一个存储桶位置
- 列出存储桶中的内容
- 从存储桶中删除文件
- 删除存储桶

在 GitHub 上查看 [bucket-operations.sh](#) 的代码。

#### test-bucket-operations.sh

shell 脚本文件 `test-bucket-operations.sh` 演示了如何调用这些函数，即调取 `bucket-operations.sh` 文件然后调用其中的每个函数。调用函数后，该测试脚本会删除它创建的所有资源。

在 GitHub 上查看 [test-bucket-operations.sh](#) 的代码。

#### awsdocs-general.sh

脚本文件 `awsdocs-general.sh` 中包含了在 AWS CLI 的高级代码示例中使用的通用函数。

在 GitHub 上查看 [awsdocs-general.sh](#) 的代码。

#### 参考信息

##### AWS CLI 参考：

- [aws s3api](#)
- [aws s3api create-bucket](#)
- [aws s3api copy-object](#)
- [aws s3api delete-bucket](#)
- [aws s3api delete-object](#)
- [aws s3api head-bucket](#)
- [aws s3api list-objects](#)
- [aws s3api put-object](#)

##### 其他参考资料：

- Amazon Simple Storage Service 用户指南中的[使用 Amazon S3 存储桶](#)
- Amazon Simple Storage Service 用户指南中的[使用 Amazon S3 对象](#)
- 要查看和贡献 AWS 开发工具包和 AWS CLI 代码示例，请参阅 GitHub 上的[AWS 代码示例存储库](#)。

## 将 Amazon SNS 与 AWS CLI 结合使用

您可以使用 AWS Command Line Interface (AWS CLI) 访问 Amazon Simple Notification Service (Amazon SNS) 的功能。要列出适用于 Amazon SNS 的 AWS CLI 命令，请使用以下命令。

```
aws sns help
```

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅[配置 AWS CLI](#)。

本主题显示执行 Amazon SNS 常见任务的 AWS CLI 命令。

### 主题

- [创建主题](#)
- [订阅主题](#)
- [向主题发布](#)
- [取消订阅主题](#)
- [删除主题](#)

### 创建主题

要创建主题，请使用 [sns create-topic](#) 命令并指定要分配给该主题的名称。

```
$ aws sns create-topic --name my-topic
{
  "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

记下响应的 TopicArn，您随后将用它来发布消息。

### 订阅主题

要订阅主题，请使用 [sns subscribe](#) 命令。

以下示例为 `email` 指定 `notification-endpoint` 协议和电子邮件地址。

```
$ aws sns subscribe --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic --
protocol email --notification-endpoint saanvi@example.com
{
  "SubscriptionArn": "pending confirmation"
}
```

AWS 通过向您 `subscribe` 命令中指定的地址发送电子邮件，立即发送确认电子邮件。电子邮件具有以下文本。

```
You have chosen to subscribe to the topic:
arn:aws:sns:us-west-2:123456789012:my-topic
To confirm this subscription, click or visit the following link (If this was in error
no action is necessary):
Confirm subscription
```

收件人单击确认订阅链接后，收件人的浏览器显示通知消息，信息类似于以下内容。

```
Subscription confirmed!

You have subscribed saanvi@example.com to the topic:my-topic.

Your subscription's id is:
arn:aws:sns:us-west-2:123456789012:my-topic:1328f057-de93-4c15-512e-8bb22EXAMPLE

If it was not your intention to subscribe, click here to unsubscribe.
```

## 向主题发布

要将消息发送给某一主题的所有订阅者，请使用 `sns publish` 命令。

以下示例发送消息“Hello World!” 特定主体的所有订阅者。

```
$ aws sns publish --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic --
message "Hello World!"
{
  "MessageId": "4e41661d-5eec-5ddf-8dab-2c867EXAMPLE"
}
```

在本示例中，AWS 将包含文本“Hello World!”的电子邮件发送到 `saanvi@example.com`。

## 取消订阅主题

要取消订阅某个主题并停止接收向该主题发布的消息，请使用 [sns unsubscribe](#) 命令并指定您要取消订阅的主题的 ARN。

```
$ aws sns unsubscribe --subscription-arn arn:aws:sns:us-west-2:123456789012:my-topic:1328f057-de93-4c15-512e-8bb22EXAMPLE
```

要验证您已成功取消订阅，请使用 [sns list-subscriptions](#) 命令以确认该 ARN 不再显示在列表中。

```
$ aws sns list-subscriptions
```

## 删除主题

要删除主题，请运行 [sns delete-topic](#) 命令。

```
$ aws sns delete-topic --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic
```

要验证 AWS 是否已成功删除主题，请使用 [sns list-topics](#) 命令以确认该主题不再显示在列表中。

```
$ aws sns list-topics
```

## AWS CLI 命令示例

本主题中的代码示例向您展示了如何使用 wit AWS Command Line Interface h AWS。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

跨服务示例是指跨多个 AWS 服务工作的示例应用程序。

示例

- [使用的操作和场景 AWS CLI](#)

## 使用的操作和场景 AWS CLI

以下代码示例展示了如何使用with来执行操作和实现常见场景 AWS 服务。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是指显示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

### 服务

- [使用的 ACM 示例 AWS CLI](#)
- [使用 API Gateway 示例 AWS CLI](#)
- [API Gateway HTTP 和 WebSocket API 示例使用 AWS CLI](#)
- [API Gateway 管理 API 示例使用 AWS CLI](#)
- [使用 App Mesh 示例 AWS CLI](#)
- [App Runner 示例使用 AWS CLI](#)
- [AWS AppConfig 使用示例 AWS CLI](#)
- [Application Auto Scaling 示例 AWS CLI](#)
- [使用 Application Discovery 服务示例 AWS CLI](#)
- [AppRegistry 使用示例 AWS CLI](#)
- [使用 Athena 的示例 AWS CLI](#)
- [使用的 Auto Scaling 示例 AWS CLI](#)
- [使用 Auto Scaling Plans 示例 AWS CLI](#)
- [AWS Backup 使用示例 AWS CLI](#)
- [AWS Batch 使用示例 AWS CLI](#)
- [AWS Budgets 使用示例 AWS CLI](#)
- [使用 Amazon Chime 示例 AWS CLI](#)
- [使用云控制 API 示例 AWS CLI](#)
- [AWS Cloud Map 使用示例 AWS CLI](#)
- [AWS Cloud9 使用示例 AWS CLI](#)



- [AWS CloudFormation 使用示例 AWS CLI](#)
- [CloudFront 使用示例 AWS CLI](#)
- [使用亚马逊的 CloudSearch 示例 AWS CLI](#)
- [CloudTrail 使用示例 AWS CLI](#)
- [CloudWatch 使用示例 AWS CLI](#)
- [CloudWatch 使用记录示例 AWS CLI](#)
- [CloudWatch 使用网络监控示例 AWS CLI](#)
- [CodeArtifact 使用示例 AWS CLI](#)
- [CodeBuild 使用示例 AWS CLI](#)
- [CodeCommit 使用示例 AWS CLI](#)
- [CodeDeploy 使用示例 AWS CLI](#)
- [CodeGuru 使用审阅者示例 AWS CLI](#)
- [CodePipeline 使用示例 AWS CLI](#)
- [AWS CodeStar 使用示例 AWS CLI](#)
- [AWS CodeStar 使用通知示例 AWS CLI](#)
- [CodeConnections 使用示例 AWS CLI](#)
- [使用 Amazon Cognito 身份示例 AWS CLI](#)
- [使用 Amazon Cognito 身份提供商示例 AWS CLI](#)
- [使用 Amazon Comprehend 示例 AWS CLI](#)
- [使用 Amazon Comprehend Medical 的示例 AWS CLI](#)
- [AWS Config 使用示例 AWS CLI](#)
- [使用 Amazon Connect 的示例 AWS CLI](#)
- [AWS 成本和使用情况报告 使用示例 AWS CLI](#)
- [使用 Cost Explorer 服务示例 AWS CLI](#)
- [使用的 Firehose 示例 AWS CLI](#)
- [使用 Amazon Data Lifecycle Manager 示例 AWS CLI](#)
- [AWS Data Pipeline 使用示例 AWS CLI](#)
- [DataSync 使用示例 AWS CLI](#)
- [使用的 DAX 示例 AWS CLI](#)

- [使用 Detective 示例 AWS CLI](#)
- [使用 Device Farm 示例 AWS CLI](#)
- [AWS Direct Connect 使用示例 AWS CLI](#)
- [AWS Directory Service 使用示例 AWS CLI](#)
- [AWS DMS 使用示例 AWS CLI](#)
- [使用 Amazon DocumentDB 示例 AWS CLI](#)
- [使用的 DynamoDB 示例 AWS CLI](#)
- [使用 DynamoDB Streams 示例 AWS CLI](#)
- [使用 Amazon EC2 的示例 AWS CLI](#)
- [使用 Amazon EC2 Instance Connect 示例 AWS CLI](#)
- [使用 Amazon ECR 示例 AWS CLI](#)
- [使用 Amazon ECS 示例 AWS CLI](#)
- [使用 Amazon EFS 示例 AWS CLI](#)
- [使用 Amazon EKS 的示例 AWS CLI](#)
- [使用 Elastic Beanstalk 示例 AWS CLI](#)
- [Elastic Load Balancing-版本 1 示例使用 AWS CLI](#)
- [Elastic Load Balancing-版本 2 示例使用 AWS CLI](#)
- [使用 Elastic Transcoder 示例 AWS CLI](#)
- [ElastiCache 使用示例 AWS CLI](#)
- [MediaStore 使用示例 AWS CLI](#)
- [使用 Amazon EMR 示例 AWS CLI](#)
- [EKS 上的 Amazon EMR 示例使用 AWS CLI](#)
- [EventBridge 使用示例 AWS CLI](#)
- [使用 Firewall Manager 示例 AWS CLI](#)
- [AWS FIS 使用示例 AWS CLI](#)
- [使用亚马逊的 GameLift 示例 AWS CLI](#)
- [使用全球加速器示例 AWS CLI](#)
- [AWS Glue 使用示例 AWS CLI](#)
- [GuardDuty 使用示例 AWS CLI](#)

- [AWS Health 使用示例 AWS CLI](#)
- [HealthImaging 使用示例 AWS CLI](#)
- [HealthLake 使用示例 AWS CLI](#)
- [HealthOmics 使用示例 AWS CLI](#)
- [使用的 IAM 示例 AWS CLI](#)
- [使用 IAM 访问分析器示例 AWS CLI](#)
- [使用的 Image Builder 示例 AWS CLI](#)
- [使用的事件管理器示例 AWS CLI](#)
- [使用的事件管理器联系人示例 AWS CLI](#)
- [Amazon Inspector 示例使用 AWS CLI](#)
- [AWS IoT 使用示例 AWS CLI](#)
- [AWS IoT 1-Click 使用的设备示例 AWS CLI](#)
- [AWS IoT 1-Click 使用项目示例 AWS CLI](#)
- [AWS IoT Analytics 使用示例 AWS CLI](#)
- [使用“设备顾问”示例 AWS CLI](#)
- [AWS IoT data 使用示例 AWS CLI](#)
- [AWS IoT Events 使用示例 AWS CLI](#)
- [AWS IoT Events-Data 使用示例 AWS CLI](#)
- [AWS IoT Greengrass 使用示例 AWS CLI](#)
- [AWS IoT Greengrass V2 使用示例 AWS CLI](#)
- [AWS IoT Jobs SDK release 使用示例 AWS CLI](#)
- [AWS IoT SiteWise 使用示例 AWS CLI](#)
- [AWS IoT Things Graph 使用示例 AWS CLI](#)
- [AWS IoT Wireless 使用示例 AWS CLI](#)
- [使用 Amazon IVS 示例 AWS CLI](#)
- [使用 Amazon IVS 聊天示例 AWS CLI](#)
- [使用 Amazon IVS 实时直播示例 AWS CLI](#)
- [使用 Amazon Kendra 的示例 AWS CLI](#)
- [使用的 Kinesis 示例 AWS CLI](#)

- [AWS KMS 使用示例 AWS CLI](#)
- [使用 Lake Formation 示例 AWS CLI](#)
- [使用 Lambda 示例 AWS CLI](#)
- [使用的 License Manager 示例 AWS CLI](#)
- [使用 Lightsail 示例 AWS CLI](#)
- [使用 Macie 的示例 AWS CLI](#)
- [使用 Amazon Managed Grafana 示例 AWS CLI](#)
- [MediaConnect 使用示例 AWS CLI](#)
- [MediaConvert 使用示例 AWS CLI](#)
- [MediaLive 使用示例 AWS CLI](#)
- [MediaPackage 使用示例 AWS CLI](#)
- [MediaPackage 使用的 VOD 示例 AWS CLI](#)
- [MediaStore 使用数据平面示例 AWS CLI](#)
- [MediaTailor 使用示例 AWS CLI](#)
- [使用的 MemoryDB 示例 AWS CLI](#)
- [使用 Amazon MSK 示例 AWS CLI](#)
- [使用网络管理器示例 AWS CLI](#)
- [使用灵活的工作室示例 AWS CLI](#)
- [OpenSearch 使用的服务示例 AWS CLI](#)
- [AWS OpsWorks 使用示例 AWS CLI](#)
- [AWS OpsWorks CM 使用示例 AWS CLI](#)
- [使用 Organizati AWS CLI](#)
- [AWS Outposts 使用示例 AWS CLI](#)
- [AWS Payment Cryptography 使用示例 AWS CLI](#)
- [AWS Payment Cryptography 使用数据平面示例 AWS CLI](#)
- [使用 Amazon Pinpoint 示例 AWS CLI](#)
- [使用 Amazon Polly 的示例 AWS CLI](#)
- [AWS 价目表 使用示例 AWS CLI](#)
- [AWS Private CA 使用示例 AWS CLI](#)

- [AWS Proton 使用示例 AWS CLI](#)
- [使用的 QLDB 示例 AWS CLI](#)
- [使用 Amazon RDS 示例 AWS CLI](#)
- [使用 Amazon RDS 数据服务示例 AWS CLI](#)
- [使用 Amazon RDS 性能见解示例 AWS CLI](#)
- [使用 Amazon Redshift 示例 AWS CLI](#)
- [使用 Amazon Rekognition 的示例 AWS CLI](#)
- [AWS RAM 使用示例 AWS CLI](#)
- [使用资源管理器示例 AWS CLI](#)
- [使用的 Resource Groups AWS CLI](#)
- [Resource Groups 使用标记 API 示例 AWS CLI](#)
- [AWS RoboMaker 使用示例 AWS CLI](#)
- [使用 Route 53 示例 AWS CLI](#)
- [使用 Route 53 的域名注册示例 AWS CLI](#)
- [使用 Route 53 的解析器示例 AWS CLI](#)
- [使用 Amazon S3 的示例 AWS CLI](#)
- [使用 Amazon S3 控制示例 AWS CLI](#)
- [S3 Glacier 示例使用 AWS CLI](#)
- [使用的 Secrets Manager 示例 AWS CLI](#)
- [使用的 Security Hub 示例 AWS CLI](#)
- [AWS Serverless Application Repository 使用示例 AWS CLI](#)
- [使用 Service Catalog 示例 AWS CLI](#)
- [使用 Service Quotas 示例 AWS CLI](#)
- [使用 Amazon SES 示例 AWS CLI](#)
- [使用 Shield 示例 AWS CLI](#)
- [签名者使用示例 AWS CLI](#)
- [使用 Snowball 的示例 AWS CLI](#)
- [使用 Amazon SNS 示例 AWS CLI](#)
- [使用亚马逊 SQS 示例 AWS CLI](#)

- [使用 Storage Gateway 示例 AWS CLI](#)
- [AWS STS 使用示例 AWS CLI](#)
- [AWS Support 使用示例 AWS CLI](#)
- [使用 Amazon SWF 示例 AWS CLI](#)
- [使用 Systems Manager 示例 AWS CLI](#)
- [使用 Amazon Textract 的示例 AWS CLI](#)
- [使用 Amazon Transcribe 示例 AWS CLI](#)
- [Amazon Translate 示例使用 AWS CLI](#)
- [Trusted Advisor 使用示例 AWS CLI](#)
- [使用验证权限示例 AWS CLI](#)
- [使用的 VPC 莱迪思示例 AWS CLI](#)
- [AWS WAF Classic 使用示例 AWS CLI](#)
- [AWS WAF Classic Regional 使用示例 AWS CLI](#)
- [AWS WAFV2 使用示例 AWS CLI](#)
- [使用亚马逊的 WorkDocs 示例 AWS CLI](#)
- [使用亚马逊的 WorkMail 示例 AWS CLI](#)
- [使用的 Amazon WorkMail 消息流示例 AWS CLI](#)
- [WorkSpaces 使用示例 AWS CLI](#)
- [使用的 X-ray 示例 AWS CLI](#)

## 使用的 ACM 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 ACM 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### **add-tags-to-certificate**

以下代码示例演示了如何使用 `add-tags-to-certificate`。

#### AWS CLI

向现有 ACM 证书添加标签

以下 `add-tags-to-certificate` 命令向指定证书添加两个标签。使用空格分隔多个标签：

```
aws acm add-tags-to-certificate --certificate-arn
arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012 --tags
Key=Admin,Value=Alice Key=Purpose,Value=Website
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddTagsToCertificate](#)中的。

### **delete-certificate**

以下代码示例演示了如何使用 `delete-certificate`。

#### AWS CLI

从账户中删除 ACM 证书

以下 `delete-certificate` 命令删除具有指定 ARN 的证书：

```
aws acm delete-certificate --certificate-arn
arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCertificate](#)中的。

### **describe-certificate**

以下代码示例演示了如何使用 `describe-certificate`。

## AWS CLI

### 检索 ACM 证书中包含的字段

以下 `describe-certificate` 命令检索具有指定 ARN 的证书的所有字段：

```
aws acm describe-certificate --certificate-arn
arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012
```

输出类似于以下内容：

```
{
  "Certificate": {
    "CertificateArn":
"arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012",
    "CreatedAt": 1446835267.0,
    "DomainName": "www.example.com",
    "DomainValidationOptions": [
      {
        "DomainName": "www.example.com",
        "ValidationDomain": "www.example.com",
        "ValidationEmails": [
          "hostmaster@example.com",
          "admin@example.com",
          "owner@example.com.whoisprivacyservice.org",
          "tech@example.com.whoisprivacyservice.org",
          "admin@example.com.whoisprivacyservice.org",
          "postmaster@example.com",
          "webmaster@example.com",
          "administrator@example.com"
        ]
      },
      {
        "DomainName": "www.example.net",
        "ValidationDomain": "www.example.net",
        "ValidationEmails": [
          "postmaster@example.net",
          "admin@example.net",
          "owner@example.net.whoisprivacyservice.org",
          "tech@example.net.whoisprivacyservice.org",
          "admin@example.net.whoisprivacyservice.org",
          "hostmaster@example.net",
          "administrator@example.net",

```



```

        "webmaster@example.net"
    ]
}
],
"InUseBy": [],
"IssuedAt": 1446835815.0,
"Issuer": "Amazon",
"KeyAlgorithm": "RSA-2048",
"NotAfter": 1478433600.0,
"NotBefore": 1446768000.0,
"Serial": "0f:ac:b0:a3:8d:ea:65:52:2d:7d:01:3a:39:36:db:d6",
"SignatureAlgorithm": "SHA256WITHRSA",
>Status": "ISSUED",
"Subject": "CN=www.example.com",
"SubjectAlternativeNames": [
    "www.example.com",
    "www.example.net"
]
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCertificate](#)中的。

## export-certificate

以下代码示例演示了如何使用 export-certificate。

### AWS CLI

导出私有 CA 颁发的私有证书。

以下export-certificate命令将私有证书、证书链和私钥导出到您的显示屏上：

```
aws acm export-certificate --certificate-arn
arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012 --
passphrase file://path-to-passphrase-file
```

要将证书、证书链和私钥导出到本地文件，请使用以下命令：

```
aws acm export-certificate --certificate-arn
arn:aws:acm:region:sccount:certificate/12345678-1234-1234-1234-123456789012 --
passphrase file://path-to-passphrase-file > c:\temp\export.txt
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ExportCertificate](#) 中的。

## get-certificate

以下代码示例演示了如何使用 `get-certificate`。

### AWS CLI

#### 检索 ACM 证书

以下 `get-certificate` 命令检索指定 ARN 的证书和证书链：

```
aws acm get-certificate --certificate-arn
arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012
```

输出类似于以下内容：

```
{
  "Certificate": "-----BEGIN CERTIFICATE-----
MIICiTCCAFICCCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAKGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBA5TC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWVxHmAd
BgkqhkiG9w0BCQEWEG5vb25lQGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAKGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC0lBTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWVxHmAdBgkqhkiG9w0BCQEWEG5vb25lQGft
YXpvbi5jb20wZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLygVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnczvQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvjx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----",
  "CertificateChain": "-----BEGIN CERTIFICATE-----
MIICiTCCAFICCCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAKGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBA5TC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWVxHmAd
BgkqhkiG9w0BCQEWEG5vb25lQGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAKGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC0lBTSBDb25z
```

```

b2x1MRIwEAYDVQQDEw1UZsXN0Q21sYWMxHzAdBgkqhkiG9w0BCQEWEG5vb251QGft
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySwTc2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZncvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVvxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFbjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----",
"-----BEGIN CERTIFICATE-----
MIICiTCcAfICcQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xZDASBgNVBASTC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZsXN0Q21sYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xZDASBgNVBASTC01BTSBDb25z
b2x1MRIwEAYDVQQDEw1UZsXN0Q21sYWMxHzAdBgkqhkiG9w0BCQEWEG5vb251QGft
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySwTc2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZncvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVvxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFbjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----",
"-----BEGIN CERTIFICATE-----
MIICiTCcAfICcQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xZDASBgNVBASTC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZsXN0Q21sYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xZDASBgNVBASTC01BTSBDb25z
b2x1MRIwEAYDVQQDEw1UZsXN0Q21sYWMxHzAdBgkqhkiG9w0BCQEWEG5vb251QGft
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySwTc2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZncvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVvxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFbjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCertificate](#)中的。

## import-certificate

以下代码示例演示了如何使用 `import-certificate`。

### AWS CLI

将证书导入 ACM。

以下 `import-certificate` 命令将证书导入 ACM。将文件名替换为您自己的文件名：

```
aws acm import-certificate --certificate file://Certificate.pem --certificate-chain
file://CertificateChain.pem --private-key file://PrivateKey.pem
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ImportCertificate](#) 中的。

## list-certificates

以下代码示例演示了如何使用 `list-certificates`。

### AWS CLI

列出账户的 ACM 证书 AWS

以下 `list-certificates` 命令列出了您账户中证书的 ARN：

```
aws acm list-certificates
```

上述命令生成类似于以下内容的输出：

```
{
  "CertificateSummaryList": [
    {
      "CertificateArn":
"arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012",
      "DomainName": "www.example.com"
    },
    {
      "CertificateArn": "arn:aws:acm:region:account:certificate/aaaaaaaa-bbbb-
cccc-dddd-eeeeeeeeeeee",
      "DomainName": "www.example.net"
    }
  ]
}
```

```
]
}
```

您可以决定每次调用 `list-certificates` 时要显示多少证书。例如，如果您有四个证书，且希望每次显示的证书不超过两个，请将 `max-items` 参数设置为 2，如下例所示：

```
aws acm list-certificates --max-items 2
```

将显示两个证书 ARN 和 `NextToken` 值：

```
"CertificateSummaryList": [
  {
    "CertificateArn": "arn:aws:acm:region:account: \
      certificate/12345678-1234-1234-1234-123456789012",
    "DomainName": "www.example.com"
  },
  {
    "CertificateArn": "arn:aws:acm:region:account: \
      certificate/aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "DomainName": "www.example.net"
  }
],
"NextToken": "9f4d9f69-275a-41fe-b58e-2b837bd9ba48"
```

要在您的账户中显示接下来的两个证书，请在下次调用中设置此 `NextToken` 值：

```
aws acm list-certificates --max-items 2 --next-token 9f4d9f69-275a-41fe-
b58e-2b837bd9ba48
```

您可以使用 `certificate-statuses` 参数筛选输出。以下命令显示处于 `PENDING_VALIDATION` 状态的证书：

```
aws acm list-certificates --certificate-statuses PENDING_VALIDATION
```

您也可以使用 `includes` 参数筛选输出。以下命令显示根据以下属性筛选的证书。要显示的证书：

- Specify that the RSA algorithm and a 2048 bit key are used to generate key pairs.
- Contain a Key Usage extension that specifies that the certificates can be used to create digital signatures.

```
- Contain an Extended Key Usage extension that specifies that the certificates can be used for code signing.
```

```
aws acm list-certificates --max-items 10 --includes
extendedKeyUsage=CODE_SIGNING,keyUsage=DIGITAL_SIGNATURE,keyTypes=RSA_2048
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCertificates](#)中的。

## list-tags-for-certificate

以下代码示例演示了如何使用 `list-tags-for-certificate`。

### AWS CLI

列出应用于 ACM 证书的标签

以下 `list-tags-for-certificate` 命令列出了应用于您账户中证书的标签：

```
aws acm list-tags-for-certificate --certificate-arn
arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012
```

上述命令生成类似于以下内容的输出：

```
{
  "Tags": [
    {
      "Value": "Website",
      "Key": "Purpose"
    },
    {
      "Value": "Alice",
      "Key": "Admin"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForCertificate](#)中的。

## remove-tags-from-certificate

以下代码示例演示了如何使用 `remove-tags-from-certificate`。

## AWS CLI

### 删除 ACM 证书的标签

以下 `remove-tags-from-certificate` 命令删除了指定证书的两个标签。使用空格分隔多个标签：

```
aws acm remove-tags-from-certificate --certificate-arn
arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012 --tags
Key=Admin,Value=Alice Key=Purpose,Value=Website
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RemoveTagsFromCertificate](#) 中的。

## **request-certificate**

以下代码示例演示了如何使用 `request-certificate`。

## AWS CLI

### 请求新的 ACM 证书

以下 `request-certificate` 命令使用 DNS 验证为 `www.example.com` 域请求新证书：

```
aws acm request-certificate --domain-name www.example.com --validation-method DNS
```

您可以输入幂等性令牌，以区分对 `request-certificate` 的调用：

```
aws acm request-certificate --domain-name www.example.com --validation-method DNS --
idempotency-token 91adc45q
```

您可以输入一个或多个主题备用名称来请求能够保护多个顶级域的证书：

```
aws acm request-certificate --domain-name example.com --validation-method DNS --
idempotency-token 91adc45q --subject-alternative-names www.example.net
```

您可以输入一个备用名称，该名称也可用于访问您的网站：

```
aws acm request-certificate --domain-name example.com --validation-method DNS --
idempotency-token 91adc45q --subject-alternative-names www.example.com
```

您可以使用星号 ( \* ) 作为通配符为同一域中的多个子域创建证书：

```
aws acm request-certificate --domain-name example.com --validation-method DNS --
idempotency-token 91adc45q --subject-alternative-names *.example.com
```

您也可以输入多个备用名称：

```
aws acm request-certificate --domain-name example.com --validation-method DNS --
subject-alternative-names b.example.com c.example.com d.example.com
```

如果您使用电子邮件进行验证，则可以输入域验证选项来指定将验证电子邮件发送到的域：

```
aws acm request-certificate --domain-name example.com --validation-method
EMAIL --subject-alternative-names www.example.com --domain-validation-options
DomainName=example.com,ValidationDomain=example.com
```

以下命令会在您请求新证书时选择退出证书透明度日志：

```
aws acm request-certificate --domain-name www.example.com --validation-method DNS --
options CertificateTransparencyLoggingPreference=DISABLED --idempotency-token 184627
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RequestCertificate](#)中的。

## resend-validation-email

以下代码示例演示了如何使用 resend-validation-email。

### AWS CLI

重新发送 ACM 证书请求的验证电子邮件

以下 resend-validation-email 命令指示 Amazon 证书颁发机构向相应的地址发送验证电子邮件：

```
aws acm resend-validation-email --certificate-arn
arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012 --
domain www.example.com --validation-domain example.com
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ResendValidationEmail](#)中的。



## update-certificate-options

以下代码示例演示了如何使用 update-certificate-options。

### AWS CLI

#### 更新证书选项

以下update-certificate-options命令可选择退出证书透明度日志：

```
aws acm update-certificate-options --certificate-arn
arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012 --
options CertificateTransparencyLoggingPreference=DISABLED
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateCertificateOptions](#)中的。

## 使用 API Gateway 示例 AWS CLI

以下代码示例向您展示了如何使用 with API Gateway 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### create-api-key

以下代码示例演示了如何使用 create-api-key。

### AWS CLI

创建为现有 API 和舞台启用的 API 密钥

**命令:**

```
aws apigateway create-api-key --name 'Dev API Key' --description 'Used for development' --enabled --stage-keys restApiId='a1b2c3d4e5',stageName='dev'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateApiKey](#)中的。

**create-authorizer**

以下代码示例演示了如何使用 create-authorizer。

**AWS CLI**

示例 1：为 API 创建基于令牌的 API Gateway 自定义授权器

以下create-authorizer示例创建了一个基于令牌的授权者。

```
aws apigateway create-authorizer \  
  --rest-api-id 1234123412 \  
  --name 'First-Token-Custom-Authorizer' \  
  --type TOKEN \  
  --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/  
arn:aws:lambda:us-west-2:123412341234:function:customAuthFunction/invocations' \  
  --identity-source 'method.request.header.Authorization' \  
  --authorizer-result-ttl-in-seconds 300
```

**输出：**

```
{  
  "authType": "custom",  
  "name": "First-Token-Custom-Authorizer",  
  "authorizerUri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/  
arn:aws:lambda:us-west-2:123412341234:function:customAuthFunction/invocations",  
  "authorizerResultTtlInSeconds": 300,  
  "identitySource": "method.request.header.Authorization",  
  "type": "TOKEN",  
  "id": "z40xj0"  
}
```

示例 2：为该 API 创建基于 Cognito 用户池的 API Gateway 自定义授权器

以下create-authorizer示例创建了基于 Cognito 用户池的 API Gateway 自定义授权器。

```
aws apigateway create-authorizer \  
  --rest-api-id 1234123412 \  
  --name 'First_Cognito_Custom_Authorizer' \  
  --type COGNITO_USER_POOLS \  
  --provider-arns 'arn:aws:cognito-idp:us-east-1:123412341234:userpool/us-  
east-1_aWcZeQbuD' \  
  --identity-source 'method.request.header.Authorization'
```

输出：

```
{  
  "authType": "cognito_user_pools",  
  "identitySource": "method.request.header.Authorization",  
  "name": "First_Cognito_Custom_Authorizer",  
  "providerARNs": [  
    "arn:aws:cognito-idp:us-east-1:342398297714:userpool/us-east-1_qWbZzQhzE"  
  ],  
  "type": "COGNITO_USER_POOLS",  
  "id": "5yidl1t"  
}
```

示例 3：为 API 创建基于请求的 API Gateway 自定义授权器

以下 `create-authorizer` 示例创建了一个基于请求的授权者。

```
aws apigateway create-authorizer \  
  --rest-api-id 1234123412 \  
  --name 'First_Request_Custom_Authorizer' \  
  --type REQUEST \  
  --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/  
arn:aws:lambda:us-west-2:123412341234:function:customAuthFunction/invocations' \  
  --identity-source 'method.request.header.Authorization,context.accountId' \  
  --authorizer-result-ttl-in-seconds 300
```

输出：

```
{  
  "id": "z40xj0",  
  "name": "First_Request_Custom_Authorizer",  
  "type": "REQUEST",  
  "authType": "custom",
```

```
"authorizerUri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/arn:aws:lambda:us-west-2:123412341234:function:customAuthFunction/invocations",
"identitySource": "method.request.header.Authorization,context.accountId",
"authorizerResultTtlInSeconds": 300
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAuthorizer](#)中的。

## create-base-path-mapping

以下代码示例演示了如何使用 create-base-path-mapping。

### AWS CLI

为自定义域名创建基本路径映射

命令:

```
aws apigateway create-base-path-mapping --domain-name subdomain.domain.tld --rest-api-id 1234123412 --stage prod --base-path v1
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateBasePathMapping](#)中的。

## create-deployment

以下代码示例演示了如何使用 create-deployment。

### AWS CLI

将为 API 配置的资源部署到新阶段

命令:

```
aws apigateway create-deployment --rest-api-id 1234123412 --stage-name dev --stage-description 'Development Stage' --description 'First deployment to the dev stage'
```

将为 API 配置的资源部署到现有阶段

命令:

```
aws apigateway create-deployment --rest-api-id 1234123412 --stage-name dev --description 'Second deployment to the dev stage'
```

通过 Stage 变量将为 API 配置的资源部署到现有阶段

`aws apigateway 创建部署 — rest-api-id 1234123412 — stage-name dev — 描述 “第三次部署到开发阶段” — variables key='value' , otherkey='otherValue' , otherkey='otherValue'`

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDeployment](#)中的。

## create-domain-name

以下代码示例演示了如何使用 `create-domain-name`。

### AWS CLI

创建自定义域名

命令:

```
aws apigateway create-domain-name --domain-name 'my.domain.tld' --
certificate-name 'my.domain.tld cert' --certificate-arn 'arn:aws:acm:us-
east-1:012345678910:certificate/fb1b9770-a305-495d-aefb-27e5e101ff3'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDomainName](#)中的。

## create-model

以下代码示例演示了如何使用 `create-model`。

### AWS CLI

为 API 创建模型

命令:

```
aws apigateway create-model --rest-api-id 1234123412 --name 'firstModel' --
description 'The First Model' --content-type 'application/json' --schema
'{"$schema": "http://json-schema.org/draft-04/schema#", "title": "firstModel",
"type": "object", "properties": { "firstProperty" : { "type": "object",
"properties": { "key": { "type": "string" } } } } }'
```

输出:

```
{
  "contentType": "application/json",
```

```
"description": "The First Model",
"name": "firstModel",
"id": "2rzg01",
"schema": "{ \"$schema\": \"http://json-schema.org/draft-04/schema#\", \"title\": \"firstModel\", \"type\": \"object\", \"properties\": { \"firstProperty\": { \"type\": \"object\", \"properties\": { \"key\": { \"type\": \"string\" } } } } } }
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateModel](#)中的。

## create-resource

以下代码示例演示了如何使用 `create-resource`。

### AWS CLI

在 API 中创建资源

命令:

```
aws apigateway create-resource --rest-api-id 1234123412 --parent-id a1b2c3 --path-part 'new-resource'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateResource](#)中的。

## create-rest-api

以下代码示例演示了如何使用 `create-rest-api`。

### AWS CLI

创建 API

命令:

```
aws apigateway create-rest-api --name 'My First API' --description 'This is my first API'
```

使用现有 API 创建重复的 API

命令:

```
aws apigateway create-rest-api --name 'Copy of My First API' --description 'This is a copy of my first API' --clone-from 1234123412
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRestApi](#)中的。

## create-stage

以下代码示例演示了如何使用 create-stage。

### AWS CLI

在 API 中创建包含现有部署的阶段

命令:

```
aws apigateway create-stage --rest-api-id 1234123412 --stage-name 'dev' --description 'Development stage' --deployment-id a1b2c3
```

在 API 中创建包含现有部署和自定义阶段变量的阶段

命令:

```
aws apigateway create-stage --rest-api-id 1234123412 --stage-name 'dev' --description 'Development stage' --deployment-id a1b2c3 --variables key='value',otherKey='otherValue'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateStage](#)中的。

## create-usage-plan-key

以下代码示例演示了如何使用 create-usage-plan-key。

### AWS CLI

将现有 API 密钥与使用计划关联

命令:

```
aws apigateway create-usage-plan-key --usage-plan-id a1b2c3 --key-type "API_KEY" --key-id 4vq3yryqm5
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUsagePlanKey](#)中的。

## create-usage-plan

以下代码示例演示了如何使用 create-usage-plan。

### AWS CLI

创建具有限制和配额限制的使用计划，并在月初重置

命令:

```
aws apigateway create-usage-plan --name "New Usage Plan" --description "A new usage plan" --throttle burstLimit=10,rateLimit=5 --quota limit=500,offset=0,period=MONTH
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUsagePlan](#)中的。

## delete-api-key

以下代码示例演示了如何使用 delete-api-key。

### AWS CLI

删除 API 密钥

命令:

```
aws apigateway delete-api-key --api-key 8bk1k8b11k3sB38D9B310enyWT8c09B301kq0b1k
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteApiKey](#)中的。

## delete-authorizer

以下代码示例演示了如何使用 delete-authorizer。

### AWS CLI

在 API 中删除自定义授权方

命令:



```
aws apigateway delete-authorizer --rest-api-id 1234123412 --authorizer-id 7gkfbo
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAuthorizer](#)中的。

## delete-base-path-mapping

以下代码示例演示了如何使用 delete-base-path-mapping。

### AWS CLI

删除自定义域名的基本路径映射

命令:

```
aws apigateway delete-base-path-mapping --domain-name 'api.domain.tld' --base-path 'dev'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBasePathMapping](#)中的。

## delete-client-certificate

以下代码示例演示了如何使用 delete-client-certificate。

### AWS CLI

删除客户证书

命令:

```
aws apigateway delete-client-certificate --client-certificate-id a1b2c3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteClientCertificate](#)中的。

## delete-deployment

以下代码示例演示了如何使用 delete-deployment。

### AWS CLI

在 API 中删除部署

命令:

```
aws apigateway delete-deployment --rest-api-id 1234123412 --deployment-id a1b2c3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDeployment](#)中的。

## delete-domain-name

以下代码示例演示了如何使用 delete-domain-name。

AWS CLI

删除自定义域名

命令:

```
aws apigateway delete-domain-name --domain-name 'api.domain.tld'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDomainName](#)中的。

## delete-integration-response

以下代码示例演示了如何使用 delete-integration-response。

AWS CLI

删除 API 中给定资源、方法和状态代码的集成响应

命令:

```
aws apigateway delete-integration-response --rest-api-id 1234123412 --resource-id a1b2c3 --http-method GET --status-code 200
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteIntegrationResponse](#)中的。

## delete-integration

以下代码示例演示了如何使用 delete-integration。

## AWS CLI

删除 API 中给定资源和方法的集成

命令:

```
aws apigateway delete-integration --rest-api-id 1234123412 --resource-id a1b2c3 --http-method GET
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteIntegration](#)中的。

## delete-method-response

以下代码示例演示了如何使用 delete-method-response。

### AWS CLI

删除 API 中给定资源、方法和状态代码的方法响应

命令:

```
aws apigateway delete-method-response --rest-api-id 1234123412 --resource-id a1b2c3 --http-method GET --status-code 200
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMethodResponse](#)中的。

## delete-method

以下代码示例演示了如何使用 delete-method。

### AWS CLI

在 API 中删除给定资源的方法

命令:

```
aws apigateway delete-method --rest-api-id 1234123412 --resource-id a1b2c3 --http-method GET
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMethod](#)中的。

## delete-model

以下代码示例演示了如何使用 delete-model。

### AWS CLI

在给定 API 中删除模型

命令:

```
aws apigateway delete-model --rest-api-id 1234123412 --model-name 'customModel'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteModel](#)中的。

## delete-resource

以下代码示例演示了如何使用 delete-resource。

### AWS CLI

删除 API 中的资源

命令:

```
aws apigateway delete-resource --rest-api-id 1234123412 --resource-id a1b2c3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteResource](#)中的。

## delete-rest-api

以下代码示例演示了如何使用 delete-rest-api。

### AWS CLI

删除 API

命令:

```
aws apigateway delete-rest-api --rest-api-id 1234123412
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRestApi](#)中的。

## delete-stage

以下代码示例演示了如何使用 delete-stage。

### AWS CLI

在 API 中删除阶段

命令:

```
aws apigateway delete-stage --rest-api-id 1234123412 --stage-name 'dev'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteStage](#)中的。

## delete-usage-plan-key

以下代码示例演示了如何使用 delete-usage-plan-key。

### AWS CLI

从使用计划中删除 API 密钥

命令:

```
aws apigateway delete-usage-plan-key --usage-plan-id a1b2c3 --key-id  
1NbjQzMReAkeEQPNAW8r3dXsU2rDD7fc7f2Sipnu
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUsagePlanKey](#)中的。

## delete-usage-plan

以下代码示例演示了如何使用 delete-usage-plan。

### AWS CLI

删除使用计划

命令:

```
aws apigateway delete-usage-plan --usage-plan-id a1b2c3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUsagePlan](#)中的。

## flush-stage-authorizers-cache

以下代码示例演示了如何使用 flush-stage-authorizers-cache。

### AWS CLI

刷新舞台上的所有授权者缓存条目

命令:

```
aws apigateway flush-stage-authorizers-cache --rest-api-id 1234123412 --stage-name dev
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[FlushStageAuthorizersCache](#)中的。

## flush-stage-cache

以下代码示例演示了如何使用 flush-stage-cache。

### AWS CLI

刷新 API 阶段的缓存

命令:

```
aws apigateway flush-stage-cache --rest-api-id 1234123412 --stage-name dev
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[FlushStageCache](#)中的。

## generate-client-certificate

以下代码示例演示了如何使用 generate-client-certificate。

### AWS CLI

创建客户端 SSL 证书

命令:

```
aws apigateway generate-client-certificate --description 'My First Client Certificate'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GenerateClientCertificate](#)中的。

## get-account

以下代码示例演示了如何使用 get-account。

### AWS CLI

获取 API Gateway 账户设置

命令:

```
aws apigateway get-account
```

输出:

```
{
  "cloudwatchRoleArn": "arn:aws:iam::123412341234:role/
APIGatewayToCloudWatchLogsRole",
  "throttleSettings": {
    "rateLimit": 500.0,
    "burstLimit": 1000
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAccount](#)中的。

## get-api-key

以下代码示例演示了如何使用 get-api-key。

### AWS CLI

获取有关特定 API 密钥的信息

命令:

```
aws apigateway get-api-key --api-key 8bk1k8b11k3sB38D9B310enyWT8c09B301kq0b1k
```

输出:

```
{
  "description": "My first key",
  "enabled": true,
  "stageKeys": [
    "a1b2c3d4e5/dev",
    "e5d4c3b2a1/dev"
  ],
  "lastUpdatedDate": 1456184515,
  "createdDate": 1456184452,
  "id": "8bk1k8b11k3sB38D9B310enyWT8c09B301kq0b1k",
  "name": "My key"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetApiKey](#)中的。

## get-api-keys

以下代码示例演示了如何使用 get-api-keys。

### AWS CLI

获取 API 密钥列表

命令:

```
aws apigateway get-api-keys
```

输出:

```
{
  "items": [
    {
      "description": "My first key",
      "enabled": true,
      "stageKeys": [
        "a1b2c3d4e5/dev",
        "e5d4c3b2a1/dev"
      ],
      "lastUpdatedDate": 1456184515,
      "createdDate": 1456184452,
      "id": "8bk1k8b11k3sB38D9B310enyWT8c09B301kq0b1k",
    }
  ]
}
```



```
        "name": "My key"
      }
    ]
  }
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetApiKeys](#)中的。

## get-authorizer

以下代码示例演示了如何使用 get-authorizer。

### AWS CLI

获取每个 API 授权方的 API Gateway 设置

命令:

```
aws apigateway get-authorizer --rest-api-id 1234123412 --authorizer-id gfi4n3
```

输出:

```
{
  "authorizerResultTtlInSeconds": 300,
  "name": "MyAuthorizer",
  "type": "TOKEN",
  "identitySource": "method.request.header.Authorization",
  "authorizerUri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/arn:aws:lambda:us-west-2:123412341234:function:authorizer_function/invocations",
  "id": "gfi4n3"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAuthorizer](#)中的。

## get-authorizers

以下代码示例演示了如何使用 get-authorizers。

### AWS CLI

获取 REST API 的授权者列表

命令:

```
aws apigateway get-authorizers --rest-api-id 1234123412
```

输出:

```
{
  "items": [
    {
      "name": "MyAuthorizer",
      "authorizerUri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-west-2:123412341234:function:My_Authorizer_Function/
invocations",
      "authorizerResultTtlInSeconds": 300,
      "identitySource": "method.request.header.Authorization",
      "type": "TOKEN",
      "id": "gfi4n3"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAuthorizers](#)中的。

## get-base-path-mapping

以下代码示例演示了如何使用 `get-base-path-mapping`。

AWS CLI

获取自定义域名的基础路径映射

命令:

```
aws apigateway get-base-path-mapping --domain-name subdomain.domain.tld --base-path
v1
```

输出:

```
{
  "basePath": "v1",
```

```
"restApiId": "1234w4321e",
"stage": "api"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBasePathMapping](#)中的。

## get-base-path-mappings

以下代码示例演示了如何使用 get-base-path-mappings。

### AWS CLI

获取自定义域名的基本路径映射

命令：

```
aws apigateway get-base-path-mappings --domain-name subdomain.domain.tld
```

输出：

```
{
  "items": [
    {
      "basePath": "(none)",
      "restApiId": "1234w4321e",
      "stage": "dev"
    },
    {
      "basePath": "v1",
      "restApiId": "1234w4321e",
      "stage": "api"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBasePathMappings](#)中的。

## get-client-certificate

以下代码示例演示了如何使用 get-client-certificate。

## AWS CLI

获取客户证书

命令:

```
aws apigateway get-client-certificate --client-certificate-id a1b2c3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetClientCertificate](#)中的。

## get-client-certificates

以下代码示例演示了如何使用 get-client-certificates。

## AWS CLI

获取客户证书列表

命令:

```
aws apigateway get-client-certificates
```

输出:

```
{
  "items": [
    {
      "pemEncodedCertificate": "-----BEGIN CERTIFICATE----- <certificate
content> -----END CERTIFICATE-----",
      "clientCertificateId": "a1b2c3",
      "expirationDate": 1483556561,
      "description": "My Client Certificate",
      "createdDate": 1452020561
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetClientCertificates](#)中的。

## get-deployment

以下代码示例演示了如何使用 get-deployment。

## AWS CLI

获取有关部署的信息

命令:

```
aws apigateway get-deployment --rest-api-id 1234123412 --deployment-id ztt4m2
```

输出:

```
{
  "description": "myDeployment",
  "id": "ztt4m2",
  "createdDate": 1455218022
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDeployment](#)中的。

## get-deployments

以下代码示例演示了如何使用 get-deployments。

### AWS CLI

获取 REST API 的部署列表

命令:

```
aws apigateway get-deployments --rest-api-id 1234123412
```

输出:

```
{
  "items": [
    {
      "createdDate": 1453797217,
      "id": "0a2b4c",
      "description": "Deployed my API for the first time"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDeployments](#)中的。

## get-domain-name

以下代码示例演示了如何使用 get-domain-name。

### AWS CLI

获取有关自定义域名的信息

命令:

```
aws apigateway get-domain-name --domain-name api.domain.tld
```

输出:

```
{
  "domainName": "api.domain.tld",
  "distributionDomainName": "d1a2f3a4c5o6d.cloudfront.net",
  "certificateName": "uploadedCertificate",
  "certificateUploadDate": 1462565487
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDomainName](#)中的。

## get-domain-names

以下代码示例演示了如何使用 get-domain-names。

### AWS CLI

获取自定义域名列表

命令:

```
aws apigateway get-domain-names
```

输出:

```
{
```

```
"items": [  
  {  
    "distributionDomainName": "d9511k3109bkd.cloudfront.net",  
    "certificateUploadDate": 1452812505,  
    "certificateName": "my_custom_domain-certificate",  
    "domainName": "subdomain.domain.tld"  
  }  
]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDomainNames](#)中的。

## get-export

以下代码示例演示了如何使用 get-export。

### AWS CLI

获取舞台的 JSON Swagger 模板

命令:

```
aws apigateway get-export --rest-api-id a1b2c3d4e5 --stage-name dev --export-type  
swagger /path/to/filename.json
```

获取舞台的 JSON Swagger 模板 + API Gateway 扩展

命令:

```
aws apigateway get-export --parameters extensions='integrations' --rest-api-id  
a1b2c3d4e5 --stage-name dev --export-type swagger /path/to/filename.json
```

要获取 JSON Swagger 模板 + 舞台的 Postman 扩展

命令:

```
aws apigateway get-export --parameters extensions='postman' --rest-api-id a1b2c3d4e5  
--stage-name dev --export-type swagger /path/to/filename.json
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetExport](#)中的。

## get-integration-response

以下代码示例演示了如何使用 `get-integration-response`。

### AWS CLI

获取在 REST API 资源下定义的 HTTP 方法的集成响应配置

命令:

```
aws apigateway get-integration-response --rest-api-id 1234123412 --resource-id
y9h6rt --http-method GET --status-code 200
```

输出:

```
{
  "statusCode": "200",
  "responseTemplates": {
    "application/json": null
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetIntegrationResponse](#)中的。

## get-integration

以下代码示例演示了如何使用 `get-integration`。

### AWS CLI

获取在 REST API 资源下定义的 HTTP 方法的集成配置

命令:

```
aws apigateway get-integration --rest-api-id 1234123412 --resource-id y9h6rt --http-
method GET
```

输出:

```
{
```



```
"httpMethod": "POST",
"integrationResponses": {
  "200": {
    "responseTemplates": {
      "application/json": null
    },
    "statusCode": "200"
  }
},
"cacheKeyParameters": [],
"type": "AWS",
"uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123412341234:function:My_Function/invocations",
"cacheNamespace": "y9h6rt"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetIntegration](#)中的。

## get-method-response

以下代码示例演示了如何使用 get-method-response。

### AWS CLI

获取在 REST API 资源下定义的 HTTP 方法的方法响应资源配置

命令:

```
aws apigateway get-method-response --rest-api-id 1234123412 --resource-id y9h6rt --
http-method GET --status-code 200
```

输出:

```
{
  "responseModels": {
    "application/json": "Empty"
  },
  "statusCode": "200"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMethodResponse](#)中的。

## get-method

以下代码示例演示了如何使用 get-method。

### AWS CLI

获取在 REST API 资源下定义的 HTTP 方法的方法资源配置

命令:

```
aws apigateway get-method --rest-api-id 1234123412 --resource-id y9h6rt --http-method GET
```

输出:

```
{
  "apiKeyRequired": false,
  "httpMethod": "GET",
  "methodIntegration": {
    "integrationResponses": {
      "200": {
        "responseTemplates": {
          "application/json": null
        },
        "statusCode": "200"
      }
    },
    "cacheKeyParameters": [],
    "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/arn:aws:lambda:us-west-2:123412341234:function:My_Function/invocations",
    "httpMethod": "POST",
    "cacheNamespace": "y9h6rt",
    "type": "AWS"
  },
  "requestParameters": {},
  "methodResponses": {
    "200": {
      "responseModels": {
        "application/json": "Empty"
      },
      "statusCode": "200"
    }
  }
},
```

```
"authorizationType": "NONE"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMethod](#)中的。

## get-model-template

以下代码示例演示了如何使用 `get-model-template`。

### AWS CLI

获取在 REST API 下定义的模型的映射模板

命令:

```
aws apigateway get-model-template --rest-api-id 1234123412 --model-name Empty
```

输出 :

```
{
  "value": "#set($inputRoot = $input.path('$'))\n{ }"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetModelTemplate](#)中的。

## get-model

以下代码示例演示了如何使用 `get-model`。

### AWS CLI

获取在 REST API 下定义的模型的配置

命令:

```
aws apigateway get-model --rest-api-id 1234123412 --model-name Empty
```

输出 :

```
{
  "contentType": "application/json",
```

```

    "description": "This is a default empty schema model",
    "name": "Empty",
    "id": "etd5w5",
    "schema": "{\n  \"schema\": \"http://json-schema.org/draft-04/schema#\",
  \"title\": \"Empty Schema\",
  \"type\": \"object\"}"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetModel](#)中的。

## get-models

以下代码示例演示了如何使用 get-models。

### AWS CLI

获取 REST API 的模型列表

命令:

```
aws apigateway get-models --rest-api-id 1234123412
```

输出:

```

{
  "items": [
    {
      "description": "This is a default error schema model",
      "schema": "{\n  \"schema\": \"http://json-schema.org/draft-04/schema#\",
  \"title\": \"Error Schema\",
  \"type\": \"object\",
  \"properties\": {
    \"message\": { \"type\": \"string\" }
  }",
      "contentType": "application/json",
      "id": "7tpbze",
      "name": "Error"
    },
    {
      "description": "This is a default empty schema model",
      "schema": "{\n  \"schema\": \"http://json-schema.org/draft-04/schema#\",
  \"title\": \"Empty Schema\",
  \"type\": \"object\"}",
      "contentType": "application/json",
      "id": "etd5w5",
      "name": "Empty"
    }
  ]
}

```

```
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetModels](#)中的。

## get-resource

以下代码示例演示了如何使用 `get-resource`。

### AWS CLI

获取有关资源的信息

命令:

```
aws apigateway get-resource --rest-api-id 1234123412 --resource-id zwo0y3
```

输出 :

```
{
  "path": "/path",
  "pathPart": "path",
  "id": "zwo0y3",
  "parentId": "uyokt6ij2g"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetResource](#)中的。

## get-resources

以下代码示例演示了如何使用 `get-resources`。

### AWS CLI

获取 REST API 的资源列表

命令:

```
aws apigateway get-resources --rest-api-id 1234123412
```

输出 :

```
{
  "items": [
    {
      "path": "/resource/subresource",
      "resourceMethods": {
        "POST": {}
      },
      "id": "024ace",
      "pathPart": "subresource",
      "parentId": "ai5b02"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetResources](#)中的。

## get-rest-api

以下代码示例演示了如何使用 get-rest-api。

### AWS CLI

获取有关 API 的信息

命令:

```
aws apigateway get-rest-api --rest-api-id 1234123412
```

输出:

```
{
  "name": "myAPI",
  "id": "o1y243m4f5",
  "createdDate": 1453416433
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRestApi](#)中的。

## get-rest-apis

以下代码示例演示了如何使用 get-rest-apis。

## AWS CLI

### 获取 REST API 列表

#### 命令:

```
aws apigateway get-rest-apis
```

#### 输出:

```
{
  "items": [
    {
      "createdDate": 1438884790,
      "id": "12s44z21rb",
      "name": "My First API"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRestApis](#)中的。

## get-sdk

以下代码示例演示了如何使用 get-sdk。

## AWS CLI

### 获取适用于 REST API 阶段的安卓 SDK

#### 命令:

```
aws apigateway get-sdk --rest-api-id 1234123412 --stage-name dev --sdk-type android
--parameters
groupId='com.mycompany',invokerPackage='com.mycompany.clientsdk',artifactId='Mycompany-
client',artifactVersion='1.0.0' /path/to/android_sdk.zip
```

#### 输出:

```
{
  "contentType": "application/octet-stream",
  "contentDisposition": "attachment; filename=\"android_2016-02-22_23-52Z.zip\""
}
```

```
}
```

## 获取适用于 REST API 阶段的 IOS 开发工具包

命令:

```
aws apigateway get-sdk --rest-api-id 1234123412 --stage-name dev --sdk-type
  objectivec --parameters classPrefix='myprefix' /path/to/iOS_sdk.zip
```

输出:

```
{
  "contentType": "application/octet-stream",
  "contentDisposition": "attachment; filename=\"objectivec_2016-02-22_23-52Z.zip\"
}"
```

## 获取适用于 REST API 阶段的 Javascript 开发工具包

命令:

```
aws apigateway get-sdk --rest-api-id 1234123412 --stage-name dev --sdk-type
  javascript /path/to/javascript_sdk.zip
```

输出:

```
{
  "contentType": "application/octet-stream",
  "contentDisposition": "attachment; filename=\"javascript_2016-02-22_23-52Z.zip\"
}"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSdk](#)中的。

## get-stage

以下代码示例演示了如何使用 get-stage。

### AWS CLI

获取有关 API 阶段的信息



## 命令:

```
aws apigateway get-stage --rest-api-id 1234123412 --stage-name dev
```

## 输出:

```
{
  "stageName": "dev",
  "cacheClusterSize": "0.5",
  "cacheClusterEnabled": false,
  "cacheClusterStatus": "NOT_AVAILABLE",
  "deploymentId": "rbh1fj",
  "lastUpdatedDate": 1466802961,
  "createdDate": 1460682074,
  "methodSettings": {
    "*/*": {
      "cacheTtlInSeconds": 300,
      "loggingLevel": "INFO",
      "dataTraceEnabled": false,
      "metricsEnabled": true,
      "unauthorizedCacheControlHeaderStrategy":
"SUCCEED_WITH_RESPONSE_HEADER",
      "throttlingRateLimit": 500.0,
      "cacheDataEncrypted": false,
      "cachingEnabled": false,
      "throttlingBurstLimit": 1000,
      "requireAuthorizationForCacheControl": true
    },
    "~1resource/GET": {
      "cacheTtlInSeconds": 300,
      "loggingLevel": "INFO",
      "dataTraceEnabled": false,
      "metricsEnabled": true,
      "unauthorizedCacheControlHeaderStrategy":
"SUCCEED_WITH_RESPONSE_HEADER",
      "throttlingRateLimit": 500.0,
      "cacheDataEncrypted": false,
      "cachingEnabled": false,
      "throttlingBurstLimit": 1000,
      "requireAuthorizationForCacheControl": true
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetStage](#)中的。

## get-stages

以下代码示例演示了如何使用 get-stages。

### AWS CLI

获取 REST API 的阶段列表

命令:

```
aws apigateway get-stages --rest-api-id 1234123412
```

输出:

```
{
  "item": [
    {
      "stageName": "dev",
      "cacheClusterSize": "0.5",
      "cacheClusterEnabled": true,
      "cacheClusterStatus": "AVAILABLE",
      "deploymentId": "123h64",
      "lastUpdatedDate": 1456185138,
      "createdDate": 1453589092,
      "methodSettings": {
        "~1resource~1subresource/POST": {
          "cacheTtlInSeconds": 300,
          "loggingLevel": "INFO",
          "dataTraceEnabled": true,
          "metricsEnabled": true,
          "throttlingRateLimit": 500.0,
          "cacheDataEncrypted": false,
          "cachingEnabled": false,
          "throttlingBurstLimit": 1000
        }
      }
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetStages](#)中的。

## get-usage-plan-key

以下代码示例演示了如何使用 get-usage-plan-key。

### AWS CLI

获取与使用计划关联的 API 密钥的详细信息

命令:

```
aws apigateway get-usage-plan-key --usage-plan-id a1b2c3 --key-id
1NbjQzMReAkeEQPNAW8r3dXsU2rDD7fc7f2Sipnu
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetUsagePlanKey](#)中的。

## get-usage-plan-keys

以下代码示例演示了如何使用 get-usage-plan-keys。

### AWS CLI

获取与使用计划关联的 API 密钥列表

命令:

```
aws apigateway get-usage-plan-keys --usage-plan-id a1b2c3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetUsagePlanKeys](#)中的。

## get-usage-plan

以下代码示例演示了如何使用 get-usage-plan。

### AWS CLI

获取使用计划的详细信息

命令:

```
aws apigateway get-usage-plan --usage-plan-id a1b2c3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetUsagePlan](#)中的。

## get-usage-plans

以下代码示例演示了如何使用 `get-usage-plans`。

### AWS CLI

要获取所有使用计划的详细信息

命令:

```
aws apigateway get-usage-plans
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetUsagePlans](#)中的。

## get-usage

以下代码示例演示了如何使用 `get-usage`。

### AWS CLI

获取使用计划的使用详情

命令:

```
aws apigateway get-usage --usage-plan-id a1b2c3 --start-date "2016-08-16" --end-date "2016-08-17"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetUsage](#)中的。

## import-rest-api

以下代码示例演示了如何使用 `import-rest-api`。

### AWS CLI

导入 Swagger 模板并创建 API

命令:

```
aws apigateway import-rest-api --body 'file:///path/to/API_Swagger_template.json'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ImportRestApi](#)中的。

## put-integration-response

以下代码示例演示了如何使用 put-integration-response。

AWS CLI

使用定义的映射模板创建集成响应作为默认响应

命令:

```
aws apigateway put-integration-response --rest-api-id 1234123412 --resource-id a1b2c3 --http-method GET --status-code 200 --selection-pattern "" --response-templates '{"application/json": "{\"json\": \"template\"}"}'
```

使用正则表达式为 400 的静态定义的标头值创建集成响应

命令:

```
aws apigateway put-integration-response --rest-api-id 1234123412 --resource-id a1b2c3 --http-method GET --status-code 400 --selection-pattern 400 --response-parameters '{"method.response.header.custom-header": ""}'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutIntegrationResponse](#)中的。

## put-integration

以下代码示例演示了如何使用 put-integration。

AWS CLI

创建 MOCK 集成请求

命令:

```
aws apigateway put-integration --rest-api-id 1234123412 --resource-id a1b2c3 --http-method GET --type MOCK --request-templates '{ "application/json": "{\\"statusCode\\": 200}" }'
```

## 创建 HTTP 集成请求

命令:

```
aws apigateway put-integration --rest-api-id 1234123412 --resource-id a1b2c3 --http-method GET --type HTTP --integration-http-method GET --uri 'https://domain.tld/path'
```

## 使用 Lambda 函数终端节点创建 AWS 集成请求

命令:

```
aws apigateway put-integration --rest-api-id 1234123412 --resource-id a1b2c3 --http-method GET --type AWS --integration-http-method POST --uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/arn:aws:lambda:us-west-2:123412341234:function:function_name/invocations'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutIntegration](#)中的。

## put-method-response

以下代码示例演示了如何使用 put-method-response。

### AWS CLI

基于指定状态代码，通过自定义方法响应标头创建方法响应

命令:

```
aws apigateway put-method-response --rest-api-id 1234123412 --resource-id a1b2c3 --http-method GET --status-code 400 --response-parameters "method.response.header.custom-header=false"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutMethodResponse](#)中的。

## put-method

以下代码示例演示了如何使用 put-method。

## AWS CLI

在没有授权、没有 API 密钥和自定义方法请求标头的情况下为 API 中的资源创建方法

命令:

```
aws apigateway put-method --rest-api-id 1234123412 --resource-id a1b2c3 --http-method PUT --authorization-type "NONE" --no-api-key-required --request-parameters "method.request.header.custom-header=false"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutMethod](#)中的。

### put-rest-api

以下代码示例演示了如何使用 put-rest-api。

## AWS CLI

使用 Swagger 模板覆盖现有 API

命令:

```
aws apigateway put-rest-api --rest-api-id 1234123412 --mode overwrite --body 'fileb:///path/to/API_Swagger_template.json'
```

将 Swagger 模板合并到现有 API 中

命令:

```
aws apigateway put-rest-api --rest-api-id 1234123412 --mode merge --body 'fileb:///path/to/API_Swagger_template.json'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutRestApi](#)中的。

### test-invoke-authorizer

以下代码示例演示了如何使用 test-invoke-authorizer。

## AWS CLI

要进行测试，请调用包含所需标头和值的自定义授权器的请求

命令:

```
aws apigateway test-invoke-authorizer --rest-api-id 1234123412 --authorizer-id
5yid1t --headers Authorization='Value'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TestInvokeAuthorizer](#)中的。

## test-invoke-method

以下代码示例演示了如何使用 test-invoke-method。

### AWS CLI

通过发出 GET 请求在 API 中测试调用根资源

命令:

```
aws apigateway test-invoke-method --rest-api-id 1234123412 --resource-id av15sg8fw8
--http-method GET --path-with-query-string '/'
```

通过发出指定路径参数值的 GET 请求在 API 中测试调用子资源

命令:

```
aws apigateway test-invoke-method --rest-api-id 1234123412 --resource-id 3gapai --
http-method GET --path-with-query-string '/pets/1'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TestInvokeMethod](#)中的。

## update-account

以下代码示例演示了如何使用 update-account。

### AWS CLI

更改用于登录到日志的 IAM 角色 ARN CloudWatch

命令:

```
aws apigateway update-account --patch-operations op='replace',path='/
cloudwatchRoleArn',value='arn:aws:iam::123412341234:role/APIGatewayToCloudWatchLogs'
```



输出：

```
{
  "cloudwatchRoleArn": "arn:aws:iam::123412341234:role/
APIGatewayToCloudWatchLogs",
  "throttleSettings": {
    "rateLimit": 1000.0,
    "burstLimit": 2000
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAccount](#)中的。

## update-api-key

以下代码示例演示了如何使用 update-api-key。

### AWS CLI

更改 API 密钥的名称

命令：

```
aws apigateway update-api-key --api-key sNvjQDMReA1eEQPNAW8r37XsU2rDD7fc7m2SiMnu --
patch-operations op='replace',path='/name',value='newName'
```

输出：

```
{
  "description": "currentDescription",
  "enabled": true,
  "stageKeys": [
    "41t2j324r5/dev"
  ],
  "lastUpdatedDate": 1470086052,
  "createdDate": 1445460347,
  "id": "sNvjQDMReA1vEQPNzW8r3dXsU2rrD7fcjm2SiMnu",
  "name": "newName"
}
```

禁用 API 密钥

**命令:**

```
aws apigateway update-api-key --api-key sNvjQDMReA1eEQPNAW8r37XsU2rDD7fc7m2SiMnu --
patch-operations op='replace',path='/enabled',value='false'
```

**输出 :**

```
{
  "description": "currentDescription",
  "enabled": false,
  "stageKeys": [
    "41t2j324r5/dev"
  ],
  "lastUpdatedDate": 1470086052,
  "createdDate": 1445460347,
  "id": "sNvjQDMReA1vEQPNzW8r3dXsU2rrD7fcjm2SiMnu",
  "name": "newName"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateApiKey](#)中的。

**update-authorizer**

以下代码示例演示了如何使用 update-authorizer。

**AWS CLI****更改自定义授权者的名称****命令:**

```
aws apigateway update-authorizer --rest-api-id 1234123412 --authorizer-id gfi4n3 --
patch-operations op='replace',path='/name',value='testAuthorizer'
```

**输出 :**

```
{
  "authType": "custom",
  "name": "testAuthorizer",
  "authorizerUri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123412341234:function:customAuthorizer/invocations",
  "authorizerResultTtlInSeconds": 300,
}
```

```
"identitySource": "method.request.header.Authorization",
"type": "TOKEN",
"id": "gfi4n3"
}
```

## 更改由自定义授权方调用的 Lambda 函数

### 命令:

```
aws apigateway update-authorizer --rest-api-id 1234123412 --authorizer-id gfi4n3 --
patch-operations op='replace',path='/authorizerUri',value='arn:aws:apigateway:us-
west-2:lambda:path/2015-03-31/functions/arn:aws:lambda:us-
west-2:123412341234:function:newAuthorizer/invocations'
```

### 输出 :

```
{
  "authType": "custom",
  "name": "testAuthorizer",
  "authorizerUri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123412341234:function:newAuthorizer/invocations",
  "authorizerResultTtlInSeconds": 300,
  "identitySource": "method.request.header.Authorization",
  "type": "TOKEN",
  "id": "gfi4n3"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAuthorizer](#)中的。

## update-base-path-mapping

以下代码示例演示了如何使用 update-base-path-mapping。

### AWS CLI

#### 更改自定义域名的基础路径

### 命令:

```
aws apigateway update-base-path-mapping --domain-name api.domain.tld --base-path
prod --patch-operations op='replace',path='/basePath',value='v1'
```

输出：

```
{
  "basePath": "v1",
  "restApiId": "1234123412",
  "stage": "api"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateBasePathMapping](#)中的。

## update-client-certificate

以下代码示例演示了如何使用 `update-client-certificate`。

AWS CLI

更新客户证书的描述

命令：

```
aws apigateway update-client-certificate --client-certificate-id a1b2c3 --patch-operations op='replace',path='/description',value='My new description'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateClientCertificate](#)中的。

## update-deployment

以下代码示例演示了如何使用 `update-deployment`。

AWS CLI

更改部署的描述

命令：

```
aws apigateway update-deployment --rest-api-id 1234123412 --deployment-id ztt4m2 --patch-operations op='replace',path='/description',value='newDescription'
```

输出：

```
{
```

```
"description": "newDescription",
"id": "ztt4m2",
"createdDate": 1455218022
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDeployment](#)中的。

## update-domain-name

以下代码示例演示了如何使用 update-domain-name。

### AWS CLI

更改自定义域名的证书名称

以下update-domain-name示例更改了自定义域的证书名称。

```
aws apigateway update-domain-name \
  --domain-name api.domain.tld \
  --patch-operations op='replace',path='/certificateArn',value='arn:aws:acm:us-
west-2:111122223333:certificate/CERTEXAMPLE123EXAMPLE'
```

输出：

```
{
  "domainName": "api.domain.tld",
  "distributionDomainName": "d123456789012.cloudfront.net",
  "certificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/
CERTEXAMPLE123EXAMPLE",
  "certificateUploadDate": 1462565487
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的在 API Gateway 中为 API [设置自定义域名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDomainName](#)中的。

## update-integration-response

以下代码示例演示了如何使用 update-integration-response。

## AWS CLI

将集成响应标头更改为具有 '\*' 的静态映射

命令:

```
aws apigateway update-integration-response --rest-api-id 1234123412 --
resource-id 3gapai --http-method GET --status-code 200 --patch-operations
  op='replace',path='/responseParameters/method.response.header.Access-Control-Allow-
Origin',value='''''*''''
```

输出:

```
{
  "statusCode": "200",
  "responseParameters": {
    "method.response.header.Access-Control-Allow-Origin": "'*'"
  }
}
```

移除集成响应标头

命令:

```
aws apigateway update-integration-response --rest-api-id 1234123412 --resource-id
3gapai --http-method GET --status-code 200 --patch-operations op='remove',path='/
responseParameters/method.response.header.Access-Control-Allow-Origin'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateIntegrationResponse](#)中的。

## update-integration

以下代码示例演示了如何使用 update-integration。

## AWS CLI

添加配置了输入直通的“内容类型：应用程序/json”映射模板

命令:

```
aws apigateway update-integration \
```

```
--rest-api-id a1b2c3d4e5 \  
--resource-id a1b2c3 \  
--http-method POST \  
--patch-operations "op='add',path='/requestTemplates/application~1json'"
```

更新 ( 替换 ) 使用自定义模板配置的“内容类型：应用程序/json”映射模板

命令:

```
aws apigateway update-integration \  
  --rest-api-id a1b2c3d4e5 \  
  --resource-id a1b2c3 \  
  --http-method POST \  
  --patch-operations "op='replace',path='/requestTemplates/  
application~1json',value='{\"example\": \"json\"}'"
```

使用输入直通更新 ( 替换 ) 与“内容类型：应用程序/json”关联的自定义模板

命令:

```
aws apigateway update-integration \  
  --rest-api-id a1b2c3d4e5 \  
  --resource-id a1b2c3 \  
  --http-method POST \  
  --patch-operations "op='replace',path='requestTemplates/application~1json'"
```

移除“内容类型：应用程序/json”映射模板

命令:

```
aws apigateway update-integration \  
  --rest-api-id a1b2c3d4e5 \  
  --resource-id a1b2c3 \  
  --http-method POST \  
  --patch-operations "op='remove',path='/requestTemplates/application~1json'"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateIntegration](#)中的。

## update-method-response

以下代码示例演示了如何使用 update-method-response。

## AWS CLI

为方法中的 200 响应创建新的方法响应标头并将其定义为不需要 (默认)

命令:

```
aws apigateway update-method-response --rest-api-id 1234123412 --resource-id
a1b2c3 --http-method GET --status-code 200 --patch-operations op="add",path="/
responseParameters/method.response.header.custom-header",value="false"
```

在方法中删除 200 响应的响应模型

命令:

```
aws apigateway update-method-response --rest-api-id 1234123412 --resource-id
a1b2c3 --http-method GET --status-code 200 --patch-operations op="remove",path="/
responseModels/application~1json"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateMethodResponse](#)中的。

## update-method

以下代码示例演示了如何使用 update-method。

### AWS CLI

示例 1：将方法修改为需要 API 密钥

以下update-method示例将该方法修改为需要 API 密钥。

```
aws apigateway update-method \
  --rest-api-id 1234123412 \
  --resource-id a1b2c3 \
  --http-method GET \
  --patch-operations op="replace",path="/apiKeyRequired",value="true"
```

输出：

```
{
  "httpMethod": "GET",
  "authorizationType": "NONE",
  "apiKeyRequired": true,
```



```

    "methodResponses": {
      "200": {
        "statusCode": "200",
        "responseModels": {}
      }
    },
    "methodIntegration": {
      "type": "AWS",
      "httpMethod": "POST",
      "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789111:function:hello-world/invocations",
      "passthroughBehavior": "WHEN_NO_MATCH",
      "contentHandling": "CONVERT_TO_TEXT",
      "timeoutInMillis": 29000,
      "cacheNamespace": "h7i8j9",
      "cacheKeyParameters": [],
      "integrationResponses": {
        "200": {
          "statusCode": "200",
          "responseTemplates": {}
        }
      }
    }
  }
}

```

## 示例 2：将方法修改为要求 IAM 授权

以下update-method示例将方法修改为需要 IAM 授权。

```

aws apigateway update-method \
  --rest-api-id 1234123412 \
  --resource-id a1b2c3 \
  --http-method GET \
  --patch-operations op="replace",path="/authorizationType",value="AWS_IAM"

```

输出：

```

{
  "httpMethod": "GET",
  "authorizationType": "AWS_IAM",
  "apiKeyRequired": false,
  "methodResponses": {
    "200": {

```

```
        "statusCode": "200",
        "responseModels": {}
    }
},
"methodIntegration": {
    "type": "AWS",
    "httpMethod": "POST",
    "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789111:function:hello-world/invocations",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "contentHandling": "CONVERT_TO_TEXT",
    "timeoutInMillis": 29000,
    "cacheNamespace": "h7i8j9",
    "cacheKeyParameters": [],
    "integrationResponses": {
        "200": {
            "statusCode": "200",
            "responseTemplates": {}
        }
    }
}
}
```

### 示例 3：将方法修改为需要 Lambda 授权

以下 update-method 示例将该方法修改为所需的 Lambda 授权。

```
aws apigateway update-method --rest-api-id 1234123412 \  
  --resource-id a1b2c3 \  
  --http-method GET \  
  --patch-operations op="replace",path="/authorizationType",value="CUSTOM" \  
  op="replace",path="/authorizerId",value="e4f5g6"
```

输出：

```
{
  "httpMethod": "GET",
  "authorizationType": "CUSTOM",
  "authorizerId": "e4f5g6",
  "apiKeyRequired": false,
  "methodResponses": {
    "200": {
      "statusCode": "200",
```

```

        "responseModels": {}
    }
},
"methodIntegration": {
    "type": "AWS",
    "httpMethod": "POST",
    "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789111:function:hello-world/invocations",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "contentHandling": "CONVERT_TO_TEXT",
    "timeoutInMillis": 29000,
    "cacheNamespace": "h7i8j9",
    "cacheKeyParameters": [],
    "integrationResponses": {
        "200": {
            "statusCode": "200",
            "responseTemplates": {}
        }
    }
}
}
}
}

```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的[使用 API Gateway CLI 和 REST API 创建、配置和测试使用计划](#)以及在 API Gateway 中控制和管理 REST API 的[访问权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateMethod](#)中的。

## update-model

以下代码示例演示了如何使用 update-model。

### AWS CLI

在 API 中更改模型的描述

命令:

```
aws apigateway update-model --rest-api-id 1234123412 --model-name 'Empty' --patch-operations op=replace,path=/description,value='New Description'
```

在 API 中更改模型的架构

命令:

```
aws apigateway update-model --rest-api-id 1234123412 --model-name 'Empty' --patch-operations op=replace,path=/schema,value='{ \"$schema\": \"http://json-schema.org/draft-04/schema#\", \"title\": \"Empty Schema\", \"type\": \"object\" }'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateModel](#)中的。

## update-resource

以下代码示例演示了如何使用 update-resource。

### AWS CLI

在 API 中移动资源并将其置于不同的父资源下

命令:

```
aws apigateway update-resource --rest-api-id 1234123412 --resource-id 1a2b3c --patch-operations op=replace,path=/parentId,value='3c2b1a'
```

输出:

```
{
  "path": "/resource",
  "pathPart": "resource",
  "id": "1a2b3c",
  "parentId": "3c2b1a"
}
```

在 API 中重命名资源 (pathPart)

命令:

```
aws apigateway update-resource --rest-api-id 1234123412 --resource-id 1a2b3c --patch-operations op=replace,path=/pathPart,value=newresourceName
```

输出:

```
{
  "path": "/newresourceName",
  "pathPart": "newresourceName",
}
```

```
"id": "1a2b3c",  
"parentId": "3c2b1a"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateResource](#)中的。

## update-rest-api

以下代码示例演示了如何使用 `update-rest-api`。

### AWS CLI

更改 API 的名称

命令:

```
aws apigateway update-rest-api --rest-api-id 1234123412 --patch-operations  
op=replace,path=/name,value='New Name'
```

更改 API 的描述

命令:

```
aws apigateway update-rest-api --rest-api-id 1234123412 --patch-operations  
op=replace,path=/description,value='New Description'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRestApi](#)中的。

## update-stage

以下代码示例演示了如何使用 `update-stage`。

### AWS CLI

示例 1：覆盖资源和方法的阶段设置

以下`update-stage`示例覆盖了阶段设置并关闭了特定资源和方法的完整请求/响应日志记录。

```
aws apigateway update-stage \  
--rest-api-id 1234123412 \  
--stage-name 'dev' \  
--patch-operations op=replace,path=/logs,value='false'
```

```
--patch-operations op=replace,path=~1resourceName/GET/logging/
dataTrace,value=false
```

输出：

```
{
  "deploymentId": "5bd17",
  "stageName": "dev",
  "cacheClusterEnabled": false,
  "cacheClusterStatus": "NOT_AVAILABLE",
  "methodSettings": {
    "~1resourceName/GET": {
      "metricsEnabled": false,
      "dataTraceEnabled": false,
      "throttlingBurstLimit": 5000,
      "throttlingRateLimit": 10000.0,
      "cachingEnabled": false,
      "cacheTtlInSeconds": 300,
      "cacheDataEncrypted": false,
      "requireAuthorizationForCacheControl": true,
      "unauthorizedCacheControlHeaderStrategy": "SUCCEED_WITH_RESPONSE_HEADER"
    }
  },
  "tracingEnabled": false,
  "createdDate": "2022-07-18T10:11:18-07:00",
  "lastUpdatedDate": "2022-07-18T10:19:04-07:00"
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的为 REST API [设置阶段](#)。

示例 2：更新 API 阶段的所有资源和方法的阶段设置

以下update-stage示例为 API 阶段的所有资源和方法开启了完整的请求/响应日志记录。

```
aws apigateway update-stage \
  --rest-api-id 1234123412 \
  --stage-name 'dev' \
  --patch-operations 'op=replace,path=/*/*/logging/dataTrace,value=true'
```

输出：

```
{
```

```
"deploymentId": "5ubd17",
"stageName": "dev",
"cacheClusterEnabled": false,
"cacheClusterStatus": "NOT_AVAILABLE",
"methodSettings": {
  "/*/*": {
    "metricsEnabled": false,
    "dataTraceEnabled": true,
    "throttlingBurstLimit": 5000,
    "throttlingRateLimit": 10000.0,
    "cachingEnabled": false,
    "cacheTtlInSeconds": 300,
    "cacheDataEncrypted": false,
    "requireAuthorizationForCacheControl": true,
    "unauthorizedCacheControlHeaderStrategy": "SUCCEED_WITH_RESPONSE_HEADER"
  }
},
"tracingEnabled": false,
"createdDate": "2022-07-18T10:11:18-07:00",
"lastUpdatedDate": "2022-07-18T10:31:04-07:00"
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的为 REST API [设置阶段](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateStage](#) 中的。

## update-usage-plan

以下代码示例演示了如何使用 update-usage-plan。

### AWS CLI

更改使用计划中定义的期限

命令:

```
aws apigateway update-usage-plan --usage-plan-id a1b2c3 --patch-operations
op="replace",path="/quota/period",value="MONTH"
```

更改使用计划中定义的配额限制

命令:

```
aws apigateway update-usage-plan --usage-plan-id a1b2c3 --patch-operations
  op="replace",path="/quota/limit",value="500"
```

更改使用计划中定义的油门速率限制

命令:

```
aws apigateway update-usage-plan --usage-plan-id a1b2c3 --patch-operations
  op="replace",path="/throttle/rateLimit",value="10"
```

更改使用计划中定义的油门爆发限制

命令:

```
aws apigateway update-usage-plan --usage-plan-id a1b2c3 --patch-operations
  op="replace",path="/throttle/burstLimit",value="20"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUsagePlan](#)中的。

## update-usage

以下代码示例演示了如何使用 update-usage。

### AWS CLI

临时修改使用计划中定义的当前期间 API 密钥的配额

命令:

```
aws apigateway update-usage --usage-plan-id a1b2c3 --key-id
  1NbjQzMReAkeEQPNAW8r3dXsU2rDD7fc7f2Sipnu --patch-operations op="replace",path="/
  remaining",value="50"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUsage](#)中的。

## API Gateway HTTP 和 WebSocket API 示例使用 AWS CLI

以下代码示例向您展示了如何使用 API Gateway HTTP 和 API 来执行操作和 WebSocket 实现常见场景。AWS Command Line Interface



操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-api-mapping**

以下代码示例演示了如何使用 create-api-mapping。

AWS CLI

为 API 创建 API 映射

以下create-api-mapping示例将 API 的test阶段映射到regional.example.com自定义域名的/myApi路径。

```
aws apigatewayv2 create-api-mapping \  
  --domain-name regional.example.com \  
  --api-mapping-key myApi \  
  --api-id a1b2c3d4 \  
  --stage test
```

输出：

```
{  
  "ApiId": "a1b2c3d4",  
  "ApiMappingId": "0qzs2sy7bh",  
  "ApiMappingKey": "myApi"  
  "Stage": "test"  
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的在 API Gateway [y 中设置区域自定义域名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateApiMapping](#)中的。

## create-api

以下代码示例演示了如何使用 create-api。

### AWS CLI

#### 创建 HTTP API

以下create-api示例使用快速创建创建了一个 HTTP API。您可以使用快速创建来创建具有 AWS Lambda 或 HTTP 集成、默认包罗万象的路由以及配置为自动部署更改的默认阶段的 API。以下命令使用快速创建来创建与 Lambda 函数集成的 HTTP API。

```
aws apigatewayv2 create-api \  
  --name my-http-api \  
  --protocol-type HTTP \  
  --target arn:aws:lambda:us-west-2:123456789012:function:my-lambda-function
```

输出：

```
{  
  "ApiEndpoint": "https://a1b2c3d4.execute-api.us-west-2.amazonaws.com",  
  "ApiId": "a1b2c3d4",  
  "ApiKeySelectionExpression": "$request.header.x-api-key",  
  "CreateDate": "2020-04-08T19:05:45+00:00",  
  "Name": "my-http-api",  
  "ProtocolType": "HTTP",  
  "RouteSelectionExpression": "$request.method $request.path"  
}
```

有关更多信息，请参阅 Amazon [API Gateway 开发者指南中的在 API Gateway 中开发 HTTP API](#)。

#### 创建 WebSocket API

以下create-api示例创建了一个具有指定名称的 WebSocket API。

```
aws apigatewayv2 create-api \  
  --name "myWebSocketApi" \  
  --protocol-type WEBSOCKET \  
  --route-selection-expression '$request.body.action'
```

输出：

```
{
  "ApiKeySelectionExpression": "$request.header.x-api-key",
  "Name": "myWebSocketApi",
  "CreateDate": "2018-11-15T06:23:51Z",
  "ProtocolType": "WEBSOCKET",
  "RouteSelectionExpression": "'$request.body.action'",
  "ApiId": "aabbccdde"
}
```

有关更多信息，请参阅亚马逊 [WebSocket API Gateway 开发者指南](#) 中的 [在 API Gateway 中创建 API](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateApi](#) 中的。

## create-authorizer

以下代码示例演示了如何使用 create-authorizer。

### AWS CLI

为 HTTP API 创建 JWT 授权方

以下 create-authorizer 示例创建了一个使用 Amazon Cognito 作为身份提供者的 JWT 授权方。

```
aws apigatewayv2 create-authorizer \
  --name my-jwt-authorizer \
  --api-id a1b2c3d4 \
  --authorizer-type JWT \
  --identity-source '$request.header.Authorization' \
  --jwt-configuration Audience=123456abc,Issuer=https://cognito-idp.us-
west-2.amazonaws.com/us-west-2_abc123
```

输出：

```
{
  "AuthorizerId": "a1b2c3",
  "AuthorizerType": "JWT",
  "IdentitySource": [
```

```
    "$request.header.Authorization"
  ],
  "JwtConfiguration": {
    "Audience": [
      "123456abc"
    ],
    "Issuer": "https://cognito-idp.us-west-2.amazonaws.com/us-west-2_abc123"
  },
  "Name": "my-jwt-authorizer"
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 JWT 授权方控制 HTTP API [的访问权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAuthorizer](#)中的。

## create-deployment

以下代码示例演示了如何使用 create-deployment。

### AWS CLI

为 API 创建部署

以下create-deployment示例为 API 创建部署，并将该部署与 API 的dev阶段相关联。

```
aws apigatewayv2 create-deployment \  
  --api-id a1b2c3d4 \  
  --stage-name dev
```

输出：

```
{  
  "AutoDeployed": false,  
  "CreateDate": "2020-04-06T23:38:08Z",  
  "DeploymentId": "531z91",  
  "DeploymentStatus": "DEPLOYED"  
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的 API [部署](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDeployment](#)中的。

## create-domain-name

以下代码示例演示了如何使用 create-domain-name。

### AWS CLI

#### 创建自定义域名

以下create-domain-name示例为 API 创建区域自定义域名。

```
aws apigatewayv2 create-domain-name \  
  --domain-name regional.example.com \  
  --domain-name-configurations CertificateArn=arn:aws:acm:us-  
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678
```

输出：

```
{  
  "ApiMappingSelectionExpression": "$request.basepath",  
  "DomainName": "regional.example.com",  
  "DomainNameConfigurations": [  
    {  
      "ApiGatewayDomainName": "d-id.execute-api.us-west-2.amazonaws.com",  
      "CertificateArn": "arn:aws:acm:us-  
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678",  
      "EndpointType": "REGIONAL",  
      "HostedZoneId": "123456789111",  
      "SecurityPolicy": "TLS_1_2",  
      "DomainNameStatus": "AVAILABLE"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的在 API Gateway [y 中设置区域自定义域名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDomainName](#)中的。

## create-integration

以下代码示例演示了如何使用 create-integration。

## AWS CLI

### 创建 WebSocket API 集成

以下create-integration示例为 WebSocket API 创建了一个模拟集成。

```
aws apigatewayv2 create-integration \  
  --api-id aabbccdde \\  
  --passthrough-behavior WHEN_NO_MATCH \  
  --timeout-in-millis 29000 \  
  --connection-type INTERNET \  
  --integration-type MOCK
```

输出：

```
{  
  "ConnectionType": "INTERNET",  
  "IntegrationId": "0abcdef",  
  "IntegrationResponseSelectionExpression": "${integration.response.statuscode}",  
  "IntegrationType": "MOCK",  
  "PassthroughBehavior": "WHEN_NO_MATCH",  
  "PayloadFormatVersion": "1.0",  
  "TimeoutInMillis": 29000  
}
```

有关更多信息，请参阅 Amazon WebSocket API Gateway 开发者指南中的在 API Gateway 中设置 API [集成请求](#)。

### 创建 HTTP API 集成

以下create-integration示例为 HTTP API 创建了 AWS Lambda 集成。

```
aws apigatewayv2 create-integration \  
  --api-id a1b2c3d4 \  
  --integration-type AWS_PROXY \  
  --integration-uri arn:aws:lambda:us-west-2:123456789012:function:my-function \  
  --payload-format-version 2.0
```

输出：

```
{
```

```
"ConnectionType": "INTERNET",
"IntegrationId": "0abcdef",
"IntegrationMethod": "POST",
"IntegrationType": "AWS_PROXY",
"IntegrationUri": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
"PayloadFormatVersion": "2.0",
"TimeoutInMillis": 30000
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的为 HTTP API [配置集成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateIntegration](#) 中的。

## create-route

以下代码示例演示了如何使用 create-route。

### AWS CLI

为 WebSocket 或 HTTP API 创建 \$default 路由

以下 create-route 示例为 WebSocket 或 HTTP API 创建了一个 \$default 条路由。

```
aws apigatewayv2 create-route \
  --api-id aabbccdde \
  --route-key '$default'
```

输出：

```
{
  "ApiKeyRequired": false,
  "AuthorizationType": "NONE",
  "RouteKey": "$default",
  "RouteId": "1122334"
}
```

有关更多信息，请参阅 Amazon WebSocket API Gateway 开发者指南中的使用 API [路由](#)

为 HTTP API 创建路由

以下 create-route 示例创建了一个接受 POST 请求 signup 的名为的路由。

```
aws apigatewayv2 create-route \  
  --api-id aabbccdde \\  
  --route-key 'POST /signup'
```

输出：

```
{  
  "ApiKeyRequired": false,  
  "AuthorizationType": "NONE",  
  "RouteKey": "POST /signup",  
  "RouteId": "1122334"  
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API [的路由](#)

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRoute](#)中的。

## create-stage

以下代码示例演示了如何使用 create-stage。

### AWS CLI

#### 创建舞台

以下create-stage示例为 API 创建了一个名为 dev 的阶段。

```
aws apigatewayv2 create-stage \  
  --api-id a1b2c3d4 \  
  --stage-name dev
```

输出：

```
{  
  "CreateDate": "2020-04-06T23:23:46Z",  
  "DefaultRouteSettings": {  
    "DetailedMetricsEnabled": false  
  },  
  "LastUpdatedDate": "2020-04-06T23:23:46Z",  
  "RouteSettings": {},  
}
```



```
"StageName": "dev",
"StageVariables": {},
"Tags": {}
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API [的阶段](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateStage](#) 中的。

## create-vpc-link

以下代码示例演示了如何使用 create-vpc-link。

### AWS CLI

为 HTTP API 创建 VPC 链接

以下 create-vpc-link 示例为 HTTP API 创建一个 VPC 链接。

```
aws apigatewayv2 create-vpc-link \
  --name MyVpcLink \
  --subnet-ids subnet-aaaa subnet-bbbb \
  --security-group-ids sg1234 sg5678
```

输出：

```
{
  "CreateDate": "2020-04-07T00:11:46Z",
  "Name": "MyVpcLink",
  "SecurityGroupIds": [
    "sg1234",
    "sg5678"
  ],
  "SubnetIds": [
    "subnet-aaaa",
    "subnet-bbbb"
  ],
  "Tags": {},
  "VpcLinkId": "abcd123",
  "VpcLinkStatus": "PENDING",
  "VpcLinkStatusMessage": "VPC link is provisioning ENIs",
  "VpcLinkVersion": "V2"
}
```

```
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API 的 [VPC 链接](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVpcLink](#)中的。

## delete-access-log-settings

以下代码示例演示了如何使用 delete-access-log-settings。

### AWS CLI

禁用 API 的访问日志记录

以下delete-access-log-settings示例删除了 API \$default 阶段的访问日志设置。要禁用某个阶段的访问日志记录，请删除其访问日志设置。

```
aws apigatewayv2 delete-access-log-settings \  
  --api-id a1b2c3d4 \  
  --stage-name '$default'
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的为 HTTP API [配置日志记录](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAccessLogSettings](#)中的。

## delete-api-mapping

以下代码示例演示了如何使用 delete-api-mapping。

### AWS CLI

删除 API 映射

以下delete-api-mapping示例删除了api.example.com自定义域名的 API 映射。

```
aws apigatewayv2 delete-api-mapping \  
  --api-mapping-id a1b2c3 \  
  --domain-name api.example.com
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的在 API Gateway [y 中设置区域自定义域名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteApiMapping](#)中的。

## delete-api

以下代码示例演示了如何使用 delete-api。

### AWS CLI

#### 删除 API

以下delete-api示例删除了一个 API。

```
aws apigatewayv2 delete-api \  
  --api-id a1b2c3d4
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon [API Gateway 开发者指南中的使用 WebSocket HTTP API 和使用 API](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteApi](#)中的。

## delete-authorizer

以下代码示例演示了如何使用 delete-authorizer。

### AWS CLI

#### 删除授权者

以下delete-authorizer示例删除授权方。

```
aws apigatewayv2 delete-authorizer \  
  --api-id a1b2c3d4 \  
  --authorizer-id a1b2c3
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 JWT 授权方控制 HTTP API [的访问权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAuthorizer](#)中的。

## delete-cors-configuration

以下代码示例演示了如何使用 delete-cors-configuration。

### AWS CLI

#### 删除 HTTP API 的 CORS 配置

以下delete-cors-configuration示例通过删除 HTTP API 的 CORS 配置来禁用 HTTP API 的跨域资源共享。

```
aws apigatewayv2 delete-cors-configuration \  
  --api-id a1b2c3d4
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的为 HTTP API [配置 CORS](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCorsConfiguration](#)中的。

## delete-deployment

以下代码示例演示了如何使用 delete-deployment。

### AWS CLI

#### 删除部署

以下delete-deployment示例删除了 API 的部署。

```
aws apigatewayv2 delete-deployment \  
  --api-id a1b2c3d4 \  
  --deployment-id a1b2c3
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的 [API 部署](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteDeployment](#) 中的。

## delete-domain-name

以下代码示例演示了如何使用 delete-domain-name。

### AWS CLI

删除自定义域名

以下 delete-domain-name 示例删除自定义域名。

```
aws apigatewayv2 delete-domain-name \  
  --domain-name api.example.com
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的在 API Gateway [y 中设置区域自定义域名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteDomainName](#) 中的。

## delete-integration

以下代码示例演示了如何使用 delete-integration。

### AWS CLI

删除集成

以下 delete-integration 示例删除了 API 集成。

```
aws apigatewayv2 delete-integration \  
  --api-id a1b2c3d4 \  
  --integration-id a1b2c3
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon [API Gateway 开发者指南中的配置 HTTP WebSocket API 的集成和设置 API 集成](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteIntegration](#)中的。

## delete-route-settings

以下代码示例演示了如何使用 delete-route-settings。

### AWS CLI

#### 删除路径设置

以下delete-route-settings示例删除指定路径的路径设置。

```
aws apigatewayv2 delete-route-settings \  
  --api-id a1b2c3d4 \  
  --stage-name dev \  
  --route-key 'GET /pets'
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API [的路由](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRouteSettings](#)中的。

## delete-route

以下代码示例演示了如何使用 delete-route。

### AWS CLI

#### 删除路线

以下delete-route示例删除了 API 路由。

```
aws apigatewayv2 delete-route \  
  --api-id a1b2c3d4 \  
  --route-id a1b2c3
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API [的路由](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRoute](#)中的。

## delete-stage

以下代码示例演示了如何使用 delete-stage。

### AWS CLI

#### 删除舞台

以下delete-stage示例删除了 API 的test阶段。

```
aws apigatewayv2 delete-stage \  
  --api-id a1b2c3d4 \  
  --stage-name test
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API [的阶段](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteStage](#)中的。

## delete-vpc-link

以下代码示例演示了如何使用 delete-vpc-link。

### AWS CLI

#### 删除 HTTP API 的 VPC 链接

以下delete-vpc-link示例删除 VPC 链接。

```
aws apigatewayv2 delete-vpc-link \  
  --vpc-link-id abcd123
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API 的 [VPC 链接](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVpcLink](#)中的。

## export-api

以下代码示例演示了如何使用 export-api。

### AWS CLI

导出 HTTP API 的 OpenAPI 定义

以下export-api示例将名为的 API 阶段的 OpenAPI 3.0 定义导出prod到名为的 YAML 文件中。stage-definition.yaml默认情况下，导出的定义文件包含 API Gateway 扩展名。

```
aws apigatewayv2 export-api \  
  --api-id a1b2c3d4 \  
  --output-type YAML \  
  --specification OAS30 \  
  --stage-name prod \  
  stage-definition.yaml
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的[从 API Gateway 导出 HTTP API](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ExportApi](#)中的。

## get-api-mapping

以下代码示例演示了如何使用 get-api-mapping。

### AWS CLI

获取有关自定义域名的 API 映射的信息

以下get-api-mapping示例显示了有关api.example.com自定义域名的 API 映射的信息。

```
aws apigatewayv2 get-api-mapping \  
  --api-mapping-id a1b2c3 \  
  --domain-name api.example.com
```

输出：



```
{
  "ApiId": "a1b2c3d4",
  "ApiMappingId": "a1b2c3d5",
  "ApiMappingKey": "myTestApi"
  "Stage": "test"
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的在 API Gateway [y 中设置区域自定义域名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetApiMapping](#) 中的。

## get-api-mappings

以下代码示例演示了如何使用 get-api-mappings。

### AWS CLI

获取自定义域名的 API 映射

以下 get-api-mappings 示例显示了 api.example.com 自定义域名的所有 API 映射的列表。

```
aws apigatewayv2 get-api-mappings \
  --domain-name api.example.com
```

输出：

```
{
  "Items": [
    {
      "ApiId": "a1b2c3d4",
      "ApiMappingId": "a1b2c3d5",
      "ApiMappingKey": "myTestApi"
      "Stage": "test"
    },
    {
      "ApiId": "a5b6c7d8",
      "ApiMappingId": "a1b2c3d6",
      "ApiMappingKey": "myDevApi"
      "Stage": "dev"
    },
  ],
}
```

```
]
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的在 API Gateway [y 中设置区域自定义域名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetApiMappings](#) 中的。

## get-api

以下代码示例演示了如何使用 get-api。

### AWS CLI

检索有关 API 的信息

以下 get-api 示例显示了有关 API 的信息。

```
aws apigatewayv2 get-api \
  --api-id a1b2c3d4
```

输出：

```
{
  "ApiEndpoint": "https://a1b2c3d4.execute-api.us-west-2.amazonaws.com",
  "ApiId": "a1b2c3d4",
  "ApiKeySelectionExpression": "$request.header.x-api-key",
  "CreateDate": "2020-03-28T00:32:37Z",
  "Name": "my-api",
  "ProtocolType": "HTTP",
  "RouteSelectionExpression": "$request.method $request.path",
  "Tags": {
    "department": "finance"
  }
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetApi](#) 中的。

## get-apis

以下代码示例演示了如何使用 get-apis。

## AWS CLI

### 检索 API 列表

以下get-apis示例列出了当前用户的所有 API。

```
aws apigatewayv2 get-apis
```

输出：

```
{
  "Items": [
    {
      "ApiEndpoint": "wss://a1b2c3d4.execute-api.us-west-2.amazonaws.com",
      "ApiId": "a1b2c3d4",
      "ApiKeySelectionExpression": "$request.header.x-api-key",
      "CreateDate": "2020-04-07T20:21:59Z",
      "Name": "my-websocket-api",
      "ProtocolType": "WEBSOCKET",
      "RouteSelectionExpression": "$request.body.message",
      "Tags": {}
    },
    {
      "ApiEndpoint": "https://a1b2c3d5.execute-api.us-west-2.amazonaws.com",
      "ApiId": "a1b2c3d5",
      "ApiKeySelectionExpression": "$request.header.x-api-key",
      "CreateDate": "2020-04-07T20:23:50Z",
      "Name": "my-http-api",
      "ProtocolType": "HTTP",
      "RouteSelectionExpression": "$request.method $request.path",
      "Tags": {}
    }
  ]
}
```

有关更多信息，请参阅 Amazon [API Gateway 开发者指南中的使用 WebSocket HTTP API 和使用 API](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetApis](#)中的。

### get-authorizer

以下代码示例演示了如何使用 get-authorizer。

## AWS CLI

### 检索有关授权者的信息

以下`get-authorizer`示例显示了有关授权者的信息。

```
aws apigatewayv2 get-authorizer \  
  --api-id a1b2c3d4 \  
  --authorizer-id a1b2c3
```

输出：

```
{  
  "AuthorizerId": "a1b2c3",  
  "AuthorizerType": "JWT",  
  "IdentitySource": [  
    "$request.header.Authorization"  
  ],  
  "JwtConfiguration": {  
    "Audience": [  
      "123456abc"  
    ],  
    "Issuer": "https://cognito-idp.us-west-2.amazonaws.com/us-west-2_abc123"  
  },  
  "Name": "my-jwt-authorizer"  
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 JWT 授权方控制 HTTP API 的[访问权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAuthorizer](#)中的。

## get-authorizers

以下代码示例演示了如何使用 `get-authorizers`。

## AWS CLI

### 检索 API 的授权者列表

以下`get-authorizers`示例显示了 API 的所有授权者的列表。

```
aws apigatewayv2 get-authorizers \  
  --api-id a1b2c3d4
```

输出：

```
{  
  "Items": [  
    {  
      "AuthorizerId": "a1b2c3",  
      "AuthorizerType": "JWT",  
      "IdentitySource": [  
        "$request.header.Authorization"  
      ],  
      "JwtConfiguration": {  
        "Audience": [  
          "123456abc"  
        ],  
        "Issuer": "https://cognito-idp.us-west-2.amazonaws.com/us-  
west-2_abc123"  
      },  
      "Name": "my-jwt-authorizer"  
    },  
    {  
      "AuthorizerId": "a1b2c4",  
      "AuthorizerType": "JWT",  
      "IdentitySource": [  
        "$request.header.Authorization"  
      ],  
      "JwtConfiguration": {  
        "Audience": [  
          "6789abcde"  
        ],  
        "Issuer": "https://cognito-idp.us-west-2.amazonaws.com/us-  
west-2_abc234"  
      },  
      "Name": "new-jwt-authorizer"  
    }  
  ]  
}
```

有关更多信息，请参阅 [Amazon API Gateway 开发者指南](#) 中的使用 JWT 授权方控制 HTTP API 的 [访问权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAuthorizers](#)中的。

## get-deployment

以下代码示例演示了如何使用 get-deployment。

### AWS CLI

检索有关部署的信息

以下get-deployment示例显示了有关部署的信息。

```
aws apigatewayv2 get-deployment \  
  --api-id a1b2c3d4 \  
  --deployment-id abcdef
```

输出：

```
{  
  "AutoDeployed": true,  
  "CreateDate": "2020-04-07T23:58:40Z",  
  "DeploymentId": "abcdef",  
  "DeploymentStatus": "DEPLOYED",  
  "Description": "Automatic deployment triggered by changes to the Api  
  configuration"  
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的 API [部署](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDeployment](#)中的。

## get-deployments

以下代码示例演示了如何使用 get-deployments。

### AWS CLI

检索部署列表

以下get-deployments示例显示了 API 的所有部署的列表。

```
aws apigatewayv2 get-deployments \  
  --api-id a1b2c3d4
```

```
--api-id a1b2c3d4
```

输出：

```
{
  "Items": [
    {
      "AutoDeployed": true,
      "CreatedDate": "2020-04-07T23:58:40Z",
      "DeploymentId": "abcdef",
      "DeploymentStatus": "DEPLOYED",
      "Description": "Automatic deployment triggered by changes to the Api
configuration"
    },
    {
      "AutoDeployed": true,
      "CreatedDate": "2020-04-06T00:33:00Z",
      "DeploymentId": "bcdefg",
      "DeploymentStatus": "DEPLOYED",
      "Description": "Automatic deployment triggered by changes to the Api
configuration"
    }
  ]
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的 [API 部署](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDeployments](#) 中的。

## get-domain-name

以下代码示例演示了如何使用 get-domain-name。

### AWS CLI

检索有关自定义域名的信息

以下 get-domain-name 示例显示有关自定义域名的信息。

```
aws apigatewayv2 get-domain-name \
  --domain-name api.example.com
```

输出：

```
{
  "ApiMappingSelectionExpression": "$request.basepath",
  "DomainName": "api.example.com",
  "DomainNameConfigurations": [
    {
      "ApiGatewayDomainName": "d-1234.execute-api.us-west-2.amazonaws.com",
      "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678",
      "EndpointType": "REGIONAL",
      "HostedZoneId": "123456789111",
      "SecurityPolicy": "TLS_1_2",
      "DomainNameStatus": "AVAILABLE"
    }
  ],
  "Tags": {}
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的在 API Gateway [y 中设置区域自定义域名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDomainName](#) 中的。

## get-domain-names

以下代码示例演示了如何使用 get-domain-names。

### AWS CLI

检索自定义域名列表

以下 get-domain-names 示例显示了当前用户的所有自定义域名的列表。

```
aws apigatewayv2 get-domain-names
```

输出：

```
{
  "Items": [
    {
      "ApiMappingSelectionExpression": "$request.basepath",
      "DomainName": "api.example.com",
      "DomainNameConfigurations": [
```



```

        {
            "ApiGatewayDomainName": "d-1234.execute-api.us-
west-2.amazonaws.com",
            "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678",
            "EndpointType": "REGIONAL",
            "HostedZoneId": "123456789111",
            "SecurityPolicy": "TLS_1_2",
            "DomainNameStatus": "AVAILABLE"
        }
    ],
    {
        "ApiMappingSelectionExpression": "$request.basepath",
        "DomainName": "newApi.example.com",
        "DomainNameConfigurations": [
            {
                "ApiGatewayDomainName": "d-5678.execute-api.us-
west-2.amazonaws.com",
                "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678",
                "EndpointType": "REGIONAL",
                "HostedZoneId": "123456789222",
                "SecurityPolicy": "TLS_1_2",
                "DomainNameStatus": "AVAILABLE"
            }
        ]
    }
]
}

```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的在 API Gateway [y 中设置区域自定义域名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDomainNames](#) 中的。

## get-integration

以下代码示例演示了如何使用 get-integration。

### AWS CLI

检索有关集成的信息

以下`get-integration`示例显示了有关集成的信息。

```
aws apigatewayv2 get-integration \  
  --api-id a1b2c3d4 \  
  --integration-id a1b2c3
```

输出：

```
{  
  "ApiGatewayManaged": true,  
  "ConnectionType": "INTERNET",  
  "IntegrationId": "a1b2c3",  
  "IntegrationMethod": "POST",  
  "IntegrationType": "AWS_PROXY",  
  "IntegrationUri": "arn:aws:lambda:us-west-2:12356789012:function:hello12",  
  "PayloadFormatVersion": "2.0",  
  "TimeoutInMillis": 30000  
}
```

有关更多信息，请参阅 Amazon [API Gateway 开发者指南中的配置 HTTP WebSocket API 的集成和设置 API 集成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetIntegration](#) 中的。

## get-integrations

以下代码示例演示了如何使用 `get-integrations`。

AWS CLI

检索集成列表

以下`get-integrations`示例显示了 API 的所有集成的列表。

```
aws apigatewayv2 get-integrations \  
  --api-id a1b2c3d4
```

输出：

```
{  
  "Items": [  
    {  
      "IntegrationId": "a1b2c3",  
      "IntegrationMethod": "POST",  
      "IntegrationType": "AWS_PROXY",  
      "IntegrationUri": "arn:aws:lambda:us-west-2:12356789012:function:hello12",  
      "PayloadFormatVersion": "2.0",  
      "TimeoutInMillis": 30000  
    }  
  ]  
}
```

```
{
  "ApiGatewayManaged": true,
  "ConnectionType": "INTERNET",
  "IntegrationId": "a1b2c3",
  "IntegrationMethod": "POST",
  "IntegrationType": "AWS_PROXY",
  "IntegrationUri": "arn:aws:lambda:us-west-2:123456789012:function:my-
function",
  "PayloadFormatVersion": "2.0",
  "TimeoutInMillis": 30000
},
{
  "ConnectionType": "INTERNET",
  "IntegrationId": "a1b2c4",
  "IntegrationMethod": "ANY",
  "IntegrationType": "HTTP_PROXY",
  "IntegrationUri": "https://www.example.com",
  "PayloadFormatVersion": "1.0",
  "TimeoutInMillis": 30000
}
]
```

有关更多信息，请参阅 Amazon [API Gateway 开发者指南中的配置 HTTP WebSocket API 的集成和设置 API 集成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetIntegrations](#) 中的。

## get-route

以下代码示例演示了如何使用 get-route。

### AWS CLI

检索有关路径的信息

以下 get-route 示例显示有关路径的信息。

```
aws apigatewayv2 get-route \
  --api-id a1b2c3d4 \
  --route-id 72jz1wk
```

输出：

```
{
  "ApiKeyRequired": false,
  "AuthorizationType": "NONE",
  "RouteId": "72jz1wk",
  "RouteKey": "ANY /pets",
  "Target": "integrations/a1b2c3"
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API [的路由](#)

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetRoute](#) 中的。

## get-routes

以下代码示例演示了如何使用 get-routes。

### AWS CLI

#### 检索路线列表

以下 get-routes 示例显示了 API 的所有路由的列表。

```
aws apigatewayv2 get-routes \
  --api-id a1b2c3d4
```

输出：

```
{
  "Items": [
    {
      "ApiKeyRequired": false,
      "AuthorizationType": "NONE",
      "RouteId": "72jz1wk",
      "RouteKey": "ANY /admin",
      "Target": "integrations/a1b2c3"
    },
    {
      "ApiGatewayManaged": true,
      "ApiKeyRequired": false,
      "AuthorizationType": "NONE",
      "RouteId": "go65gqi",
      "RouteKey": "$default",

```

```
        "Target": "integrations/a1b2c4"
      }
    ]
  }
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API [的路由](#)

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRoutes](#)中的。

## get-stage

以下代码示例演示了如何使用 get-stage。

### AWS CLI

检索有关阶段的信息

以下get-stage示例显示了有关 API prod 阶段的信息。

```
aws apigatewayv2 get-stage \
  --api-id a1b2c3d4 \
  --stage-name prod
```

输出：

```
{
  "CreateDate": "2020-04-08T00:36:05Z",
  "DefaultRouteSettings": {
    "DetailedMetricsEnabled": false
  },
  "DeploymentId": "x1zwyv",
  "LastUpdatedDate": "2020-04-08T00:36:13Z",
  "RouteSettings": {},
  "StageName": "prod",
  "StageVariables": {
    "function": "my-prod-function"
  },
  "Tags": {}
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API [的阶段](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetStage](#)中的。

## get-stages

以下代码示例演示了如何使用 `get-stages`。

### AWS CLI

#### 检索阶段列表

以下 `get-stages` 示例列出了 API 的所有阶段。

```
aws apigatewayv2 get-stages \  
  --api-id a1b2c3d4
```

输出：

```
{  
  "Items": [  
    {  
      "ApiGatewayManaged": true,  
      "AutoDeploy": true,  
      "CreateDate": "2020-04-08T00:08:44Z",  
      "DefaultRouteSettings": {  
        "DetailedMetricsEnabled": false  
      },  
      "DeploymentId": "dty748",  
      "LastDeploymentStatusMessage": "Successfully deployed stage with  
deployment ID 'dty748'",  
      "LastUpdatedDate": "2020-04-08T00:09:49Z",  
      "RouteSettings": {},  
      "StageName": "$default",  
      "StageVariables": {},  
      "Tags": {}  
    },  
    {  
      "AutoDeploy": true,  
      "CreateDate": "2020-04-08T00:35:06Z",  
      "DefaultRouteSettings": {  
        "DetailedMetricsEnabled": false  
      },  
      "LastUpdatedDate": "2020-04-08T00:35:48Z",  
      "RouteSettings": {},  
      "StageName": "dev",  
      "StageVariables": {
```

```
        "function": "my-dev-function"
      },
      "Tags": {}
    },
    {
      "CreateDate": "2020-04-08T00:36:05Z",
      "DefaultRouteSettings": {
        "DetailedMetricsEnabled": false
      },
      "DeploymentId": "x1zwyv",
      "LastUpdatedDate": "2020-04-08T00:36:13Z",
      "RouteSettings": {},
      "StageName": "prod",
      "StageVariables": {
        "function": "my-prod-function"
      },
      "Tags": {}
    }
  ]
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API [的阶段](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetStages](#) 中的。

## get-tags

以下代码示例演示了如何使用 get-tags。

### AWS CLI

检索资源的标签列表

以下 get-tags 示例列出了 API 的所有标签。

```
aws apigatewayv2 get-tags \
  --resource-arn arn:aws:apigateway:us-west-2::/apis/a1b2c3d4
```

输出：

```
{
  "Tags": {
    "owner": "dev-team",
```

```
    "environment": "prod"
  }
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的为你的 API 网关[资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTags](#)中的。

## get-vpc-link

以下代码示例演示了如何使用 get-vpc-link。

### AWS CLI

检索有关 VPC 链接的信息

以下get-vpc-link示例显示有关 VPC 链接的信息。

```
aws apigatewayv2 get-vpc-link \
  --vpc-link-id abcd123
```

输出：

```
{
  "CreateDate": "2020-04-07T00:27:47Z",
  "Name": "MyVpcLink",
  "SecurityGroupIds": [
    "sg1234",
    "sg5678"
  ],
  "SubnetIds": [
    "subnet-aaaa",
    "subnet-bbbb"
  ],
  "Tags": {},
  "VpcLinkId": "abcd123",
  "VpcLinkStatus": "AVAILABLE",
  "VpcLinkStatusMessage": "VPC link is ready to route traffic",
  "VpcLinkVersion": "V2"
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API 的[VPC 链接](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVpcLink](#)中的。

## get-vpc-links

以下代码示例演示了如何使用 `get-vpc-links`。

### AWS CLI

#### 检索 VPC 链接列表

以下`get-vpc-links`示例显示了当前用户的所有 VPC 链接的列表。

```
aws apigatewayv2 get-vpc-links
```

输出：

```
{
  "Items": [
    {
      "CreateDate": "2020-04-07T00:27:47Z",
      "Name": "MyVpcLink",
      "SecurityGroupIds": [
        "sg1234",
        "sg5678"
      ],
      "SubnetIds": [
        "subnet-aaaa",
        "subnet-bbbb"
      ],
      "Tags": {},
      "VpcLinkId": "abcd123",
      "VpcLinkStatus": "AVAILABLE",
      "VpcLinkStatusMessage": "VPC link is ready to route traffic",
      "VpcLinkVersion": "V2"
    }
  ],
  {
    "CreateDate": "2020-04-07T00:27:47Z",
    "Name": "MyOtherVpcLink",
    "SecurityGroupIds": [
      "sg1234",
      "sg5678"
    ],
  ],
}
```

```
        "SubnetIds": [
            "subnet-aaaa",
            "subnet-bbbb"
        ],
        "Tags": {},
        "VpcLinkId": "abcd456",
        "VpcLinkStatus": "AVAILABLE",
        "VpcLinkStatusMessage": "VPC link is ready to route traffic",
        "VpcLinkVersion": "V2"
    }
]
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API 的 [VPC 链接](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetVpcLinks](#) 中的。

## import-api

以下代码示例演示了如何使用 import-api。

### AWS CLI

#### 导入 HTTP API

以下 import-api 示例使用名为的 OpenAPI 3.0 定义文件创建一个 HTTP API。api-definition.yaml

```
aws apigatewayv2 import-api \
    --body file://api-definition.yaml
```

api-definition.yaml 的内容：

```
openapi: 3.0.1
info:
  title: My Lambda API
  version: v1.0
paths:
  /hello:
    x-amazon-apigateway-any-method:
      x-amazon-apigateway-integration:
        payloadFormatVersion: 2.0
```

```
    type: aws_proxy
    httpMethod: POST
    uri: arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:hello/invocations
    connectionType: INTERNET
```

输出：

```
{
  "ApiEndpoint": "https://a1b2c3d4.execute-api.us-west-2.amazonaws.com",
  "ApiId": "a1b2c3d4",
  "ApiKeySelectionExpression": "$request.header.x-api-key",
  "CreateDate": "2020-04-08T17:19:38+00:00",
  "Name": "My Lambda API",
  "ProtocolType": "HTTP",
  "RouteSelectionExpression": "$request.method $request.path",
  "Tags": {},
  "Version": "v1.0"
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API 的 OpenAPI [定义](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ImportApi](#) 中的。

## reimport-api

以下代码示例演示了如何使用 reimport-api。

### AWS CLI

#### 重新导入 HTTP API

以下 reimport-api 示例将现有 HTTP API 更新为使用中指定的 OpenAPI 3.0 定义。api-definition.yaml

```
aws apigatewayv2 reimport-api \
  --body file://api-definition.yaml \
  --api-id a1b2c3d4
```

api-definition.yaml 的内容：

```
openapi: 3.0.1
```

```
info:
  title: My Lambda API
  version: v1.0
paths:
  /hello:
    x-amazon-apigateway-any-method:
      x-amazon-apigateway-integration:
        payloadFormatVersion: 2.0
        type: aws_proxy
        httpMethod: POST
        uri: arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:12356789012:function:hello/invocations
        connectionType: INTERNET
```

输出：

```
{
  "ApiEndpoint": "https://a1b2c3d4.execute-api.us-west-2.amazonaws.com",
  "ApiId": "a1b2c3d4",
  "ApiKeySelectionExpression": "$request.header.x-api-key",
  "CreateDate": "2020-04-08T17:19:38+00:00",
  "Name": "My Lambda API",
  "ProtocolType": "HTTP",
  "RouteSelectionExpression": "$request.method $request.path",
  "Tags": {},
  "Version": "v1.0"
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 HTTP API 的 OpenAPI [定义](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReimportApi](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源添加标签

以下tag-resource示例向指定的 API 添加了一个密钥名称Department和值Accounting为的标签。

```
aws apigatewayv2 tag-resource \  
  --resource-arn arn:aws:apigateway:us-west-2::/apis/a1b2c3d4 \  
  --tags Department=Accounting
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的为你的 API 网关[资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

要从资源中删除标签

以下untag-resource示例Owner从指定 API 中删除带有密钥名称Project的标签。

```
aws apigatewayv2 untag-resource \  
  --resource-arn arn:aws:apigateway:us-west-2::/apis/a1b2c3d4 \  
  --tag-keys Project Owner
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的为你的 API 网关[资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-api-mapping

以下代码示例演示了如何使用 update-api-mapping。

### AWS CLI

更新 API 映射

以下update-api-mapping示例更改了自定义域名的 API 映射。因此，使用指定 API 和阶段的自定义域名的基本 URL 变为https://api.example.com/dev。

```
aws apigatewayv2 update-api-mapping \  
  --resource-arn arn:aws:apigateway:us-west-2::/apis/a1b2c3d4 \  
  --stage-name dev \  
  --mapping-key key-name \  
  --mapping-value value
```

```
--api-id a1b2c3d4 \  
--stage dev \  
--domain-name api.example.com \  
--api-mapping-id 0qzs2sy7bh \  
--api-mapping-key dev
```

输出：

```
{  
  "ApiId": "a1b2c3d4",  
  "ApiMappingId": "0qzs2sy7bh",  
  "ApiMappingKey": "dev"  
  "Stage": "dev"  
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的在 API Gateway [y 中设置区域自定义域名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateApiMapping](#) 中的。

## update-api

以下代码示例演示了如何使用 update-api。

### AWS CLI

为 HTTP API 启用 CORS

以下 update-api 示例更新了指定 API 的 CORS 配置以允许来自 <https://www.example.com> 的请求。

```
aws apigatewayv2 update-api \  
  --api-id a1b2c3d4 \  
  --cors-configuration AllowOrigins=https://www.example.com
```

输出：

```
{  
  "ApiEndpoint": "https://a1b2c3d4.execute-api.us-west-2.amazonaws.com",  
  "ApiId": "a1b2c3d4",  
  "ApiKeySelectionExpression": "$request.header.x-api-key",  
  "CorsConfiguration": {
```

```
    "AllowCredentials": false,
    "AllowHeaders": [
      "header1",
      "header2"
    ],
    "AllowMethods": [
      "GET",
      "OPTIONS"
    ],
    "AllowOrigins": [
      "https://www.example.com"
    ]
  },
  "CreateDate": "2020-04-08T18:39:37+00:00",
  "Name": "my-http-api",
  "ProtocolType": "HTTP",
  "RouteSelectionExpression": "$request.method $request.path",
  "Tags": {},
  "Version": "v1.0"
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的为 HTTP API [配置 CORS](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateApi](#) 中的。

## update-authorizer

以下代码示例演示了如何使用 update-authorizer。

### AWS CLI

#### 更新授权者

以下 update-authorizer 示例将 JWT 授权方的身份源更改为名为的标头。Authorization

```
aws apigatewayv2 update-authorizer \
  --api-id a1b2c3d4 \
  --authorizer-id a1b2c3 \
  --identity-source '$request.header.Authorization'
```

输出：

```
{
```

```
"AuthorizerId": "a1b2c3",
"AuthorizerType": "JWT",
"IdentitySource": [
  "$request.header.Authorization"
],
"JwtConfiguration": {
  "Audience": [
    "123456abc"
  ],
  "Issuer": "https://cognito-idp.us-west-2.amazonaws.com/us-west-2_abc123"
},
"Name": "my-jwt-authorizer"
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 JWT 授权方控制 HTTP API 的 [访问权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateAuthorizer](#) 中的。

## update-deployment

以下代码示例演示了如何使用 update-deployment。

### AWS CLI

#### 更改部署的描述

以下 update-deployment 示例更新了部署的描述。

```
aws apigatewayv2 update-deployment \
  --api-id a1b2c3d4 \
  --deployment-id abcdef \
  --description 'Manual deployment to fix integration test failures.'
```

输出：

```
{
  "AutoDeployed": false,
  "CreateDate": "2020-02-05T16:21:48+00:00",
  "DeploymentId": "abcdef",
  "DeploymentStatus": "DEPLOYED",
  "Description": "Manual deployment to fix integration test failures."
}
```



```
}
```

有关更多信息，请参阅 Amazon [API Gateway 开发者指南](#)中的在 [API Gateway 中开发 HTTP API](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDeployment](#)中的。

## update-domain-name

以下代码示例演示了如何使用 update-domain-name。

### AWS CLI

#### 更新自定义域名

以下update-domain-name示例为api.example.com自定义域名指定了新的 ACM 证书。

```
aws apigatewayv2 update-domain-name \  
  --domain-name api.example.com \  
  --domain-name-configurations CertificateArn=arn:aws:acm:us-  
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678
```

输出：

```
{  
  "ApiMappingSelectionExpression": "$request.basepath",  
  "DomainName": "regional.example.com",  
  "DomainNameConfigurations": [  
    {  
      "ApiGatewayDomainName": "d-id.execute-api.us-west-2.amazonaws.com",  
      "CertificateArn": "arn:aws:acm:us-  
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678",  
      "EndpointType": "REGIONAL",  
      "HostedZoneId": "123456789111",  
      "SecurityPolicy": "TLS_1_2",  
      "DomainNameStatus": "AVAILABLE"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的在 [API Gateway 中设置区域自定义域名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDomainName](#)中的。

## update-integration

以下代码示例演示了如何使用 update-integration。

### AWS CLI

#### 更新 Lambda 集成

以下update-integration示例将现有的 AWS Lambda 集成更新为使用指定的 Lambda 函数。

```
aws apigatewayv2 update-integration \  
  --api-id a1b2c3d4 \  
  --integration-id a1b2c3 \  
  --integration-uri arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/  
arn:aws:lambda:us-west-2:123456789012:function:my-new-function/invocations
```

输出：

```
{  
  "ConnectionType": "INTERNET",  
  "IntegrationId": "a1b2c3",  
  "IntegrationMethod": "POST",  
  "IntegrationType": "AWS_PROXY",  
  "IntegrationUri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/  
functions/arn:aws:lambda:us-west-2:123456789012:function:my-new-function/  
invocations",  
  "PayloadFormatVersion": "2.0",  
  "TimeoutInMillis": 5000  
}
```

有关更多信息，请参阅 Amazon [API Gateway 开发者指南中的配置 HTTP WebSocket API 的集成和设置 API 集成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateIntegration](#) 中的。

## update-route

以下代码示例演示了如何使用 update-route。

### AWS CLI

#### 示例 1：更新路径的集成

以下update-route示例更新了指定路径的集成。

```
aws apigatewayv2 update-route \  
  --api-id a1b2c3d4 \  
  --route-id a1b2c3 \  
  --target integrations/a1b2c6
```

输出：

```
{  
  "ApiKeyRequired": false,  
  "AuthorizationType": "NONE",  
  "RouteId": "a1b2c3",  
  "RouteKey": "ANY /pets",  
  "Target": "integrations/a1b2c6"  
}
```

示例 2：向路径添加授权者

以下update-route示例将指定路由更新为使用 JWT 授权方。

```
aws apigatewayv2 update-route \  
  --api-id a1b2c3d4 \  
  --route-id a1b2c3 \  
  --authorization-type JWT \  
  --authorizer-id a1b2c5 \  
  --authorization-scopes user.id user.email
```

输出：

```
{  
  "ApiKeyRequired": false,  
  "AuthorizationScopes": [  
    "user.id",  
    "user.email"  
  ],  
  "AuthorizationType": "JWT",  
  "AuthorizerId": "a1b2c5",  
  "OperationName": "GET HTTP",  
  "RequestParameters": {},  
  "RouteId": "a1b2c3",  
  "RouteKey": "GET /pets",
```

```
"Target": "integrations/a1b2c6"
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的使用 JWT 授权方控制 HTTP API 的[访问权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateRoute](#) 中的。

## update-stage

以下代码示例演示了如何使用 update-stage。

### AWS CLI

#### 配置自定义限制

以下 update-stage 示例为 API 的指定阶段和路径配置自定义流控。

```
aws apigatewayv2 update-stage \  
  --api-id a1b2c3d4 \  
  --stage-name dev \  
  --route-settings '{"GET /pets":  
{"ThrottlingBurstLimit":100,"ThrottlingRateLimit":2000}}'
```

输出：

```
{  
  "CreateDate": "2020-04-05T16:21:16+00:00",  
  "DefaultRouteSettings": {  
    "DetailedMetricsEnabled": false  
  },  
  "DeploymentId": "shktxb",  
  "LastUpdatedDate": "2020-04-08T22:23:17+00:00",  
  "RouteSettings": {  
    "GET /pets": {  
      "ThrottlingBurstLimit": 100,  
      "ThrottlingRateLimit": 2000.0  
    }  
  },  
  "StageName": "dev",  
  "StageVariables": {},  
  "Tags": {}  
}
```

```
}
```

有关更多信息，请参阅 Amazon [API Gateway 开发者指南](#) 中的 [保护您的 HTTP API](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateStage](#) 中的。

## update-vpc-link

以下代码示例演示了如何使用 `update-vpc-link`。

### AWS CLI

#### 更新 VPC 链接

以下 `update-vpc-link` 示例更新了 VPC 链接的名称。创建 VPC 链接后，您无法更改其安全组或子网。

```
aws apigatewayv2 update-vpc-link \  
  --vpc-link-id abcd123 \  
  --name MyUpdatedVpcLink
```

输出：

```
{  
  "CreateDate": "2020-04-07T00:27:47Z",  
  "Name": "MyUpdatedVpcLink",  
  "SecurityGroupIds": [  
    "sg1234",  
    "sg5678"  
  ],  
  "SubnetIds": [  
    "subnet-aaaa",  
    "subnet-bbbb"  
  ],  
  "Tags": {},  
  "VpcLinkId": "abcd123",  
  "VpcLinkStatus": "AVAILABLE",  
  "VpcLinkStatusMessage": "VPC link is ready to route traffic",  
  "VpcLinkVersion": "V2"  
}
```

有关更多信息，请参阅 Amazon API Gateway 开发者指南中的 [使用 HTTP API 的 VPC 链接](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateVpcLink](#)中的。

## API Gateway 管理 API 示例使用 AWS CLI

以下代码示例向您展示了如何使用 with API Gateway Management API 来执行操作和实现常见场景。  
AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **delete-connection**

以下代码示例演示了如何使用 delete-connection。

AWS CLI

删除 WebSocket 连接

以下delete-connection示例断开客户端与指定 WebSocket API 的连接。

```
aws apigatewaymanagementapi delete-connection \  
  --connection-id L0SM9c0FvHcCIhw= \  
  --endpoint-url https://aabbccddee.execute-api.us-west-2.amazonaws.com/prod
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon API Gateway 开发人员指南》中的[在后端服务中使用 @connections 命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteConnection](#)中的。

## get-connection

以下代码示例演示了如何使用 `get-connection`。

### AWS CLI

获取有关 WebSocket 连接的信息

以下 `get-connection` 示例描述了与指定 WebSocket API 的连接。

```
aws apigatewaymanagementapi get-connection \  
  --connection-id L0SM9c0FvHcCIhw= \  
  --endpoint-url https://aabbccdde.execute-api.us-west-2.amazonaws.com/prod
```

输出：

```
{  
  "ConnectedAt": "2020-04-30T20:10:33.236Z",  
  "Identity": {  
    "SourceIp": "192.0.2.1"  
  },  
  "LastActiveAt": "2020-04-30T20:10:42.997Z"  
}
```

有关更多信息，请参阅《Amazon API Gateway 开发人员指南》中的[在后端服务中使用 @connections 命令](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetConnection](#)中的。

## post-to-connection

以下代码示例演示了如何使用 `post-to-connection`。

### AWS CLI

向 WebSocket 连接发送数据

以下 `post-to-connection` 示例向连接到指定 WebSocket API 的客户端发送消息。

```
aws apigatewaymanagementapi post-to-connection \  
  --connection-id L0SM9c0FvHcCIhw= \  
  --data "Hello from API Gateway!" \  
  --endpoint-url https://aabbccdde.execute-api.us-west-2.amazonaws.com/prod
```

```
--endpoint-url https://aabbccddee.execute-api.us-west-2.amazonaws.com/prod
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon API Gateway 开发人员指南》中的[在后端服务中使用 @connections 命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PostToConnection](#)中的。

## 使用 App Mesh 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 App Mesh 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### create-mesh

以下代码示例演示了如何使用 create-mesh。

AWS CLI

示例 1：创建新的服务网格

以下create-mesh示例创建了一个服务网格。

```
aws appmesh create-mesh \  
  --mesh-name app1
```

输出：



```
{
  "mesh":{
    "meshName":"app1",
    "metadata":{
      "arn":"arn:aws:appmesh:us-east-1:123456789012:mesh/app1",
      "createdAt":1563809909.282,
      "lastUpdatedAt":1563809909.282,
      "uid":"a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version":1
    },
    "spec":{},
    "status":{
      "status":"ACTIVE"
    }
  }
}
```

## 示例 2：创建包含多个标签的新服务网格

以下create-mesh示例创建了一个包含多个标签的服务网格。

```
aws appmesh create-mesh \
  --mesh-name app2 \
  --tags key=key1,value=value1 key=key2,value=value2 key=key3,value=value3
```

输出：

```
{
  "mesh":{
    "meshName":"app2",
    "metadata":{
      "arn":"arn:aws:appmesh:us-east-1:123456789012:mesh/app2",
      "createdAt":1563822121.877,
      "lastUpdatedAt":1563822121.877,
      "uid":"a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version":1
    },
    "spec":{},
    "status":{
      "status":"ACTIVE"
    }
  }
}
```

```
}
```

有关更多信息，请参阅 AWS App [Mesh 用户指南中的服务网格](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateMesh](#) 中的。

## create-route

以下代码示例演示了如何使用 create-route。

### AWS CLI

#### 创建新的 gRPC 路线

以下 create-route 示例使用 JSON 输入文件创建 gRPC 路由。元数据以 123 开头的 GRPC 流量将路由到名为 ServiceBGrpc 的虚拟节点。如果在尝试与路由目标通信时出现特定的 gRPC、HTTP 或 TCP 故障，则会重试该路由三次。每次重试之间有 15 秒的延迟。

```
aws appmesh create-route \  
  --cli-input-json file://create-route-grpc.json
```

create-route-grpc.json 的内容：

```
{  
  "meshName" : "apps",  
  "routeName" : "grpcRoute",  
  "spec" : {  
    "grpcRoute" : {  
      "action" : {  
        "weightedTargets" : [  
          {  
            "virtualNode" : "serviceBgrpc",  
            "weight" : 100  
          }  
        ]  
      },  
      "match" : {  
        "metadata" : [  
          {  
            "invert" : false,  
            "match" : {  
              "prefix" : "123"  
            }  
          }  
        ]  
      }  
    }  
  }  
}
```

```

        },
        "name" : "myMetadata"
    }
],
"methodName" : "GetColor",
"serviceName" : "com.amazonaws.services.ColorService"
},
"retryPolicy" : {
    "grpcRetryEvents" : [ "deadline-exceeded" ],
    "httpRetryEvents" : [ "server-error", "gateway-error" ],
    "maxRetries" : 3,
    "perRetryTimeout" : {
        "unit" : "s",
        "value" : 15
    },
    "tcpRetryEvents" : [ "connection-error" ]
}
},
"priority" : 100
},
"virtualRouterName" : "serviceBgrpc"
}

```

输出：

```

{
  "route": {
    "meshName": "apps",
    "metadata": {
      "arn": "arn:aws:appmesh:us-west-2:123456789012:mesh/apps/virtualRouter/serviceBgrpc/route/grpcRoute",
      "createdAt": 1572010806.008,
      "lastUpdatedAt": 1572010806.008,
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version": 1
    },
    "routeName": "grpcRoute",
    "spec": {
      "grpcRoute": {
        "action": {
          "weightedTargets": [
            {
              "virtualNode": "serviceBgrpc",

```

```
        "weight": 100
      }
    ]
  },
  "match": {
    "metadata": [
      {
        "invert": false,
        "match": {
          "prefix": "123"
        },
        "name": "mymetadata"
      }
    ],
    "methodName": "GetColor",
    "serviceName": "com.amazonaws.services.ColorService"
  },
  "retryPolicy": {
    "grpcRetryEvents": [
      "deadline-exceeded"
    ],
    "httpRetryEvents": [
      "server-error",
      "gateway-error"
    ],
    "maxRetries": 3,
    "perRetryTimeout": {
      "unit": "s",
      "value": 15
    },
    "tcpRetryEvents": [
      "connection-error"
    ]
  }
},
"priority": 100
},
"status": {
  "status": "ACTIVE"
},
"virtualRouterName": "serviceBgrpc"
}
}
```

## 创建新的 HTTP 或 HTTP/2 路由

以下 `create-route` 示例使用 JSON 输入文件创建 HTTP/2 路由。要创建 HTTP 路由，请在规范下将 `http2Route` 替换为 `httpRoute`。所有发往标头值以 123 开头的任何 URL 前缀的 HTTP/2 流量都将路由到名为 `serviceBhttp2` 的虚拟节点。如果在尝试与路由目标通信时出现特定的 HTTP 或 TCP 故障，则会重试该路由三次。每次重试之间有 15 秒的延迟。

```
aws appmesh create-route \  
  --cli-input-json file://create-route-http2.json
```

`create-route-http2.json` 的内容：

```
{  
  "meshName": "apps",  
  "routeName": "http2Route",  
  "spec": {  
    "http2Route": {  
      "action": {  
        "weightedTargets": [  
          {  
            "virtualNode": "serviceBhttp2",  
            "weight": 100  
          }  
        ]  
      },  
      "match": {  
        "headers": [  
          {  
            "invert": false,  
            "match": {  
              "prefix": "123"  
            },  
            "name": "clientRequestId"  
          }  
        ],  
        "method": "POST",  
        "prefix": "/",  
        "scheme": "http"  
      },  
      "retryPolicy": {  
        "httpRetryEvents": [  
          "server-error",  
          "gateway-error"  
        ]  
      }  
    }  
  }  
}
```

```

    ],
    "maxRetries": 3,
    "perRetryTimeout": {
      "unit": "s",
      "value": 15
    },
    "tcpRetryEvents": [
      "connection-error"
    ]
  }
},
"priority": 200
},
"virtualRouterName": "serviceBhttp2"
}

```

输出：

```

{
  "route": {
    "meshName": "apps",
    "metadata": {
      "arn": "arn:aws:appmesh:us-west-2:123456789012:mesh/apps/virtualRouter/
serviceBhttp2/route/http2Route",
      "createdAt": 1572011008.352,
      "lastUpdatedAt": 1572011008.352,
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version": 1
    },
    "routeName": "http2Route",
    "spec": {
      "http2Route": {
        "action": {
          "weightedTargets": [
            {
              "virtualNode": "serviceBhttp2",
              "weight": 100
            }
          ]
        },
        "match": {
          "headers": [
            {

```

```

        "invert": false,
        "match": {
            "prefix": "123"
        },
        "name": "clientRequestId"
    }
],
"method": "POST",
"prefix": "/",
"scheme": "http"
},
"retryPolicy": {
    "httpRetryEvents": [
        "server-error",
        "gateway-error"
    ],
    "maxRetries": 3,
    "perRetryTimeout": {
        "unit": "s",
        "value": 15
    },
    "tcpRetryEvents": [
        "connection-error"
    ]
}
},
"priority": 200
},
"status": {
    "status": "ACTIVE"
},
"virtualRouterName": "serviceBhttp2"
}
}

```

## 创建新的 TCP 路由

以下 `create-route` 示例使用 JSON 输入文件创建 TCP 路由。75% 的流量路由到名为 `serviceBTCP` 的虚拟节点，25% 的流量路由到名为 `serviceBv2TCP` 的虚拟节点。为不同的目标指定不同的权重是部署应用程序新版本的有效方法。您可以调整权重，以便最终将所有流量的 100% 路由到具有新版本应用程序的目标。

```
aws appmesh create-route \
```

```
--cli-input-json file://create-route-tcp.json
```

create-route-tcp.json 的内容：

```
{
  "meshName": "apps",
  "routeName": "tcpRoute",
  "spec": {
    "priority": 300,
    "tcpRoute": {
      "action": {
        "weightedTargets": [
          {
            "virtualNode": "serviceBtcp",
            "weight": 75
          },
          {
            "virtualNode": "serviceBv2tcp",
            "weight": 25
          }
        ]
      }
    }
  },
  "virtualRouterName": "serviceBtcp"
}
```

输出：

```
{
  "route": {
    "meshName": "apps",
    "metadata": {
      "arn": "arn:aws:appmesh:us-west-2:123456789012:mesh/apps/virtualRouter/serviceBtcp/route/tcpRoute",
      "createdAt": 1572011436.26,
      "lastUpdatedAt": 1572011436.26,
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version": 1
    },
    "routeName": "tcpRoute",
    "spec": {
      "priority": 300,

```



```

    "tcpRoute": {
      "action": {
        "weightedTargets": [
          {
            "virtualNode": "serviceBtcp",
            "weight": 75
          },
          {
            "virtualNode": "serviceBv2tcp",
            "weight": 25
          }
        ]
      }
    },
    "status": {
      "status": "ACTIVE"
    },
    "virtualRouterName": "serviceBtcp"
  }
}

```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[路由](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateRoute](#)中的。

## create-virtual-gateway

以下代码示例演示了如何使用 create-virtual-gateway。

### AWS CLI

#### 创建新的虚拟网关

以下 create-virtual-gateway 示例使用 JSON 输入文件创建带有使用端口 9080 的 HTTP 侦听器的虚拟网关。

```

aws appmesh create-virtual-gateway \
  --mesh-name meshName \
  --virtual-gateway-name virtualGatewayName \
  --cli-input-json file://create-virtual-gateway.json

```

create-virtual-gateway.json 的内容：

```
{
  "spec": {
    "listeners": [
      {
        "portMapping": {
          "port": 9080,
          "protocol": "http"
        }
      }
    ]
  }
}
```

输出：

```
{
  "virtualGateway": {
    "meshName": "meshName",
    "metadata": {
      "arn": "arn:aws:appmesh:us-west-2:123456789012:mesh/meshName/virtualGateway/virtualGatewayName",
      "createdAt": "2022-04-06T10:42:42.015000-05:00",
      "lastUpdatedAt": "2022-04-06T10:42:42.015000-05:00",
      "meshOwner": "123456789012",
      "resourceOwner": "123456789012",
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version": 1
    },
    "spec": {
      "listeners": [
        {
          "portMapping": {
            "port": 9080,
            "protocol": "http"
          }
        }
      ]
    },
    "status": {
      "status": "ACTIVE"
    },
    "virtualGatewayName": "virtualGatewayName"
  }
}
```

```
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟网关](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateVirtualGateway](#) 中的。

## create-virtual-node

以下代码示例演示了如何使用 create-virtual-node。

### AWS CLI

示例 1：创建使用 DNS 进行发现的新虚拟节点

以下 create-virtual-node 示例使用 JSON 输入文件创建使用 DNS 进行服务发现的虚拟节点。

```
aws appmesh create-virtual-node \  
  --cli-input-json file://create-virtual-node-dns.json
```

create-virtual-node-dns.json 的内容：

```
{  
  "meshName": "app1",  
  "spec": {  
    "listeners": [  
      {  
        "portMapping": {  
          "port": 80,  
          "protocol": "http"  
        }  
      }  
    ],  
    "serviceDiscovery": {  
      "dns": {  
        "hostname": "serviceBv1.svc.cluster.local"  
      }  
    }  
  },  
  "virtualNodeName": "vnServiceBv1"  
}
```

输出：

```
{
  "virtualNode": {
    "meshName": "app1",
    "metadata": {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualNode/vnServiceBv1",
      "createdAt": 1563810019.874,
      "lastUpdatedAt": 1563810019.874,
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version": 1
    },
    "spec": {
      "listeners": [
        {
          "portMapping": {
            "port": 80,
            "protocol": "http"
          }
        }
      ],
      "serviceDiscovery": {
        "dns": {
          "hostname": "serviceBv1.svc.cluster.local"
        }
      }
    },
    "status": {
      "status": "ACTIVE"
    },
    "virtualNodeName": "vnServiceBv1"
  }
}
```

## 示例 2：创建使用 AWS Cloud Map 进行发现的新虚拟节点

以下 `create-virtual-node` 示例使用 JSON 输入文件创建使用 AWS Cloud Map 进行服务发现的虚拟节点。

```
aws appmesh create-virtual-node \
  --cli-input-json file://create-virtual-node-cloud-map.json
```

`create-virtual-node-cloud-map.json` 的内容：

```
{
  "meshName": "app1",
  "spec": {
    "backends": [
      {
        "virtualService": {
          "virtualServiceName": "serviceA.svc.cluster.local"
        }
      }
    ],
    "listeners": [
      {
        "portMapping": {
          "port": 80,
          "protocol": "http"
        }
      }
    ],
    "serviceDiscovery": {
      "awsCloudMap": {
        "attributes": [
          {
            "key": "Environment",
            "value": "Testing"
          }
        ],
        "namespaceName": "namespace1",
        "serviceName": "serviceA"
      }
    }
  },
  "virtualNodeName": "vnServiceA"
}
```

输出：

```
{
  "virtualNode": {
    "meshName": "app1",
    "metadata": {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualNode/vnServiceA",
      "createdAt": 1563810859.465,
```

```
    "lastUpdatedAt": 1563810859.465,
    "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "version": 1
  },
  "spec": {
    "backends": [
      {
        "virtualService": {
          "virtualServiceName": "serviceA.svc.cluster.local"
        }
      }
    ],
    "listeners": [
      {
        "portMapping": {
          "port": 80,
          "protocol": "http"
        }
      }
    ],
    "serviceDiscovery": {
      "awsCloudMap": {
        "attributes": [
          {
            "key": "Environment",
            "value": "Testing"
          }
        ],
        "namespaceName": "namespace1",
        "serviceName": "serviceA"
      }
    }
  },
  "status": {
    "status": "ACTIVE"
  },
  "virtualNodeName": "vnServiceA"
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateVirtualNode](#)中的。

## create-virtual-router

以下代码示例演示了如何使用 `create-virtual-router`。

### AWS CLI

#### 创建新的虚拟路由器

以下 `create-virtual-router` 示例使用 JSON 输入文件创建带有使用端口 80 的 HTTP 侦听器的虚拟路由器。

```
aws appmesh create-virtual-router \  
  --cli-input-json file://create-virtual-router.json
```

`create-virtual-router.json` 的内容：

```
{  
  "meshName": "app1",  
  "spec": {  
    "listeners": [  
      {  
        "portMapping": {  
          "port": 80,  
          "protocol": "http"  
        }  
      }  
    ]  
  },  
  "virtualRouterName": "vrServiceB"  
}
```

输出：

```
{  
  "virtualRouter": {  
    "meshName": "app1",  
    "metadata": {  
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualRouter/  
vrServiceB",  
      "createdAt": 1563810546.59,  
      "lastUpdatedAt": 1563810546.59,  
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
      "version": 1  
    }  
  }  
}
```

```

    },
    "spec": {
      "listeners": [
        {
          "portMapping": {
            "port": 80,
            "protocol": "http"
          }
        }
      ]
    },
    "status": {
      "status": "ACTIVE"
    },
    "virtualRouterName": "vrServiceB"
  }
}

```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟路由器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateVirtualRouter](#)中的。

## create-virtual-service

以下代码示例演示了如何使用 create-virtual-service。

### AWS CLI

示例 1：使用虚拟节点提供商创建新的虚拟服务

以下 create-virtual-service 示例使用 JSON 输入文件通过虚拟节点提供者创建虚拟服务。

```

aws appmesh create-virtual-service \
  --cli-input-json file://create-virtual-service-virtual-node.json

```

create-virtual-service-virtual-node.json 的内容：

```

{
  "meshName": "app1",
  "spec": {
    "provider": {
      "virtualNode": {
        "virtualNodeName": "vnServiceA"
      }
    }
  }
}

```



```

    }
  },
  "virtualServiceName": "serviceA.svc.cluster.local"
}

```

输出：

```

{
  "virtualService": {
    "meshName": "app1",
    "metadata": {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualService/serviceA.svc.cluster.local",
      "createdAt": 1563810859.474,
      "lastUpdatedAt": 1563810967.179,
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version": 2
    },
    "spec": {
      "provider": {
        "virtualNode": {
          "virtualNodeName": "vnServiceA"
        }
      }
    },
    "status": {
      "status": "ACTIVE"
    },
    "virtualServiceName": "serviceA.svc.cluster.local"
  }
}

```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟节点](#)。

示例 2：使用虚拟路由器提供商创建新的虚拟服务

以下 create-virtual-service 示例使用 JSON 输入文件通过虚拟路由器提供商创建虚拟服务。

```

aws appmesh create-virtual-service \
  --cli-input-json file://create-virtual-service-virtual-router.json

```

create-virtual-service-virtual-router.json 的内容：

```
{
  "meshName": "app1",
  "spec": {
    "provider": {
      "virtualRouter": {
        "virtualRouterName": "vrServiceB"
      }
    }
  },
  "virtualServiceName": "serviceB.svc.cluster.local"
}
```

输出：

```
{
  "virtualService": {
    "meshName": "app1",
    "metadata": {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualService/
serviceB.svc.cluster.local",
      "createdAt": 1563908363.999,
      "lastUpdatedAt": 1563908363.999,
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version": 1
    },
    "spec": {
      "provider": {
        "virtualRouter": {
          "virtualRouterName": "vrServiceB"
        }
      }
    },
    "status": {
      "status": "ACTIVE"
    },
    "virtualServiceName": "serviceB.svc.cluster.local"
  }
}
```

有关更多信息，请参阅 < [https://docs.aws.amazon.com/app-mesh/latest/userguide/virtual\\_services.html](https://docs.aws.amazon.com/app-mesh/latest/userguide/virtual_services.html) > AWS App Mesh 用户指南中的虚拟服务

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateVirtualService](#) 中的。

## delete-mesh

以下代码示例演示了如何使用 delete-mesh。

### AWS CLI

#### 删除服务网格

以下delete-mesh示例删除了指定的服务网格。

```
aws appmesh delete-mesh \  
  --mesh-name app1
```

输出：

```
{  
  "mesh": {  
    "meshName": "app1",  
    "metadata": {  
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1",  
      "createdAt": 1563809909.282,  
      "lastUpdatedAt": 1563824981.248,  
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
      "version": 2  
    },  
    "spec": {  
      "egressFilter": {  
        "type": "ALLOW_ALL"  
      }  
    },  
    "status": {  
      "status": "DELETED"  
    }  
  }  
}
```

有关更多信息，请参阅 AWS App [Mesh 用户指南中的服务网格](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMesh](#)中的。

## delete-route

以下代码示例演示了如何使用 delete-route。

## AWS CLI

### 删除路线

以下delete-route示例删除了指定的路由。

```
aws appmesh delete-route \  
  --mesh-name app1 \  
  --virtual-router-name vrServiceB \  
  --route-name toVnServiceB-weighted
```

输出：

```
{  
  "route": {  
    "meshName": "app1",  
    "metadata": {  
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualRouter/  
vrServiceB/route/toVnServiceB-weighted",  
      "createdAt": 1563811384.015,  
      "lastUpdatedAt": 1563823915.936,  
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
      "version": 3  
    },  
    "routeName": "toVnServiceB-weighted",  
    "spec": {  
      "httpRoute": {  
        "action": {  
          "weightedTargets": [  
            {  
              "virtualNode": "vnServiceBv1",  
              "weight": 80  
            },  
            {  
              "virtualNode": "vnServiceBv2",  
              "weight": 20  
            }  
          ]  
        },  
        "match": {  
          "prefix": "/"  
        }  
      }  
    }  
  }  
}
```

```

    },
    "status": {
      "status": "DELETED"
    },
    "virtualRouterName": "vrServiceB"
  }
}

```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[路由](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteRoute](#)中的。

## delete-virtual-node

以下代码示例演示了如何使用 delete-virtual-node。

### AWS CLI

#### 删除虚拟节点

以下 delete-virtual-node 示例删除了指定的虚拟节点。

```

aws appmesh delete-virtual-node \
  --mesh-name app1 \
  --virtual-node-name vnServiceBv2

```

#### 输出：

```

{
  "virtualNode": {
    "meshName": "app1",
    "metadata": {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualNode/vnServiceBv2",
      "createdAt": 1563810117.297,
      "lastUpdatedAt": 1563824700.678,
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version": 2
    },
    "spec": {
      "backends": [],
      "listeners": [
        {

```

```

        "portMapping": {
            "port": 80,
            "protocol": "http"
        }
    ],
    "serviceDiscovery": {
        "dns": {
            "hostname": "serviceBv2.svc.cluster.local"
        }
    }
},
"status": {
    "status": "DELETED"
},
"virtualNodeName": "vnServiceBv2"
}
}

```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteVirtualNode](#)中的。

## delete-virtual-router

以下代码示例演示了如何使用 delete-virtual-router。

### AWS CLI

#### 删除虚拟路由器

以下 delete-virtual-router 示例删除了指定的虚拟路由器。

```

aws appmesh delete-virtual-router \
  --mesh-name app1 \
  --virtual-router-name vrServiceB

```

输出：

```

{
  "virtualRouter": {
    "meshName": "app1",
    "metadata": {

```

```
    "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualRouter/vrServiceB",
    "createdAt": 1563810546.59,
    "lastUpdatedAt": 1563824253.467,
    "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "version": 3
  },
  "spec": {
    "listeners": [
      {
        "portMapping": {
          "port": 80,
          "protocol": "http"
        }
      }
    ]
  },
  "status": {
    "status": "DELETED"
  },
  "virtualRouterName": "vrServiceB"
}
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟路由器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteVirtualRouter](#)中的。

## delete-virtual-service

以下代码示例演示了如何使用 delete-virtual-service。

### AWS CLI

#### 删除虚拟服务

以下 delete-virtual-service 示例删除了指定的虚拟服务。

```
aws appmesh delete-virtual-service \
  --mesh-name app1 \
  --virtual-service-name serviceB.svc.cluster.local
```

输出：

```
{
  "virtualService": {
    "meshName": "app1",
    "metadata": {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualService/
serviceB.svc.cluster.local",
      "createdAt": 1563908363.999,
      "lastUpdatedAt": 1563913940.866,
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version": 3
    },
    "spec": {},
    "status": {
      "status": "DELETED"
    },
    "virtualServiceName": "serviceB.svc.cluster.local"
  }
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟服务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteVirtualService](#)中的。

## describe-mesh

以下代码示例演示了如何使用 describe-mesh。

### AWS CLI

#### 描述服务网格

以下 describe-mesh 示例返回有关指定服务网格的详细信息。

```
aws appmesh describe-mesh \
  --mesh-name app1
```

输出：

```
{
  "mesh": {
    "meshName": "app1",
    "metadata": {
```



```

        "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1",
        "createdAt": 1563809909.282,
        "lastUpdatedAt": 1563809909.282,
        "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
        "version": 1
    },
    "spec": {},
    "status": {
        "status": "ACTIVE"
    }
}
}
}

```

有关更多信息，请参阅 AWS App [Mesh 用户指南中的服务](#) 网格。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeMesh](#) 中的。

## describe-route

以下代码示例演示了如何使用 describe-route。

### AWS CLI

#### 描述路线

以下 describe-route 示例返回有关指定路径的详细信息。

```

aws appmesh describe-route \
  --mesh-name app1 \
  --virtual-router-name vrServiceB \
  --route-name toVnServiceB-weighted

```

#### 输出：

```

{
  "route": {
    "meshName": "app1",
    "metadata": {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualRouter/
vrServiceB/route/toVnServiceB-weighted",
      "createdAt": 1563811384.015,
      "lastUpdatedAt": 1563811384.015,

```

```
    "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "version": 1
  },
  "routeName": "toVnServiceB-weighted",
  "spec": {
    "httpRoute": {
      "action": {
        "weightedTargets": [
          {
            "virtualNode": "vnServiceBv1",
            "weight": 90
          },
          {
            "virtualNode": "vnServiceBv2",
            "weight": 10
          }
        ]
      },
      "match": {
        "prefix": "/"
      }
    }
  },
  "status": {
    "status": "ACTIVE"
  },
  "virtualRouterName": "vrServiceB"
}
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[路由](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeRoute](#)中的。

## describe-virtual-node

以下代码示例演示了如何使用 describe-virtual-node。

### AWS CLI

#### 描述虚拟节点

以下 describe-virtual-node 示例返回有关指定虚拟节点的详细信息。

```
aws appmesh describe-virtual-node \  
  --mesh-name app1 \  
  --virtual-node-name vnServiceBv1
```

输出：

```
{  
  "virtualNode": {  
    "meshName": "app1",  
    "metadata": {  
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualNode/  
vnServiceBv1",  
      "createdAt": 1563810019.874,  
      "lastUpdatedAt": 1563810019.874,  
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
      "version": 1  
    },  
    "spec": {  
      "backends": [],  
      "listeners": [  
        {  
          "portMapping": {  
            "port": 80,  
            "protocol": "http"  
          }  
        }  
      ],  
      "serviceDiscovery": {  
        "dns": {  
          "hostname": "serviceBv1.svc.cluster.local"  
        }  
      }  
    },  
    "status": {  
      "status": "ACTIVE"  
    },  
    "virtualNodeName": "vnServiceBv1"  
  }  
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeVirtualNode](#)中的。

## describe-virtual-router

以下代码示例演示了如何使用 `describe-virtual-router`。

### AWS CLI

#### 描述虚拟路由器

以下 `describe-virtual-router` 示例返回有关指定虚拟路由器的详细信息。

```
aws appmesh describe-virtual-router \  
  --mesh-name app1 \  
  --virtual-router-name vrServiceB
```

输出：

```
{  
  "virtualRouter": {  
    "meshName": "app1",  
    "metadata": {  
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualRouter/  
vrServiceB",  
      "createdAt": 1563810546.59,  
      "lastUpdatedAt": 1563810546.59,  
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
      "version": 1  
    },  
    "spec": {  
      "listeners": [  
        {  
          "portMapping": {  
            "port": 80,  
            "protocol": "http"  
          }  
        }  
      ]  
    },  
    "status": {  
      "status": "ACTIVE"  
    },  
    "virtualRouterName": "vrServiceB"  
  }  
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟路由器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeVirtualRouter](#)中的。

## describe-virtual-service

以下代码示例演示了如何使用 describe-virtual-service。

### AWS CLI

#### 描述虚拟服务

以下 describe-virtual-service 示例返回有关指定虚拟服务的详细信息。

```
aws appmesh describe-virtual-service \  
  --mesh-name app1 \  
  --virtual-service-name serviceB.svc.cluster.local
```

输出：

```
{  
  "virtualService": {  
    "meshName": "app1",  
    "metadata": {  
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualService/  
serviceB.svc.cluster.local",  
      "createdAt": 1563908363.999,  
      "lastUpdatedAt": 1563908363.999,  
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
      "version": 1  
    },  
    "spec": {  
      "provider": {  
        "virtualRouter": {  
          "virtualRouterName": "vrServiceB"  
        }  
      }  
    },  
    "status": {  
      "status": "ACTIVE"  
    },  
    "virtualServiceName": "serviceB.svc.cluster.local"  
  }  
}
```

```
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟服务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeVirtualService](#)中的。

## list-meshes

以下代码示例演示了如何使用 `list-meshes`。

### AWS CLI

列出服务网格

以下 `list-meshes` 示例列出了当前 AWS 区域中的所有服务网格。

```
aws appmesh list-meshes
```

输出：

```
{
  "meshes": [
    {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1",
      "meshName": "app1"
    }
  ]
}
```

有关更多信息，请参阅 AWS App [Mesh 用户指南中的服务网格](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListMeshes](#)中的。

## list-routes

以下代码示例演示了如何使用 `list-routes`。

### AWS CLI

列出路线

以下 `list-routes` 示例列出了指定虚拟路由器的所有路由。

```
aws appmesh list-routes \  
  --mesh-name app1 \  
  --virtual-router-name vrServiceB
```

输出：

```
{  
  "routes": [  
    {  
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualRouter/  
vrServiceB/route/toVnServiceB",  
      "meshName": "app1",  
      "routeName": "toVnServiceB-weighted",  
      "virtualRouterName": "vrServiceB"  
    }  
  ]  
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[路由](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRoutes](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 list-tags-for-resource。

AWS CLI

列出资源的标签

以下list-tags-for-resource示例列出了分配给指定资源的所有标签。

```
aws appmesh list-tags-for-resource \  
  --resource-arn arn:aws:appmesh:us-east-1:123456789012:mesh/app1
```

输出：

```
{  
  "tags": [  
    {  
      "key": "key1",
```

```
        "value": "value1"
      },
      {
        "key": "key2",
        "value": "value2"
      },
      {
        "key": "key3",
        "value": "value3"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## list-virtual-nodes

以下代码示例演示了如何使用 `list-virtual-nodes`。

### AWS CLI

列出虚拟节点

以下`list-virtual-nodes`示例列出了指定服务网格中的所有虚拟节点。

```
aws appmesh list-virtual-nodes \
  --mesh-name app1
```

输出：

```
{
  "virtualNodes": [
    {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualNode/vnServiceBv1",
      "meshName": "app1",
      "virtualNodeName": "vnServiceBv1"
    },
    {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualNode/vnServiceBv2",
      "meshName": "app1",

```



```
        "virtualNodeName": "vnServiceBv2"
      }
    ]
  }
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListVirtualNodes](#)中的。

## list-virtual-routers

以下代码示例演示了如何使用 `list-virtual-routers`。

### AWS CLI

列出虚拟路由器

以下 `list-virtual-routers` 示例列出了指定服务网格中的所有虚拟路由器。

```
aws appmesh list-virtual-routers \
  --mesh-name app1
```

输出：

```
{
  "virtualRouters": [
    {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualRouter/vrServiceB",
      "meshName": "app1",
      "virtualRouterName": "vrServiceB"
    }
  ]
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟路由器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListVirtualRouters](#)中的。

## list-virtual-services

以下代码示例演示了如何使用 `list-virtual-services`。

## AWS CLI

### 列出虚拟服务

以下`list-virtual-services`示例列出了指定服务网格中的所有虚拟服务。

```
aws appmesh list-virtual-services \  
  --mesh-name app1
```

输出：

```
{  
  "virtualServices": [  
    {  
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualService/  
serviceA.svc.cluster.local",  
      "meshName": "app1",  
      "virtualServiceName": "serviceA.svc.cluster.local"  
    },  
    {  
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualService/  
serviceB.svc.cluster.local",  
      "meshName": "app1",  
      "virtualServiceName": "serviceB.svc.cluster.local"  
    }  
  ]  
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟服务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListVirtualServices](#)中的。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

### AWS CLI

#### 为资源添加标签

以下`tag-resource`示例将`key1`带有值的标签`value1`添加到指定资源。

```
aws appmesh tag-resource \  
  --resource-arn arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualService/  
serviceA.svc.cluster.local
```

```
--resource-arn arn:aws:appmesh:us-east-1:123456789012:mesh/app1 \  
--tags key=key1,value=value1
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

取消对资源的标记

以下untag-resource示例key1从指定资源中删除带有密钥的标签。

```
aws appmesh untag-resource \  
  --resource-arn arn:aws:appmesh:us-east-1:123456789012:mesh/app1 \  
  --tag-keys key1
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-mesh

以下代码示例演示了如何使用 update-mesh。

### AWS CLI

更新服务网格

以下update-mesh示例使用 JSON 输入文件更新服务网格，以允许所有外部出口流量通过 Envoy 代理保持不变。

```
aws appmesh update-mesh \  
  --cli-input-json file://update-mesh.json
```

update-mesh.json 的内容：

```
{  
  "meshName": "app1",
```

```
"spec": {
  "egressFilter": {
    "type": "ALLOW_ALL"
  }
}
```

输出：

```
{
  "mesh": {
    "meshName": "app1",
    "metadata": {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1",
      "createdAt": 1563809909.282,
      "lastUpdatedAt": 1563812829.687,
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version": 2
    },
    "spec": {
      "egressFilter": {
        "type": "ALLOW_ALL"
      }
    },
    "status": {
      "status": "ACTIVE"
    }
  }
}
```

有关更多信息，请参阅 AWS App [Mesh 用户指南中的服务](#) 网格。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateMesh](#) 中的。

## update-route

以下代码示例演示了如何使用 update-route。

### AWS CLI

#### 更新路线

以下 update-route 示例使用 JSON 输入文件更新路径的权重。

```
aws appmesh update-route \  
  --cli-input-json file://update-route-weighted.json
```

update-route-weighted.json 的内容：

```
{  
  "meshName": "app1",  
  "routeName": "toVnServiceB-weighted",  
  "spec": {  
    "httpRoute": {  
      "action": {  
        "weightedTargets": [  
          {  
            "virtualNode": "vnServiceBv1",  
            "weight": 80  
          },  
          {  
            "virtualNode": "vnServiceBv2",  
            "weight": 20  
          }  
        ]  
      },  
      "match": {  
        "prefix": "/"  
      }  
    }  
  },  
  "virtualRouterName": "vrServiceB"  
}
```

输出：

```
{  
  "route": {  
    "meshName": "app1",  
    "metadata": {  
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualRouter/  
vrServiceB/route/toVnServiceB-weighted",  
      "createdAt": 1563811384.015,  
      "lastUpdatedAt": 1563819600.022,  
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
      "version": 2  
    }  
  }  
}
```

```
    },
    "routeName": "toVnServiceB-weighted",
    "spec": {
      "httpRoute": {
        "action": {
          "weightedTargets": [
            {
              "virtualNode": "vnServiceBv1",
              "weight": 80
            },
            {
              "virtualNode": "vnServiceBv2",
              "weight": 20
            }
          ]
        },
        "match": {
          "prefix": "/"
        }
      }
    },
    "status": {
      "status": "ACTIVE"
    },
    "virtualRouterName": "vrServiceB"
  }
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[路由](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateRoute](#)中的。

## update-virtual-node

以下代码示例演示了如何使用 update-virtual-node。

### AWS CLI

#### 更新虚拟节点

以下 update-virtual-node 示例使用 JSON 输入文件向虚拟节点添加运行状况检查。

```
aws appmesh update-virtual-node \
```

```
--cli-input-json file://update-virtual-node.json
```

update-virtual-node.json 的内容：

```
{
  "clientToken": "500",
  "meshName": "app1",
  "spec": {
    "listeners": [
      {
        "healthCheck": {
          "healthyThreshold": 5,
          "intervalMillis": 10000,
          "path": "/",
          "port": 80,
          "protocol": "http",
          "timeoutMillis": 3000,
          "unhealthyThreshold": 3
        },
        "portMapping": {
          "port": 80,
          "protocol": "http"
        }
      }
    ],
    "serviceDiscovery": {
      "dns": {
        "hostname": "serviceBv1.svc.cluster.local"
      }
    }
  },
  "virtualNodeName": "vnServiceBv1"
}
```

输出：

```
{
  "virtualNode": {
    "meshName": "app1",
    "metadata": {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualNode/vnServiceBv1",
      "createdAt": 1563810019.874,

```

```
    "lastUpdatedAt": 1563819234.825,
    "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "version": 2
  },
  "spec": {
    "listeners": [
      {
        "healthCheck": {
          "healthyThreshold": 5,
          "intervalMillis": 10000,
          "path": "/",
          "port": 80,
          "protocol": "http",
          "timeoutMillis": 3000,
          "unhealthyThreshold": 3
        },
        "portMapping": {
          "port": 80,
          "protocol": "http"
        }
      }
    ],
    "serviceDiscovery": {
      "dns": {
        "hostname": "serviceBv1.svc.cluster.local"
      }
    }
  },
  "status": {
    "status": "ACTIVE"
  },
  "virtualNodeName": "vnServiceBv1"
}
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateVirtualNode](#)中的。

## update-virtual-router

以下代码示例演示了如何使用 update-virtual-router。



## AWS CLI

### 更新虚拟路由器

以下update-virtual-router示例使用 JSON 输入文件更新虚拟路由器侦听器端口。

```
aws appmesh update-virtual-router \  
  --cli-input-json file://update-virtual-router.json
```

update-virtual-router.json 的内容：

```
{  
  "meshName": "app1",  
  "spec": {  
    "listeners": [  
      {  
        "portMapping": {  
          "port": 8080,  
          "protocol": "http"  
        }  
      }  
    ]  
  },  
  "virtualRouterName": "vrServiceB"  
}
```

输出：

```
{  
  "virtualRouter": {  
    "meshName": "app1",  
    "metadata": {  
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualRouter/  
vrServiceB",  
      "createdAt": 1563810546.59,  
      "lastUpdatedAt": 1563819431.352,  
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
      "version": 2  
    },  
    "spec": {  
      "listeners": [  
        {
```

```
        "portMapping": {
            "port": 8080,
            "protocol": "http"
        }
    ]
},
"status": {
    "status": "ACTIVE"
},
"virtualRouterName": "vrServiceB"
}
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟路由器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateVirtualRouter](#)中的。

## update-virtual-service

以下代码示例演示了如何使用 update-virtual-service。

### AWS CLI

#### 更新虚拟服务

以下 update-virtual-service 示例使用 JSON 输入文件更新虚拟服务以使用虚拟路由器提供商。

```
aws appmesh update-virtual-service \
  --cli-input-json file://update-virtual-service.json
```

update-virtual-service.json 的内容：

```
{
  "meshName": "app1",
  "spec": {
    "provider": {
      "virtualRouter": {
        "virtualRouterName": "vrServiceA"
      }
    }
  }
}
```

```
    },
    "virtualServiceName": "serviceA.svc.cluster.local"
  }
```

输出：

```
{
  "virtualService": {
    "meshName": "app1",
    "metadata": {
      "arn": "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualService/
serviceA.svc.cluster.local",
      "createdAt": 1563810859.474,
      "lastUpdatedAt": 1563820257.411,
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version": 3
    },
    "spec": {
      "provider": {
        "virtualRouter": {
          "virtualRouterName": "vrServiceA"
        }
      }
    },
    "status": {
      "status": "ACTIVE"
    },
    "virtualServiceName": "serviceA.svc.cluster.local"
  }
}
```

有关更多信息，请参阅 AWS App Mesh 用户指南中的[虚拟服务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateVirtualService](#)中的。

## App Runner 示例使用 AWS CLI

以下代码示例向您展示了如何使用 with App Runner 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## associate-custom-domain

以下代码示例演示了如何使用 `associate-custom-domain`。

AWS CLI

将域名和 `www` 子域与服务相关联

以下 `associate-custom-domain` 示例将您控制的自定义域名与 App Runner 服务相关联。域名是根域 `example.com`，包括特殊情况的子域名。 `www.example.com`

```
aws apprunner associate-custom-domain \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-  
app/8fe1e10304f84fd2b0df550fe98a71fa",  
  "DomainName": "example.com",  
  "EnableWWWSubdomain": true  
}
```

输出：

```
{  
  "CustomDomain": {  
    "CertificateValidationRecords": [  
      {  
        "Name": "_70d3f50a94f7c72dc28784cf55db2f6b.example.com",  
        "Status": "PENDING_VALIDATION",  
      }  
    ]  
  }  
}
```

```

        "Type": "CNAME",
        "Value": "_1270c137383c6307b6832db02504c4b0.bsgbmzkfwj.acm-
validations.aws."
    },
    {
        "Name": "_287870d3f50a94f7c72dc4cf55db2f6b.www.example.com",
        "Status": "PENDING_VALIDATION",
        "Type": "CNAME",
        "Value": "_832db01270c137383c6307b62504c4b0.mzkbsgbfwj.acm-
validations.aws."
    }
],
"DomainName": "example.com",
"EnableWWWSubdomain": true,
"Status": "CREATING"
},
"DNSTarget": "psbqam834h.us-east-1.awsapprunner.com",
"ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateCustomDomain](#)中的。

## create-auto-scaling-configuration

以下代码示例演示了如何使用 create-auto-scaling-configuration。

### AWS CLI

#### 创建高可用性 auto Scaling 配置

以下create-auto-scaling-configuration示例通过将设置MinSize为 5 来创建针对高可用性进行优化的 auto Scaling 配置。使用此配置，App Runner 会尝试将您的服务实例分布在尽可能多的可用区（最多五个可用区），具体视 AWS 区域而定。

该调用将返回一个其他设置为默认值的AutoScalingConfiguration对象。在示例中，这是第一次调用创建名为的配置high-availability。修订版设置为 1，这是最新的修订版。

```
aws apprunner create-auto-scaling-configuration \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "AutoScalingConfigurationName": "high-availability",
  "MinSize": 5
}
```

输出：

```
{
  "AutoScalingConfiguration": {
    "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-1:123456789012:autoscalingconfiguration/high-
availability/1/2f50e7656d7819fead0f59672e68042e",
    "AutoScalingConfigurationName": "high-availability",
    "AutoScalingConfigurationRevision": 1,
    "CreatedAt": "2020-11-03T00:29:17Z",
    "Latest": true,
    "Status": "ACTIVE",
    "MaxConcurrency": 100,
    "MaxSize": 50,
    "MinSize": 5
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAutoScalingConfiguration](#)中的。

## create-connection

以下代码示例演示了如何使用 create-connection。

### AWS CLI

#### 创建 GitHub 连接

以下create-connection示例创建了与私有 GitHub 代码存储库的连接。成功呼叫后的连接状态为PENDING\_HANDSHAKE。这是因为仍未与提供商进行身份验证握手。使用 App Runner 控制台完成握手。

```
aws apprunner create-connection \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "ConnectionName": "my-github-connection",
  "ProviderType": "GITHUB"
}
```

输出：

```
{
  "Connection": {
    "ConnectionArn": "arn:aws:apprunner:us-east-1:123456789012:connection/my-github-connection",
    "ConnectionName": "my-github-connection",
    "Status": "PENDING_HANDSHAKE",
    "CreatedAt": "2020-11-03T00:32:51Z",
    "ProviderType": "GITHUB"
  }
}
```

有关更多信息，请参阅 [《App Runner 开发者指南》中的管理AWS App Runner 连接](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateConnection](#)中的。

## create-service

以下代码示例演示了如何使用 create-service。

### AWS CLI

示例 1：创建源代码存储库服务

以下create-service示例基于 Python 源代码存储库创建了 App Runner 服务。

```
aws apprunner create-service \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "ServiceName": "python-app",
  "SourceConfiguration": {
    "AuthenticationConfiguration": {
```

```

    "ConnectionArn": "arn:aws:apprunner:us-east-1:123456789012:connection/
my-github-connection/e7656250f67242d7819feade6800f59e"
  },
  "AutoDeploymentsEnabled": true,
  "CodeRepository": {
    "RepositoryUrl": "https://github.com/my-account/python-hello",
    "SourceCodeVersion": {
      "Type": "BRANCH",
      "Value": "main"
    },
  },
  "CodeConfiguration": {
    "ConfigurationSource": "API",
    "CodeConfigurationValues": {
      "Runtime": "PYTHON_3",
      "BuildCommand": "pip install -r requirements.txt",
      "StartCommand": "python server.py",
      "Port": "8080",
      "RuntimeEnvironmentVariables": [
        {
          "NAME": "Jane"
        }
      ]
    }
  }
}
},
"InstanceConfiguration": {
  "CPU": "1 vCPU",
  "Memory": "3 GB"
}
}

```

输出：

```

{
  "OperationId": "17fe9f55-7e91-4097-b243-fcabbb69a4cf",
  "Service": {
    "CreatedAt": "2020-11-20T19:05:25Z",
    "UpdatedAt": "2020-11-20T19:05:25Z",
    "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceName": "python-app",
  }
}

```



```
"ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
"SourceConfiguration": {
  "AuthenticationConfiguration": {
    "ConnectionArn": "arn:aws:apprunner:us-
east-1:123456789012:connection/my-github-connection/
e7656250f67242d7819feade6800f59e"
  },
  "AutoDeploymentsEnabled": true,
  "CodeRepository": {
    "CodeConfiguration": {
      "CodeConfigurationValues": {
        "BuildCommand": "pip install -r requirements.txt",
        "Port": "8080",
        "Runtime": "PYTHON_3",
        "RuntimeEnvironmentVariables": [
          {
            "NAME": "Jane"
          }
        ],
        "StartCommand": "python server.py"
      },
      "ConfigurationSource": "Api"
    },
    "RepositoryUrl": "https://github.com/my-account/python-hello",
    "SourceCodeVersion": {
      "Type": "BRANCH",
      "Value": "main"
    }
  }
},
"Status": "OPERATION_IN_PROGRESS",
"InstanceConfiguration": {
  "CPU": "1 vCPU",
  "Memory": "3 GB"
}
}
```

## 示例 2：创建源代码存储库服务

以下create-service示例基于 Python 源代码存储库创建了 App Runner 服务。

```
aws apprunner create-service \
```

```
--cli-input-json file://input.json
```

input.json 的内容：

```
{
  "ServiceName": "python-app",
  "SourceConfiguration": {
    "AuthenticationConfiguration": {
      "ConnectionArn": "arn:aws:apprunner:us-east-1:123456789012:connection/my-github-connection/e7656250f67242d7819feade6800f59e"
    },
    "AutoDeploymentsEnabled": true,
    "CodeRepository": {
      "RepositoryUrl": "https://github.com/my-account/python-hello",
      "SourceCodeVersion": {
        "Type": "BRANCH",
        "Value": "main"
      }
    },
    "CodeConfiguration": {
      "ConfigurationSource": "API",
      "CodeConfigurationValues": {
        "Runtime": "PYTHON_3",
        "BuildCommand": "pip install -r requirements.txt",
        "StartCommand": "python server.py",
        "Port": "8080",
        "RuntimeEnvironmentVariables": [
          {
            "NAME": "Jane"
          }
        ]
      }
    }
  },
  "InstanceConfiguration": {
    "CPU": "1 vCPU",
    "Memory": "3 GB"
  }
}
```

输出：

```
{
```

```
"OperationId": "17fe9f55-7e91-4097-b243-fcabbb69a4cf",
"Service": {
  "CreatedAt": "2020-11-20T19:05:25Z",
  "UpdatedAt": "2020-11-20T19:05:25Z",
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
  "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
  "ServiceName": "python-app",
  "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
  "SourceConfiguration": {
    "AuthenticationConfiguration": {
      "ConnectionArn": "arn:aws:apprunner:us-
east-1:123456789012:connection/my-github-connection/
e7656250f67242d7819feade6800f59e"
    },
    "AutoDeploymentsEnabled": true,
    "CodeRepository": {
      "CodeConfiguration": {
        "CodeConfigurationValues": {
          "BuildCommand": "pip install -r requirements.txt",
          "Port": "8080",
          "Runtime": "PYTHON_3",
          "RuntimeEnvironmentVariables": [
            {
              "NAME": "Jane"
            }
          ],
          "StartCommand": "python server.py"
        },
        "ConfigurationSource": "Api"
      },
      "RepositoryUrl": "https://github.com/my-account/python-hello",
      "SourceCodeVersion": {
        "Type": "BRANCH",
        "Value": "main"
      }
    }
  },
  "Status": "OPERATION_IN_PROGRESS",
  "InstanceConfiguration": {
    "CPU": "1 vCPU",
    "Memory": "3 GB"
  }
}
```

```
}
```

### 示例 3：创建源图像存储库服务

以下 `create-service` 示例基于存储在弹性容器注册表 (ECR) 中的图像创建 App Runner 服务。

```
aws apprunner create-service \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "ServiceName": "golang-container-app",  
  "SourceConfiguration": {  
    "AuthenticationConfiguration": {  
      "AccessRoleArn": "arn:aws:iam::123456789012:role/my-ecr-role"  
    },  
    "AutoDeploymentsEnabled": true,  
    "ImageRepository": {  
      "ImageIdentifier": "123456789012.dkr.ecr.us-east-1.amazonaws.com/golang-  
app:latest",  
      "ImageConfiguration": {  
        "Port": "8080",  
        "RuntimeEnvironmentVariables": [  
          {  
            "NAME": "Jane"  
          }  
        ]  
      },  
      "ImageRepositoryType": "ECR"  
    }  
  },  
  "InstanceConfiguration": {  
    "CPU": "1 vCPU",  
    "Memory": "3 GB"  
  }  
}
```

输出：

```
{  
  "OperationId": "17fe9f55-7e91-4097-b243-fcabbb69a4cf",  
  "Service": {
```

```
"CreatedAt": "2020-11-06T23:15:30Z",
"UpdatedAt": "2020-11-06T23:15:30Z",
"ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/golang-
container-app/51728f8a20ce46d39b25398a6c8e9d1a",
"ServiceId": "51728f8a20ce46d39b25398a6c8e9d1a",
"ServiceName": "golang-container-app",
"ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
"SourceConfiguration": {
  "AuthenticationConfiguration": {
    "AccessRoleArn": "arn:aws:iam::123456789012:role/my-ecr-role"
  },
  "AutoDeploymentsEnabled": true,
  "ImageRepository": {
    "ImageIdentifier": "123456789012.dkr.ecr.us-east-1.amazonaws.com/
golang-app:latest",
    "ImageConfiguration": {
      "Port": "8080",
      "RuntimeEnvironmentVariables": [
        {
          "NAME": "Jane"
        }
      ]
    },
    "ImageRepositoryType": "ECR"
  }
},
"Status": "OPERATION_IN_PROGRESS",
"InstanceConfiguration": {
  "CPU": "1 vCPU",
  "Memory": "3 GB"
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateService](#)中的。

## delete-auto-scaling-configuration

以下代码示例演示了如何使用 delete-auto-scaling-configuration。

### AWS CLI

示例 1：删除 auto Scaling 配置的最新活动版本

以下delete-auto-scaling-configuration示例删除了 App Runner 自动缩放配置的最新活动版本。要删除最新的有效修订版，请指定以配置名称结尾的 Amazon 资源名称 (ARN)，不包括修订组件。

在示例中，在此操作之前存在两个修订版。因此，修订版 2 (最新) 已删除。但是，它现在会显示"Latest": false，因为删除后，它不再是最新的有效修订版。

```
aws apprunner delete-auto-scaling-configuration \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{  
  "AutoScalingConfigurationArn": "arn:aws:apprunner:us-  
east-1:123456789012:autoscalingconfiguration/high-availability"  
}
```

输出：

```
{  
  "AutoScalingConfiguration": {  
    "AutoScalingConfigurationArn": "arn:aws:apprunner:us-  
east-1:123456789012:autoscalingconfiguration/high-availability/2/  
e76562f50d78042e819fead0f59672e6",  
    "AutoScalingConfigurationName": "high-availability",  
    "AutoScalingConfigurationRevision": 2,  
    "CreatedAt": "2021-02-25T17:42:59Z",  
    "DeletedAt": "2021-03-02T08:07:06Z",  
    "Latest": false,  
    "Status": "INACTIVE",  
    "MaxConcurrency": 30,  
    "MaxSize": 90,  
    "MinSize": 5  
  }  
}
```

### 示例 2：删除 auto Scaling 配置的特定修订版

以下delete-auto-scaling-configuration示例删除了 App Runner 自动缩放配置的特定版本。要删除特定修订版，请指定包含修订号的 ARN。

在示例中，在此操作之前存在多个修订版。该操作会删除修订1。

```
aws apprunner delete-auto-scaling-configuration \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{  
  "AutoScalingConfigurationArn": "arn:aws:apprunner:us-  
east-1:123456789012:autoscalingconfiguration/high-availability/1"  
}
```

输出：

```
{  
  "AutoScalingConfiguration": {  
    "AutoScalingConfigurationArn": "arn:aws:apprunner:us-  
east-1:123456789012:autoscalingconfiguration/high-  
availability/1/2f50e7656d7819fead0f59672e68042e",  
    "AutoScalingConfigurationName": "high-availability",  
    "AutoScalingConfigurationRevision": 1,  
    "CreatedAt": "2020-11-03T00:29:17Z",  
    "DeletedAt": "2021-03-02T08:07:06Z",  
    "Latest": false,  
    "Status": "INACTIVE",  
    "MaxConcurrency": 100,  
    "MaxSize": 50,  
    "MinSize": 5  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAutoScalingConfiguration](#)中的。

## delete-connection

以下代码示例演示了如何使用 delete-connection。

### AWS CLI

#### 删除连接

以下delete-connection示例删除了 App Runner 连接。成功呼叫后的连接状态为DELETED。这是因为该连接不再可用。

```
aws apprunner delete-connection \  
  --cli-input-json file://input.json
```

input.json 的内容 :

```
{  
  "ConnectionArn": "arn:aws:apprunner:us-east-1:123456789012:connection/my-github-  
connection"  
}
```

输出 :

```
{  
  "Connection": {  
    "ConnectionArn": "arn:aws:apprunner:us-east-1:123456789012:connection/my-  
github-connection",  
    "ConnectionName": "my-github-connection",  
    "Status": "DELETED",  
    "CreatedAt": "2020-11-03T00:32:51Z",  
    "ProviderType": "GITHUB"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteConnection](#)中的。

## delete-service

以下代码示例演示了如何使用 delete-service。

### AWS CLI

#### 删除服务

以下delete-service示例删除了 App Runner 服务。

```
aws apprunner delete-service \  
  --cli-input-json file://input.json
```

input.json 的内容 :



```
{
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa"
}
```

输出：

```
{
  "OperationId": "17fe9f55-7e91-4097-b243-fcabbb69a4cf",
  "Service": {
    "CreatedAt": "2020-11-20T19:05:25Z",
    "UpdatedAt": "2020-11-20T19:05:25Z",
    "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceName": "python-app",
    "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
    "SourceConfiguration": {
      "AuthenticationConfiguration": {
        "ConnectionArn": "arn:aws:apprunner:us-
east-1:123456789012:connection/my-github-connection/
e7656250f67242d7819feade6800f59e"
      },
      "AutoDeploymentsEnabled": true,
      "CodeRepository": {
        "CodeConfiguration": {
          "CodeConfigurationValues": {
            "BuildCommand": "pip install -r requirements.txt",
            "Port": "8080",
            "Runtime": "PYTHON_3",
            "RuntimeEnvironmentVariables": [
              {
                "NAME": "Jane"
              }
            ],
            "StartCommand": "python server.py"
          },
          "ConfigurationSource": "Api"
        },
        "RepositoryUrl": "https://github.com/my-account/python-hello",
        "SourceCodeVersion": {
          "Type": "BRANCH",
          "Value": "main"
        }
      }
    }
  }
}
```

```

        }
      }
    },
    "Status": "OPERATION_IN_PROGRESS",
    "InstanceConfiguration": {
      "CPU": "1 vCPU",
      "Memory": "3 GB"
    }
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteService](#)中的。

## describe-auto-scaling-configuration

以下代码示例演示了如何使用 `describe-auto-scaling-configuration`。

### AWS CLI

示例 1：描述 auto Scaling 配置的最新活动版本

以下 `describe-auto-scaling-configuration` 示例描述了 App Runner 自动缩放配置的最新活动版本。要描述最新的活动修订版，请指定一个以配置名称结尾的 ARN，不包括修订版组件。

在示例中，存在两个修订版。因此，描述了修订版2（最新）。将显示生成的对象 `"Latest": true`。

```
aws apprunner describe-auto-scaling-configuration \
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{
  "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-1:123456789012:autoscalingconfiguration/high-availability"
}
```

输出：

```
{
```

```
"AutoScalingConfiguration": {
  "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-1:123456789012:autoscalingconfiguration/high-availability/2/
e76562f50d78042e819fead0f59672e6",
  "AutoScalingConfigurationName": "high-availability",
  "AutoScalingConfigurationRevision": 2,
  "CreatedAt": "2021-02-25T17:42:59Z",
  "Latest": true,
  "Status": "ACTIVE",
  "MaxConcurrency": 30,
  "MaxSize": 90,
  "MinSize": 5
}
}
```

## 示例 2：描述 auto Scaling 配置的特定版本

以下describe-auto-scaling-configuration示例描述了 App Runner 自动缩放配置的特定版本。要描述特定的修订版，请指定包含修订号的 ARN。

在示例中，存在多个修订版本并查询了修订版本1。将显示生成的对象"Latest": false。

```
aws apprunner describe-auto-scaling-configuration \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-1:123456789012:autoscalingconfiguration/high-availability/1"
}
```

输出：

```
{
  "AutoScalingConfiguration": {
    "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-1:123456789012:autoscalingconfiguration/high-
availability/1/2f50e7656d7819fead0f59672e68042e",
    "AutoScalingConfigurationName": "high-availability",
    "AutoScalingConfigurationRevision": 1,
```

```
    "CreatedAt": "2020-11-03T00:29:17Z",
    "Latest": false,
    "Status": "ACTIVE",
    "MaxConcurrency": 100,
    "MaxSize": 50,
    "MinSize": 5
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAutoScalingConfiguration](#)中的。

## describe-custom-domains

以下代码示例演示了如何使用 describe-custom-domains。

### AWS CLI

获取与服务关联的自定义域名的描述

以下describe-custom-domains示例获取与 App Runner 服务关联的自定义域名的描述和状态。

```
aws apprunner describe-custom-domains \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
  "DomainName": "example.com",
  "EnableWWWSubdomain": true
}
```

输出：

```
{
  "CustomDomains": [
    {
      "CertificateValidationRecords": [
        {
```

```

        "Name": "_70d3f50a94f7c72dc28784cf55db2f6b.example.com",
        "Status": "PENDING_VALIDATION",
        "Type": "CNAME",
        "Value": "_1270c137383c6307b6832db02504c4b0.bsgbmzkfwj.acm-
validations.aws."
    },
    {
        "Name": "_287870d3f50a94f7c72dc4cf55db2f6b.www.example.com",
        "Status": "PENDING_VALIDATION",
        "Type": "CNAME",
        "Value": "_832db01270c137383c6307b62504c4b0.mzkbsgbfwj.acm-
validations.aws."
    }
],
"DomainName": "example.com",
"EnableWWWSubdomain": true,
"Status": "PENDING_CERTIFICATE_DNS_VALIDATION"
},
{
    "CertificateValidationRecords": [
        {
            "Name": "_a94f784c70d3f507c72dc28f55db2f6b.deals.example.com",
            "Status": "SUCCESS",
            "Type": "CNAME",
            "Value": "_2db02504c1270c137383c6307b6834b0.bsgbmzkfwj.acm-
validations.aws."
        }
    ],
    "DomainName": "deals.example.com",
    "EnableWWWSubdomain": false,
    "Status": "ACTIVE"
}
],
"DNSTarget": "psbqam834h.us-east-1.awsapprunner.com",
"ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCustomDomains](#)中的。

## describe-service

以下代码示例演示了如何使用 describe-service。

## AWS CLI

### 描述一项服务

以下describe-service示例获取了 App Runner 服务的描述。

```
aws apprunner describe-service \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{  
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-  
app/8fe1e10304f84fd2b0df550fe98a71fa"  
}
```

输出：

```
{  
  "Service": {  
    "CreatedAt": "2020-11-20T19:05:25Z",  
    "UpdatedAt": "2020-11-20T19:05:25Z",  
    "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-  
app/8fe1e10304f84fd2b0df550fe98a71fa",  
    "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",  
    "ServiceName": "python-app",  
    "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",  
    "SourceConfiguration": {  
      "AuthenticationConfiguration": {  
        "ConnectionArn": "arn:aws:apprunner:us-  
east-1:123456789012:connection/my-github-connection/  
e7656250f67242d7819feade6800f59e"  
      },  
      "AutoDeploymentsEnabled": true,  
      "CodeRepository": {  
        "CodeConfiguration": {  
          "CodeConfigurationValues": {  
            "BuildCommand": "pip install -r requirements.txt",  
            "Port": "8080",  
            "Runtime": "PYTHON_3",  
            "RuntimeEnvironmentVariables": [  
              {  
                "NAME": "Jane"  
              }  
            ]  
          }  
        }  
      }  
    }  
  }  
}
```

```

        }
      ],
      "StartCommand": "python server.py"
    },
    "ConfigurationSource": "Api"
  },
  "RepositoryUrl": "https://github.com/my-account/python-hello",
  "SourceCodeVersion": {
    "Type": "BRANCH",
    "Value": "main"
  }
}
},
"Status": "RUNNING",
"InstanceConfiguration": {
  "CPU": "1 vCPU",
  "Memory": "3 GB"
}
}
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeService](#)中的。

## disassociate-custom-domain

以下代码示例演示了如何使用 disassociate-custom-domain。

### AWS CLI

#### 取消域名与服务的关联

以下disassociate-custom-domain示例取消该域example.com与 App Runner 服务的关联。该调用还会取消与根域名关联www.example.com的子域的关联。

```
aws apprunner disassociate-custom-domain \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
```

```
"DomainName": "example.com"
}
```

输出：

```
{
  "CustomDomain": {
    "CertificateValidationRecords": [
      {
        "Name": "_70d3f50a94f7c72dc28784cf55db2f6b.example.com",
        "Status": "PENDING_VALIDATION",
        "Type": "CNAME",
        "Value": "_1270c137383c6307b6832db02504c4b0.bsgbmzkfwj.acm-
validations.aws."
      },
      {
        "Name": "_287870d3f50a94f7c72dc4cf55db2f6b.www.example.com",
        "Status": "PENDING_VALIDATION",
        "Type": "CNAME",
        "Value": "_832db01270c137383c6307b62504c4b0.mzkbsgbfwj.acm-
validations.aws."
      }
    ],
    "DomainName": "example.com",
    "EnableWWWSubdomain": true,
    "Status": "DELETING"
  },
  "DNSTarget": "psbqam834h.us-east-1.awsapprunner.com",
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateCustomDomain](#)中的。

## list-auto-scaling-configurations

以下代码示例演示了如何使用 list-auto-scaling-configurations。

### AWS CLI

要获取 App Runner 自动缩放配置的分页列表



以下`list-auto-scaling-configurations`示例列出了您 AWS 账户中的所有 App Runner 自动缩放配置。每个响应中最多列出五个 auto Scaling 配置。`AutoScalingConfigurationName`并且`LatestOnly`未指定。它们的默认值会导致列出所有活动配置的最新版本。

在此示例中，响应包含两个结果，但没有其他结果，因此`NextToken`不返回任何结果。

```
aws apprunner list-auto-scaling-configurations \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "MaxResults": 5  
}
```

输出：

```
{  
  "AutoScalingConfigurationSummaryList": [  
    {  
      "AutoScalingConfigurationArn": "arn:aws:apprunner:us-  
east-1:123456789012:autoscalingconfiguration/high-availability/2/  
e76562f50d78042e819fead0f59672e6",  
      "AutoScalingConfigurationName": "high-availability",  
      "AutoScalingConfigurationRevision": 2  
    },  
    {  
      "AutoScalingConfigurationArn": "arn:aws:apprunner:us-  
east-1:123456789012:autoscalingconfiguration/low-  
cost/1/50d7804e7656fead0f59672e62f2e819",  
      "AutoScalingConfigurationName": "low-cost",  
      "AutoScalingConfigurationRevision": 1  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAutoScalingConfigurations](#)中的。

## list-connections

以下代码示例演示了如何使用 `list-connections`。

### AWS CLI

#### 示例 1：列出所有连接

以下 `list-connections` 示例列出了该 AWS 账户中的所有 App Runner 连接。

```
aws apprunner list-connections
```

输出：

```
{
  "ConnectionSummaryList": [
    {
      "ConnectionArn": "arn:aws:apprunner:us-east-1:123456789012:connection/my-github-connection",
      "ConnectionName": "my-github-connection",
      "Status": "AVAILABLE",
      "CreatedAt": "2020-11-03T00:32:51Z",
      "ProviderType": "GITHUB"
    },
    {
      "ConnectionArn": "arn:aws:apprunner:us-east-1:123456789012:connection/my-github-org-connection",
      "ConnectionName": "my-github-org-connection",
      "Status": "AVAILABLE",
      "CreatedAt": "2020-11-03T02:54:17Z",
      "ProviderType": "GITHUB"
    }
  ]
}
```

#### 示例 2：按名称列出连接

以下 `list-connections` 示例按名称列出了一个连接。

```
aws apprunner list-connections \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "ConnectionName": "my-github-org-connection"
}
```

输出：

```
{
  "ConnectionSummaryList": [
    {
      "ConnectionArn": "arn:aws:apprunner:us-east-1:123456789012:connection/
my-github-org-connection",
      "ConnectionName": "my-github-org-connection",
      "Status": "AVAILABLE",
      "CreatedAt": "2020-11-03T02:54:17Z",
      "ProviderType": "GITHUB"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListConnections](#)中的。

## list-operations

以下代码示例演示了如何使用 list-operations。

### AWS CLI

列出在服务上发生的操作

以下list-operations示例列出了迄今为止在 App Runner 服务上发生的所有操作。在此示例中，该服务是新的，并且只CREATE\_SERVICE发生了单一类型的操作。

```
aws apprunner list-operations \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
```

```
"ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa"
}
```

输出：

```
{
  "OperationSummaryList": [
    {
      "EndedAt": 1606156217,
      "Id": "17fe9f55-7e91-4097-b243-fcabbb69a4cf",
      "StartedAt": 1606156014,
      "Status": "SUCCEEDED",
      "TargetArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
      "Type": "CREATE_SERVICE",
      "UpdatedAt": 1606156217
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListOperations](#)中的。

## list-services

以下代码示例演示了如何使用 `list-services`。

### AWS CLI

获取 App Runner 服务的分页列表

以下`list-services`示例列出了该 AWS 账户中的所有 App Runner 服务。每个响应中最多列出两项服务。此示例显示了第一个请求。响应包括两个结果和一个可在下一个请求中使用的标记。如果后续响应不包含令牌，则表示所有服务都已列出。

```
aws apprunner list-services \
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{
```

```
"MaxResults": 2
}
```

输出：

```
{
  "NextToken":
  "eyJJdDhN0b21lckFjY291bnRjZCI6IjI3MDIwNTQwMjg0NSIsI1NlcnZpY2VtdGF0dXNDb2RlIjojIUFJcVklTSU90SU
  "ServiceSummaryList": [
    {
      "CreatedAt": "2020-11-20T19:05:25Z",
      "UpdatedAt": "2020-11-23T12:41:37Z",
      "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
      "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
      "ServiceName": "python-app",
      "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
      "Status": "RUNNING"
    },
    {
      "CreatedAt": "2020-11-06T23:15:30Z",
      "UpdatedAt": "2020-11-23T13:21:22Z",
      "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/golang-
container-app/ab8f94cfe29a460fb8760afd2ee87555",
      "ServiceId": "ab8f94cfe29a460fb8760afd2ee87555",
      "ServiceName": "golang-container-app",
      "ServiceUrl": "e2m8rrrx33.us-east-1.awsapprunner.com",
      "Status": "RUNNING"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListServices](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出与 App Runner 服务关联的标签

以下`list-tags-for-resource`示例列出了与 App Runner 服务关联的所有标签。

```
aws apprunner list-tags-for-resource \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{  
  "ResourceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-  
app/8fe1e10304f84fd2b0df550fe98a71fa"  
}
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "Department",  
      "Value": "Retail"  
    },  
    {  
      "Key": "CustomerId",  
      "Value": "56439872357912"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## pause-service

以下代码示例演示了如何使用 pause-service。

### AWS CLI

#### 暂停服务

以下pause-service示例暂停了 App Runner 服务。

```
aws apprunner pause-service \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa"
}
```

输出：

```
{
  "OperationId": "17fe9f55-7e91-4097-b243-fcabbb69a4cf",
  "Service": {
    "CreatedAt": "2020-11-20T19:05:25Z",
    "UpdatedAt": "2020-11-23T12:41:37Z",
    "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceName": "python-app",
    "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
    "SourceConfiguration": {
      "AuthenticationConfiguration": {
        "ConnectionArn": "arn:aws:apprunner:us-
east-1:123456789012:connection/my-github-connection/
e7656250f67242d7819feade6800f59e"
      },
      "AutoDeploymentsEnabled": true,
      "CodeRepository": {
        "CodeConfiguration": {
          "CodeConfigurationValues": {
            "BuildCommand": "pip install -r requirements.txt",
            "Port": "8080",
            "Runtime": "PYTHON_3",
            "RuntimeEnvironmentVariables": [
              {
                "NAME": "Jane"
              }
            ],
            "StartCommand": "python server.py"
          },
          "ConfigurationSource": "Api"
        },
        "RepositoryUrl": "https://github.com/my-account/python-hello",
        "SourceCodeVersion": {
```

```
        "Type": "BRANCH",
        "Value": "main"
      }
    },
    "Status": "OPERATION_IN_PROGRESS",
    "InstanceConfiguration": {
      "CPU": "1 vCPU",
      "Memory": "3 GB"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PauseService](#)中的。

## resume-service

以下代码示例演示了如何使用 `resume-service`。

### AWS CLI

#### 恢复服务

以下 `resume-service` 示例恢复 App Runner 服务。

```
aws apprunner resume-service \
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa"
}
```

输出：

```
{
  "OperationId": "17fe9f55-7e91-4097-b243-fcabbb69a4cf",
  "Service": {
    "CreatedAt": "2020-11-20T19:05:25Z",
```



```

    "UpdatedAt": "2020-11-23T12:41:37Z",
    "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceName": "python-app",
    "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
    "SourceConfiguration": {
      "AuthenticationConfiguration": {
        "ConnectionArn": "arn:aws:apprunner:us-
east-1:123456789012:connection/my-github-connection/
e7656250f67242d7819feade6800f59e"
      },
      "AutoDeploymentsEnabled": true,
      "CodeRepository": {
        "CodeConfiguration": {
          "CodeConfigurationValues": {
            "BuildCommand": "pip install -r requirements.txt",
            "Port": "8080",
            "Runtime": "PYTHON_3",
            "RuntimeEnvironmentVariables": [
              {
                "NAME": "Jane"
              }
            ],
            "StartCommand": "python server.py"
          },
          "ConfigurationSource": "Api"
        },
        "RepositoryUrl": "https://github.com/my-account/python-hello",
        "SourceCodeVersion": {
          "Type": "BRANCH",
          "Value": "main"
        }
      }
    },
    "Status": "OPERATION_IN_PROGRESS",
    "InstanceConfiguration": {
      "CPU": "1 vCPU",
      "Memory": "3 GB"
    }
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ResumeService](#)中的。

## start-deployment

以下代码示例演示了如何使用 start-deployment。

### AWS CLI

#### 启动手动部署

以下start-deployment示例执行对 App Runner 服务的手动部署。

```
aws apprunner start-deployment \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{  
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-  
app/8fe1e10304f84fd2b0df550fe98a71fa"  
}
```

输出：

```
{  
  "OperationId": "853a7d5b-fc9f-4730-831b-fd8037ab832a"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartDeployment](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

#### 向 App Runner 服务添加标签

以下tag-resource示例向 App Runner 服务添加了两个标签。

```
aws apprunner tag-resource \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{
  "ResourceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
  "Tags": [
    {
      "Key": "Department",
      "Value": "Retail"
    },
    {
      "Key": "CustomerId",
      "Value": "56439872357912"
    }
  ]
}
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 `untag-resource`。

### AWS CLI

从 App Runner 服务中移除标签

以下`untag-resource`示例从 App Runner 服务中删除了两个标签。

```
aws apprunner untag-resource \
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{
  "ResourceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
  "TagKeys": [
    "Department",
    "CustomerId"
  ]
}
```

```
]
}
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-service

以下代码示例演示了如何使用 update-service。

### AWS CLI

#### 更新内存大小

以下update-service示例将 App Runner 服务的实例的内存大小（缩放单位）更新为 2048 MiB。

调用成功后，App Runner 将启动异步更新进程。调用返回的Service结构反映了此调用所应用的新内存值。

```
aws apprunner update-service \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{  
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-  
app/8fe1e10304f84fd2b0df550fe98a71fa",  
  "InstanceConfiguration": {  
    "Memory": "4 GB"  
  }  
}
```

输出：

```
{  
  "OperationId": "17fe9f55-7e91-4097-b243-fcabbb69a4cf",  
  "Service": {  
    "CreatedAt": "2020-11-20T19:05:25Z",
```

```
    "UpdatedAt": "2020-11-23T12:41:37Z",
    "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceName": "python-app",
    "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
    "SourceConfiguration": {
      "AuthenticationConfiguration": {
        "ConnectionArn": "arn:aws:apprunner:us-
east-1:123456789012:connection/my-github-connection/
e7656250f67242d7819feade6800f59e"
      },
      "AutoDeploymentsEnabled": true,
      "CodeRepository": {
        "CodeConfiguration": {
          "CodeConfigurationValues": {
            "BuildCommand": "pip install -r requirements.txt",
            "Port": "8080",
            "Runtime": "PYTHON_3",
            "RuntimeEnvironmentVariables": [
              {
                "NAME": "Jane"
              }
            ],
            "StartCommand": "python server.py"
          },
          "ConfigurationSource": "Api"
        },
        "RepositoryUrl": "https://github.com/my-account/python-hello",
        "SourceCodeVersion": {
          "Type": "BRANCH",
          "Value": "main"
        }
      }
    },
    "Status": "OPERATION_IN_PROGRESS",
    "InstanceConfiguration": {
      "CPU": "1 vCPU",
      "Memory": "4 GB"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateService](#)中的。

## AWS AppConfig 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS AppConfig。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-application**

以下代码示例演示了如何使用 `create-application`。

AWS CLI

创建应用程序

以下 `create-application` 示例在中创建了一个应用程序 AWS AppConfig。

```
aws appconfig create-application \  
  --name "example-application" \  
  --description "An application used for creating an example."
```

输出：

```
{  
  "Description": "An application used for creating an example.",  
  "Id": "339ohji",  
  "Name": "example-application"  
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 1：创建 AWS AppConfig 应用程序](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateApplication](#)中的。

## create-configuration-profile

以下代码示例演示了如何使用 create-configuration-profile。

### AWS CLI

#### 创建配置文件

以下create-configuration-profile示例使用存储在 Parameter Store ( Systems Manager 的一项功能 ) 中的配置创建配置文件。

```
aws appconfig create-configuration-profile \  
  --application-id "339ohji" \  
  --name "Example-Configuration-Profile" \  
  --location-uri "ssm-parameter://Example-Parameter" \  
  --retrieval-role-arn "arn:aws:iam::111122223333:role/Example-App-Config-Role"
```

输出：

```
{  
  "ApplicationId": "339ohji",  
  "Description": null,  
  "Id": "ur8hx2f",  
  "LocationUri": "ssm-parameter://Example-Parameter",  
  "Name": "Example-Configuration-Profile",  
  "RetrievalRoleArn": "arn:aws:iam::111122223333:role/Example-App-Config-Role",  
  "Type": null,  
  "Validators": null  
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 3：创建配置和配置配置文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateConfigurationProfile](#)中的。

## create-environment

以下代码示例演示了如何使用 create-environment。

## AWS CLI

### 创建环境

以下 `create-environment` 示例使用您使用 `create-application` AWS AppConfig 创建的应用程序创建了一个名为 `Example-Environment` 的环境。

```
aws appconfig create-environment \  
  --application-id "339ohji" \  
  --name "Example-Environment"
```

输出：

```
{  
  "ApplicationId": "339ohji",  
  "Description": null,  
  "Id": "54j1r29",  
  "Monitors": null,  
  "Name": "Example-Environment",  
  "State": "ReadyForDeployment"  
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的 [步骤 2：创建环境](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateEnvironment](#) 中的。

## **create-extension-association**

以下代码示例演示了如何使用 `create-extension-association`。

## AWS CLI

### 创建扩展关联

以下 `create-extension-association` 示例在中创建了新的扩展关联 AWS AppConfig。

```
aws appconfig create-extension-association \  
  --region us-west-2 \  
  --extension-identifier S3-backup-extension \  
  --resource-identifier "arn:aws:appconfig:us-west-2:123456789012:application/  
Finance" \  
  --parameters S3bucket=FinanceConfigurationBackup
```



输出：

```
{
  "Id": "a1b2c3d4",
  "ExtensionArn": "arn:aws:appconfig:us-west-2:123456789012:extension/S3-backup-extension/1",
  "ResourceArn": "arn:aws:appconfig:us-west-2:123456789012:application/Finance",
  "Parameters": {
    "S3bucket": "FinanceConfigurationBackup"
  },
  "ExtensionVersionNumber": 1
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[使用 AWS AppConfig 扩展](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateExtensionAssociation](#)中的。

## create-extension

以下代码示例演示了如何使用 create-extension。

### AWS CLI

#### 创建扩展

以下 create-extension 示例在中创建了一个新的扩展 AWS AppConfig。

```
aws appconfig create-extension \
  --region us-west-2 \
  --name S3-backup-extension \
  --actions
  PRE_CREATE_HOSTED_CONFIGURATION_VERSION=[{Name=S3backup,Uri=arn:aws:lambda:us-west-2:123456789012:function:s3backupfunction,RoleArn=arn:aws:iam::123456789012:role/appconfigextensionrole}] \
  --parameters S3bucket={Required=true}
```

输出：

```
{
  "Id": "1A2B3C4D",
  "Name": "S3-backup-extension",
  "VersionNumber": 1,
```

```

    "Arn": "arn:aws:appconfig:us-west-2:123456789012:extension/1A2B3C4D/1",
    "Actions": {
      "PRE_CREATE_HOSTED_CONFIGURATION_VERSION": [
        {
          "Name": "S3backup",
          "Uri": "arn:aws:lambda:us-
west-2:123456789012:function:s3backupfunction",
          "RoleArn": "arn:aws:iam::123456789012:role/appconfigextensionrole"
        }
      ]
    },
    "Parameters": {
      "S3bucket": {
        "Required": true
      }
    }
  }
}

```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[使用 AWS AppConfig 扩展](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateExtension](#)中的。

## create-hosted-configuration-version

以下代码示例演示了如何使用 create-hosted-configuration-version。

### AWS CLI

#### 创建托管配置版本

以下 create-hosted-configuration-version 示例在 AWS AppConfig 托管配置存储中创建新配置。必须先将配置内容转换为 base64。

```

aws appconfig create-hosted-configuration-version \
  --application-id "339ohji" \
  --configuration-profile-id "ur8hx2f" \
  --content
eyAiTmFtZSI6ICJFeGFtcGx1QXBwbGljYXRpb24iLCAiSWQiOiBFFeGFtcGx1SUQsICJSYW5rIjogNyB9 \
  --content-type "application/json" \
  configuration_version_output_file

```

configuration\_version\_output\_file 的内容：

```
{ "Name": "ExampleApplication", "Id": ExampleID, "Rank": 7 }
```

输出：

```
{
  "ApplicationId": "339ohji",
  "ConfigurationProfileId": "ur8hx2f",
  "VersionNumber": "1",
  "ContentType": "application/json"
}
```

有关更多信息，请参阅 AWS AppConfig 用户指南中的关于 AWS AppConfig 托管配置[存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateHostedConfigurationVersion](#)中的。

## delete-application

以下代码示例演示了如何使用 delete-application。

### AWS CLI

删除应用程序

以下 delete-application 示例删除了指定的应用程序。

```
aws appconfig delete-application \
--application-id 339ohji
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 1：创建 AWS AppConfig 应用程序](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteApplication](#)中的。

## delete-configuration-profile

以下代码示例演示了如何使用 delete-configuration-profile。

### AWS CLI

删除配置文件

以下delete-configuration-profile示例删除了指定的配置文件。

```
aws appconfig delete-configuration-profile \  
  --application-id 339ohji \  
  --configuration-profile-id ur8hx2f
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 3：创建配置和配置配置文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteConfigurationProfile](#)中的。

## delete-deployment-strategy

以下代码示例演示了如何使用 delete-deployment-strategy。

### AWS CLI

#### 删除部署策略

以下delete-deployment-strategy示例删除了指定的部署策略。

```
aws appconfig delete-deployment-strategy \  
  --deployment-strategy-id 1225qzk
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 4：创建部署策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDeploymentStrategy](#)中的。

## delete-environment

以下代码示例演示了如何使用 delete-environment。

### AWS CLI

#### 删除环境

以下delete-environment示例删除了指定的应用程序环境。

```
aws appconfig delete-environment \  
  --application-id 339ohji \  
  --environment-id ur8hx2f
```

```
--environment-id 54j1r29
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 2：创建环境](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteEnvironment](#)中的。

## delete-extension-association

以下代码示例演示了如何使用 delete-extension-association。

### AWS CLI

删除扩展关联

以下delete-extension-association示例从中删除扩展关联 AWS AppConfig。

```
aws appconfig delete-extension-association \  
  --region us-west-2 \  
  --extension-association-id a1b2c3d4
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[使用 AWS AppConfig 扩展](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteExtensionAssociation](#)中的。

## delete-extension

以下代码示例演示了如何使用 delete-extension。

### AWS CLI

删除扩展

以下delete-extension示例从中删除一个扩展 AWS AppConfig。

```
aws appconfig delete-extension \  
  --region us-west-2 \  
  --extension-identifier S3-backup-extension
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[使用 AWS AppConfig 扩展](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteExtension](#)中的。

## delete-hosted-configuration-version

以下代码示例演示了如何使用 delete-hosted-configuration-version。

### AWS CLI

删除托管配置版本

以下delete-hosted-configuration-version示例删除托管在托 AWS AppConfig 管配置存储中的配置版本。

```
aws appconfig delete-hosted-configuration-version \  
  --application-id 339ohji \  
  --configuration-profile-id ur8hx2f \  
  --version-number 1
```

输出:: 此命令不产生任何输出。

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 3：创建配置和配置配置文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteHostedConfigurationVersion](#)中的。

## get-application

以下代码示例演示了如何使用 get-application。

### AWS CLI

列出应用程序的详细信息

以下get-application示例列出了指定应用程序的详细信息。

```
aws appconfig get-application \  
  --application-id 339ohji
```

输出：

```
{  
  "Description": "An application used for creating an example.",
```

```
"Id": "339ohji",  
"Name": "example-application"  
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的 [AWS AppConfig 工作原理](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetApplication](#) 中的。

## get-configuration-profile

以下代码示例演示了如何使用 get-configuration-profile。

### AWS CLI

检索配置文件详细信息

以下 get-configuration-profile 示例返回指定配置文件的详细信息。

```
aws appconfig get-configuration-profile \  
  --application-id 339ohji \  
  --configuration-profile-id ur8hx2f
```

输出：

```
{  
  "ApplicationId": "339ohji",  
  "Id": "ur8hx2f",  
  "Name": "Example-Configuration-Profile",  
  "LocationUri": "ssm-parameter://Example-Parameter",  
  "RetrievalRoleArn": "arn:aws:iam::111122223333:role/Example-App-Config-Role"  
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的 [步骤 3：创建配置和配置配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetConfigurationProfile](#) 中的。

## get-configuration

以下代码示例演示了如何使用 get-configuration。

### AWS CLI

检索配置详细信息

以下`get-configuration`示例返回示例应用程序的配置详细信息。在后续调用 `get-configuration` 时，使用`client-configuration-version`参数仅在版本已更改时更新应用程序的配置。只有在版本更改时才更新配置可以避免调用 `get-configuration` 所产生的额外费用。

```
aws appconfig get-configuration \  
  --application "example-application" \  
  --environment "Example-Environment" \  
  --configuration "Example-Configuration-Profile" \  
  --client-id "test-id" \  
  configuration-output-file
```

`configuration-output-file` 的内容：

```
{ "Name": "ExampleApplication", "Id": ExampleID, "Rank": 7 }
```

输出：

```
{  
  "ConfigurationVersion": "1",  
  "ContentType": "application/json"  
}
```

有关更多信息，请参阅 [《AWS AppConfig 用户指南》中的步骤 6：接收配置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetConfiguration](#)中的。

## get-deployment-strategy

以下代码示例演示了如何使用 `get-deployment-strategy`。

AWS CLI

检索部署策略的详细信息

以下`get-deployment-strategy`示例列出了指定部署策略的详细信息。

```
aws appconfig get-deployment-strategy \  
  --deployment-strategy-id 1225qzk
```

输出：



```
{
  "Id": "1225qzk",
  "Name": "Example-Deployment",
  "DeploymentDurationInMinutes": 15,
  "GrowthType": "LINEAR",
  "GrowthFactor": 25.0,
  "FinalBakeTimeInMinutes": 0,
  "ReplicateTo": "SSM_DOCUMENT"
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 4：创建部署策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetDeploymentStrategy](#)中的。

## get-deployment

以下代码示例演示了如何使用 get-deployment。

### AWS CLI

检索部署详细信息

以下 get-deployment 示例列出了在指定环境中部署到应用程序和部署的详细信息。

```
aws appconfig get-deployment \
  --application-id 339ohji \
  --environment-id 54j1r29 \
  --deployment-number 1
```

输出：

```
{
  "ApplicationId": "339ohji",
  "EnvironmentId": "54j1r29",
  "DeploymentStrategyId": "1225qzk",
  "ConfigurationProfileId": "ur8hx2f",
  "DeploymentNumber": 1,
  "ConfigurationName": "Example-Configuration-Profile",
  "ConfigurationLocationUri": "ssm-parameter://Example-Parameter",
  "ConfigurationVersion": "1",
  "DeploymentDurationInMinutes": 15,
  "GrowthType": "LINEAR",
  "GrowthFactor": 25.0,
```

```
"FinalBakeTimeInMinutes": 0,
"State": "COMPLETE",
"EventLog": [
  {
    "EventType": "DEPLOYMENT_COMPLETED",
    "TriggeredBy": "APPCONFIG",
    "Description": "Deployment completed",
    "OccurredAt": "2021-09-17T21:59:03.888000+00:00"
  },
  {
    "EventType": "BAKE_TIME_STARTED",
    "TriggeredBy": "APPCONFIG",
    "Description": "Deployment bake time started",
    "OccurredAt": "2021-09-17T21:58:57.722000+00:00"
  },
  {
    "EventType": "PERCENTAGE_UPDATED",
    "TriggeredBy": "APPCONFIG",
    "Description": "Configuration available to 100.00% of clients",
    "OccurredAt": "2021-09-17T21:55:56.816000+00:00"
  },
  {
    "EventType": "PERCENTAGE_UPDATED",
    "TriggeredBy": "APPCONFIG",
    "Description": "Configuration available to 75.00% of clients",
    "OccurredAt": "2021-09-17T21:52:56.567000+00:00"
  },
  {
    "EventType": "PERCENTAGE_UPDATED",
    "TriggeredBy": "APPCONFIG",
    "Description": "Configuration available to 50.00% of clients",
    "OccurredAt": "2021-09-17T21:49:55.737000+00:00"
  },
  {
    "EventType": "PERCENTAGE_UPDATED",
    "TriggeredBy": "APPCONFIG",
    "Description": "Configuration available to 25.00% of clients",
    "OccurredAt": "2021-09-17T21:46:55.187000+00:00"
  },
  {
    "EventType": "DEPLOYMENT_STARTED",
    "TriggeredBy": "USER",
    "Description": "Deployment started",
    "OccurredAt": "2021-09-17T21:43:54.205000+00:00"
  }
]
```

```
    }
  ],
  "PercentageComplete": 100.0,
  "StartedAt": "2021-09-17T21:43:54.205000+00:00",
  "CompletedAt": "2021-09-17T21:59:03.888000+00:00"
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 5：部署配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetDeployment](#)中的。

## get-environment

以下代码示例演示了如何使用 get-environment。

### AWS CLI

检索环境详细信息

以下 get-environment 示例返回指定环境的详细信息和状态。

```
aws appconfig get-environment \
  --application-id 339ohji \
  --environment-id 54j1r29
```

输出：

```
{
  "ApplicationId": "339ohji",
  "Id": "54j1r29",
  "Name": "Example-Environment",
  "State": "ReadyForDeployment"
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 2：创建环境](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetEnvironment](#)中的。

## get-extension-association

以下代码示例演示了如何使用 get-extension-association。

## AWS CLI

要获取扩展关联的详细信息

以下`get-extension-association`示例显示了有关扩展关联的信息。

```
aws appconfig get-extension-association \  
  --region us-west-2 \  
  --extension-association-id a1b2c3d4
```

输出：

```
{  
  "Id": "a1b2c3d4",  
  "ExtensionArn": "arn:aws:appconfig:us-west-2:123456789012:extension/S3-backup-  
extension/1",  
  "ResourceArn": "arn:aws:appconfig:us-west-2:123456789012:application/Finance",  
  "Parameters": {  
    "S3bucket": "FinanceConfigurationBackup"  
  },  
  "ExtensionVersionNumber": 1  
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[使用 AWS AppConfig 扩展](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetExtensionAssociation](#)中的。

## get-extension

以下代码示例演示了如何使用 `get-extension`。

## AWS CLI

获取分机详情

以下`get-extension`示例显示了有关扩展程序的信息。

```
aws appconfig get-extension \  
  --region us-west-2 \  
  --extension-identifier S3-backup-extension
```

输出：

```
{
  "Id": "1A2B3C4D",
  "Name": "S3-backup-extension",
  "VersionNumber": 1,
  "Arn": "arn:aws:appconfig:us-west-2:123456789012:extension/S3-backup-extension/1",
  "Actions": {
    "PRE_CREATE_HOSTED_CONFIGURATION_VERSION": [
      {
        "Name": "S3backup",
        "Uri": "arn:aws:lambda:us-west-2:123456789012:function:S3backupfunction",
        "RoleArn": "arn:aws:iam::123456789012:role/appconfigextensionrole"
      }
    ]
  },
  "Parameters": {
    "S3bucket": {
      "Required": true
    }
  }
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[使用 AWS AppConfig 扩展](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetExtension](#)中的。

## get-hosted-configuration-version

以下代码示例演示了如何使用 `get-hosted-configuration-version`。

### AWS CLI

检索托管配置的详细信息

以下 `get-hosted-configuration-version` 示例检索 AWS AppConfig 托管配置的配置详细信息。

```
aws appconfig get-hosted-configuration-version \
  --application-id 339ohji \
  --configuration-profile-id ur8hx2f \
  --version-number 1 \
```

```
hosted-configuration-version-output
```

hosted-configuration-version-output 的内容：

```
{ "Name": "ExampleApplication", "Id": ExampleID, "Rank": 7 }
```

输出：

```
{
  "ApplicationId": "339ohji",
  "ConfigurationProfileId": "ur8hx2f",
  "VersionNumber": "1",
  "ContentType": "application/json"
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[关于 AWS AppConfig 托管配置存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetHostedConfigurationVersion](#)中的。

## list-applications

以下代码示例演示了如何使用 list-applications。

### AWS CLI

列出可用的应用程序

以下 list-applications 示例列出了您 AWS 账户中的可用应用程序。

```
aws appconfig list-applications
```

输出：

```
{
  "Items": [
    {
      "Id": "339ohji",
      "Name": "test-application",
      "Description": "An application used for creating an example."
    },
  ],
}
```

```
{
  "Id": "rwalwu7",
  "Name": "Test-Application"
}
]
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 1：创建 AWS AppConfig 应用程序](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListApplications](#)中的。

## list-configuration-profiles

以下代码示例演示了如何使用 list-configuration-profiles。

### AWS CLI

列出可用的配置文件

以下 list-configuration-profiles 示例列出了指定应用程序的可用配置文件。

```
aws appconfig list-configuration-profiles \
  --application-id 339ohji
```

输出：

```
{
  "Items": [
    {
      "ApplicationId": "339ohji",
      "Id": "ur8hx2f",
      "Name": "Example-Configuration-Profile",
      "LocationUri": "ssm-parameter://Example-Parameter"
    }
  ]
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 3：创建配置和配置配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListConfigurationProfiles](#)中的。

## list-deployment-strategies

以下代码示例演示了如何使用 `list-deployment-strategies`。

### AWS CLI

列出可用的部署策略

以下 `list-deployment-strategies` 示例列出了您的 AWS 账户中可用的部署策略。

```
aws appconfig list-deployment-strategies
```

输出：

```
{
  "Items": [
    {
      "Id": "1225qzk",
      "Name": "Example-Deployment",
      "DeploymentDurationInMinutes": 15,
      "GrowthType": "LINEAR",
      "GrowthFactor": 25.0,
      "FinalBakeTimeInMinutes": 0,
      "ReplicateTo": "SSM_DOCUMENT"
    },
    {
      "Id": "AppConfig.AllAtOnce",
      "Name": "AppConfig.AllAtOnce",
      "Description": "Quick",
      "DeploymentDurationInMinutes": 0,
      "GrowthType": "LINEAR",
      "GrowthFactor": 100.0,
      "FinalBakeTimeInMinutes": 10,
      "ReplicateTo": "NONE"
    },
    {
      "Id": "AppConfig.Linear50PercentEvery30Seconds",
      "Name": "AppConfig.Linear50PercentEvery30Seconds",
      "Description": "Test/Demo",
      "DeploymentDurationInMinutes": 1,
      "GrowthType": "LINEAR",
      "GrowthFactor": 50.0,
      "FinalBakeTimeInMinutes": 1,

```



```
        "ReplicateTo": "NONE"
    },
    {
        "Id": "AppConfig.Canary10Percent20Minutes",
        "Name": "AppConfig.Canary10Percent20Minutes",
        "Description": "AWS Recommended",
        "DeploymentDurationInMinutes": 20,
        "GrowthType": "EXPONENTIAL",
        "GrowthFactor": 10.0,
        "FinalBakeTimeInMinutes": 10,
        "ReplicateTo": "NONE"
    }
]
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 4：创建部署策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListDeploymentStrategies](#)中的。

## list-deployments

以下代码示例演示了如何使用 list-deployments。

### AWS CLI

#### 列出可用部署

以下 list-deployments 示例列出了您的 AWS 账户中针对指定应用程序和环境的可用部署。

```
aws appconfig list-deployments \
  --application-id 339ohji \
  --environment-id 54j1r29
```

#### 输出：

```
{
  "Items": [
    {
      "DeploymentNumber": 1,
      "ConfigurationName": "Example-Configuration-Profile",
      "ConfigurationVersion": "1",
      "DeploymentDurationInMinutes": 15,
      "GrowthType": "LINEAR",
```

```
        "GrowthFactor": 25.0,  
        "FinalBakeTimeInMinutes": 0,  
        "State": "COMPLETE",  
        "PercentageComplete": 100.0,  
        "StartedAt": "2021-09-17T21:43:54.205000+00:00",  
        "CompletedAt": "2021-09-17T21:59:03.888000+00:00"  
    }  
]  
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 5：部署配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListDeployments](#) 中的。

## list-environments

以下代码示例演示了如何使用 list-environments。

### AWS CLI

列出可用环境

以下 list-environments 示例列出了您 AWS 账户中指定应用程序的可用环境。

```
aws appconfig list-environments \  
  --application-id 339ohji
```

输出：

```
{  
  "Items": [  
    {  
      "ApplicationId": "339ohji",  
      "Id": "54j1r29",  
      "Name": "Example-Environment",  
      "State": "ReadyForDeployment"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 2：创建环境](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListEnvironments](#) 中的。

## list-extension-associations

以下代码示例演示了如何使用 list-extension-associations。

### AWS CLI

列出您 AWS 账户中某个 AWS 地区的所有 AWS AppConfig 分机关联

以下list-extension-associations示例列出了特定 AWS 区域中当前 AWS 账户的所有 AWS AppConfig 分机关联。

```
aws appconfig list-extension-associations \  
  --region us-west-2
```

输出：

```
{  
  "Items": [  
    {  
      "Id": "a1b2c3d4",  
      "ExtensionArn": "arn:aws:appconfig:us-west-2:123456789012:extension/S3-  
backup-extension/1",  
      "ResourceArn": "arn:aws:appconfig:us-west-2:123456789012:application/  
Finance"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[使用 AWS AppConfig 扩展](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListExtensionAssociations](#)中的。

## list-extensions

以下代码示例演示了如何使用 list-extensions。

### AWS CLI

列出您 AWS 账户中某个 AWS 地区的所有 AWS AppConfig 扩展程序

以下list-extensions示例列出了特定 AWS 区域中当前 AWS 账户的所有 AWS AppConfig 扩展名。该命令返回自定义和 AWS 创作的扩展。

```
aws appconfig list-extensions \  
  --region us-west-2
```

输出：

```
{  
  "Items": [  
    {  
      "Id": "1A2B3C4D",  
      "Name": "S3-backup-extension",  
      "VersionNumber": 1,  
      "Arn": "arn:aws:appconfig:us-west-2:123456789012:extension/1A2B3C4D/1"  
    },  
    {  
      "Id": "AWS.AppConfig.FeatureFlags",  
      "Name": "AppConfig Feature Flags Helper",  
      "VersionNumber": 1,  
      "Arn": "arn:aws:appconfig:us-west-2::extension/  
AWS.AppConfig.FeatureFlags/1",  
      "Description": "Validates AppConfig feature flag data automatically  
against a JSON schema that includes structure and constraints. Also transforms  
feature flag data prior to sending to the client. This extension is automatically  
associated to configuration profiles with type \"AWS.AppConfig.FeatureFlags\"."  
    },  
    {  
      "Id": "AWS.AppConfig.JiraIntegration",  
      "Name": "AppConfig integration with Atlassian Jira",  
      "VersionNumber": 1,  
      "Arn": "arn:aws:appconfig:us-west-2::extension/  
AWS.AppConfig.JiraIntegration/1",  
      "Description": "Exports feature flag data from AWS AppConfig into  
Jira. The lifecycle of each feature flag in AppConfig is tracked in Jira as an  
individual issue. Customers can see in Jira when flags are updated, turned on or  
off. Works in conjunction with the AppConfig app in the Atlassian Marketplace and  
is automatically associated to configuration profiles configured within that app."  
    },  
    {  
      "Id": "AWS.AppConfig.DeploymentNotificationsToEventBridge",  
      "Name": "AppConfig deployment events to Amazon EventBridge",  
      "VersionNumber": 1,  
      "Arn": "arn:aws:appconfig:us-west-2::extension/  
AWS.AppConfig.DeploymentNotificationsToEventBridge/1",
```

```

        "Description": "Sends events to Amazon EventBridge when a deployment
of configuration data in AppConfig is started, completed, or rolled back. Can
be associated to the following resources in AppConfig: Application, Environment,
Configuration Profile."
    },
    {
        "Id": "AWS.AppConfig.DeploymentNotificationsToSqs",
        "Name": "AppConfig deployment events to Amazon SQS",
        "VersionNumber": 1,
        "Arn": "arn:aws:appconfig:us-west-2::extension/
AWS.AppConfig.DeploymentNotificationsToSqs/1",
        "Description": "Sends messages to the configured Amazon SQS queue when
a deployment of configuration data in AppConfig is started, completed, or rolled
back. Can be associated to the following resources in AppConfig: Application,
Environment, Configuration Profile."
    },
    {
        "Id": "AWS.AppConfig.DeploymentNotificationsToSns",
        "Name": "AppConfig deployment events to Amazon SNS",
        "VersionNumber": 1,
        "Description": "Sends events to the configured Amazon SNS topic when
a deployment of configuration data in AppConfig is started, completed, or rolled
back. Can be associated to the following resources in AppConfig: Application,
Environment, Configuration Profile."
    }
]
}

```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[使用 AWS AppConfig 扩展](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListExtensions](#)中的。

## list-hosted-configuration-versions

以下代码示例演示了如何使用 list-hosted-configuration-versions。

### AWS CLI

列出可用的托管配置版本

以下 list-hosted-configuration-versions 示例列出了托管在托 AWS AppConfig 管配置存储中针对指定应用程序和配置文件的配置版本。

```
aws appconfig list-hosted-configuration-versions \  
  --application-id 339ohji \  
  --configuration-profile-id ur8hx2f
```

输出：

```
{  
  "Items": [  
    {  
      "ApplicationId": "339ohji",  
      "ConfigurationProfileId": "ur8hx2f",  
      "VersionNumber": 1,  
      "ContentType": "application/json"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[关于 AWS AppConfig 托管配置存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListHostedConfigurationVersions](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

AWS CLI

列出应用程序的标签

以下 `list-tags-for-resource` 示例列出了指定应用程序的标签。

```
aws appconfig list-tags-for-resource \  
  --resource-arn arn:aws:appconfig:us-east-1:682428703967:application/339ohji
```

输出：

```
{  
  "Tags": {  
    "group1": "1"  
  }  
}
```

```
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 1：创建 AWS AppConfig 应用程序](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListTagsForResource](#)中的。

## start-deployment

以下代码示例演示了如何使用 start-deployment。

### AWS CLI

#### 开始配置部署

以下 start-deployment 示例使用指定的环境、部署策略和配置文件开始部署到应用程序。

```
aws appconfig start-deployment \  
  --application-id 339ohji \  
  --environment-id 54j1r29 \  
  --deployment-strategy-id 1225qzk \  
  --configuration-profile-id ur8hx2f \  
  --configuration-version 1
```

输出：

```
{  
  "ApplicationId": "339ohji",  
  "EnvironmentId": "54j1r29",  
  "DeploymentStrategyId": "1225qzk",  
  "ConfigurationProfileId": "ur8hx2f",  
  "DeploymentNumber": 1,  
  "ConfigurationName": "Example-Configuration-Profile",  
  "ConfigurationLocationUri": "ssm-parameter://Example-Parameter",  
  "ConfigurationVersion": "1",  
  "DeploymentDurationInMinutes": 15,  
  "GrowthType": "LINEAR",  
  "GrowthFactor": 25.0,  
  "FinalBakeTimeInMinutes": 0,  
  "State": "DEPLOYING",  
  "EventLog": [  
    {
```

```
        "EventType": "DEPLOYMENT_STARTED",
        "TriggeredBy": "USER",
        "Description": "Deployment started",
        "OccurredAt": "2021-09-17T21:43:54.205000+00:00"
    }
],
"PercentageComplete": 0.0,
"StartedAt": "2021-09-17T21:43:54.205000+00:00"
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 5：部署配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[StartDeployment](#)中的。

## stop-deployment

以下代码示例演示了如何使用 stop-deployment。

### AWS CLI

#### 停止配置部署

以下 stop-deployment 示例停止将应用程序配置部署到指定环境。

```
aws appconfig stop-deployment \
  --application-id 339ohji \
  --environment-id 54j1r29 \
  --deployment-number 2
```

输出：

```
{
  "DeploymentNumber": 0,
  "DeploymentDurationInMinutes": 0,
  "GrowthFactor": 0.0,
  "FinalBakeTimeInMinutes": 0,
  "PercentageComplete": 0.0
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 5：部署配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[StopDeployment](#)中的。



## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为应用程序添加标签

以下tag-resource示例为应用程序资源添加了标签。

```
aws appconfig tag-resource \  
  --resource-arn arn:aws:appconfig:us-east-1:682428703967:application/339ohji \  
  --tags '{"group1" : "1"}'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 1：创建 AWS AppConfig 应用程序](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从应用程序中移除标签

以下untag-resource示例从指定的应用程序中删除 group1 标签。

```
aws appconfig untag-resource \  
  --resource-arn arn:aws:appconfig:us-east-1:111122223333:application/339ohji \  
  --tag-keys '["group1"]'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 1：创建 AWS AppConfig 应用程序](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-application

以下代码示例演示了如何使用 update-application。

### AWS CLI

#### 更新应用程序

以下update-application示例更新了指定应用程序的名称。

```
aws appconfig update-application \  
  --application-id 339ohji \  
  --name "Example-Application"
```

输出：

```
{  
  "Id": "339ohji",  
  "Name": "Example-Application",  
  "Description": "An application used for creating an example."  
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 1：创建 AWS AppConfig 应用程序](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateApplication](#)中的。

## update-configuration-profile

以下代码示例演示了如何使用 update-configuration-profile。

### AWS CLI

#### 更新配置文件

以下update-configuration-profile示例更新了指定配置文件的描述。

```
aws appconfig update-configuration-profile \  
  --application-id 339ohji \  
  --configuration-profile-id ur8hx2f \  
  --description "Configuration profile used for examples."
```

输出：

```
{
  "ApplicationId": "339ohji",
  "Id": "ur8hx2f",
  "Name": "Example-Configuration-Profile",
  "Description": "Configuration profile used for examples.",
  "LocationUri": "ssm-parameter://Example-Parameter",
  "RetrievalRoleArn": "arn:aws:iam::111122223333:role/Example-App-Config-Role"
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 3：创建配置和配置配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateConfigurationProfile](#) 中的。

## update-deployment-strategy

以下代码示例演示了如何使用 update-deployment-strategy。

### AWS CLI

#### 更新部署策略

以下 update-deployment-strategy 示例将指定部署策略中的最终烘焙时间更新为 20 分钟。

```
aws appconfig update-deployment-strategy \
  --deployment-strategy-id 1225qzk \
  --final-bake-time-in-minutes 20
```

输出：

```
{
  "Id": "1225qzk",
  "Name": "Example-Deployment",
  "DeploymentDurationInMinutes": 15,
  "GrowthType": "LINEAR",
  "GrowthFactor": 25.0,
  "FinalBakeTimeInMinutes": 20,
  "ReplicateTo": "SSM_DOCUMENT"
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 4：创建部署策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDeploymentStrategy](#)中的。

## update-environment

以下代码示例演示了如何使用 update-environment。

### AWS CLI

#### 更新环境

以下update-environment示例更新了环境的描述。

```
aws appconfig update-environment \  
  --application-id 339ohji \  
  --environment-id 54j1r29 \  
  --description "An environment for examples."
```

输出：

```
{  
  "ApplicationId": "339ohji",  
  "Id": "54j1r29",  
  "Name": "Example-Environment",  
  "Description": "An environment for examples.",  
  "State": "RolledBack"  
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 2：创建环境](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateEnvironment](#)中的。

## update-extension-association

以下代码示例演示了如何使用 update-extension-association。

### AWS CLI

#### 更新 AWS AppConfig 扩展关联

以下update-extension-association示例向中的扩展关联添加了一个新的参数值 AWS AppConfig。

```
aws appconfig update-extension-association \  
  --region us-west-2 \  
  --extension-association-id a1b2c3d4 \  
  --parameters S3bucket=FinanceMobileApp
```

输出：

```
{  
  "Id": "a1b2c3d4",  
  "ExtensionArn": "arn:aws:appconfig:us-west-2:123456789012:extension/S3-backup-  
extension/1",  
  "ResourceArn": "arn:aws:appconfig:us-west-2:123456789012:application/Finance",  
  "Parameters": {  
    "S3bucket": "FinanceMobileApp"  
  },  
  "ExtensionVersionNumber": 1  
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[使用 AWS AppConfig 扩展](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateExtensionAssociation](#)中的。

## update-extension

以下代码示例演示了如何使用 update-extension。

### AWS CLI

更新 AWS AppConfig 扩展程序

以下 update-extension 示例向中的扩展添加了附加参数 Key AWS AppConfig。

```
aws appconfig update-extension \  
  --region us-west-2 \  
  --extension-identifier S3-backup-extension \  
  --parameters S3bucket={Required=true},CampaignID={Required=false}
```

输出：

```
{
```

```
"Id": "1A2B3C4D",
>Name": "S3-backup-extension",
>VersionNumber": 1,
>Arn": "arn:aws:appconfig:us-west-2:123456789012:extension/1A2B3C4D/1",
>Actions": {
>  "PRE_CREATE_HOSTED_CONFIGURATION_VERSION": [
>    {
>      "Name": "S3backup",
>      "Uri": "arn:aws:lambda:us-
west-2:123456789012:function:S3backupfunction",
>      "RoleArn": "arn:aws:iam::123456789012:role/appconfigextensionrole"
>    }
>  ]
},
>Parameters": {
>  "CampaignID": {
>    "Required": false
>  },
>  "S3bucket": {
>    "Required": true
>  }
> }
}
```

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[使用 AWS AppConfig 扩展](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateExtension](#) 中的。

## validate-configuration

以下代码示例演示了如何使用 validate-configuration。

### AWS CLI

#### 验证配置

以下 validate-configuration 示例使用配置文件中的验证器来验证配置。

```
aws appconfig validate-configuration \
  --application-id abc1234 \
  --configuration-profile-id ur8hx2f \
  --configuration-version 1
```

该命令不产生任何输出。

有关更多信息，请参阅《AWS AppConfig 用户指南》中的[步骤 3：创建配置和配置配置文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ValidateConfiguration](#)中的。

## Application Auto Scaling 示例 AWS CLI

以下代码示例向您展示了如何使用与 Application Auto Scaling AWS Command Line Interface 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **delete-scaling-policy**

以下代码示例演示了如何使用 delete-scaling-policy。

AWS CLI

删除扩展策略

此示例删除了在默认集群中运行的 Amazon ECS 服务 Web 应用程序的扩展策略。

命令:

```
aws application-autoscaling delete-scaling-policy --policy-name web-app-cpu-1t-25 --scalable-dimension ecs:service:DesiredCount --resource-id service/default/web-app --service-namespace ecs
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteScalingPolicy](#)中的。

## delete-scheduled-action

以下代码示例演示了如何使用 delete-scheduled-action。

### AWS CLI

#### 删除计划操作

以下delete-scheduled-action示例从指定的 Amazon AppStream 2.0 队列中删除指定的计划操作：

```
aws application-autoscaling delete-scheduled-action \  
  --service-namespace appstream \  
  --scalable-dimension appstream:fleet:DesiredCapacity \  
  --resource-id fleet/sample-fleet \  
  --scheduled-action-name my-recurring-action
```

此命令不生成任何输出。

有关更多信息，请参阅《Application Auto Scaling 用户指南》中的[计划扩展](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteScheduledAction](#)中的。

## deregister-scalable-target

以下代码示例演示了如何使用 deregister-scalable-target。

### AWS CLI

#### 取消注册可扩展目标

此示例取消注册在默认集群中运行的名为 Web 应用程序的 Amazon ECS 服务的可扩展目标。

命令:

```
aws application-autoscaling deregister-scalable-target --service-namespace ecs --  
scalable-dimension ecs:service:DesiredCount --resource-id service/default/web-app
```

此示例取消注册自定义资源的可扩展目标。 custom-resource-id.txt 文件包含一个用于标识资源 ID 的字符串，对于自定义资源，该字符串是通过您的 Amazon API Gateway 终端节点指向自定义资源的路径。



**命令:**

```
aws application-autoscaling deregister-scalable-target --service-namespace custom-resource --scalable-dimension custom-resource:ResourceType:Property --resource-id file:///~/custom-resource-id.txt
```

custom-resource-id.txt 文件的内容 :

```
https://example.execute-api.us-west-2.amazonaws.com/prod/scalableTargetDimensions/1-23456789
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterScalableTarget](#)中的。

**describe-scalable-targets**

以下代码示例演示了如何使用 describe-scalable-targets。

**AWS CLI**

描述可扩展的目标

以下describe-scalable-targets示例描述了ecs服务命名空间的可扩展目标。

```
aws application-autoscaling describe-scalable-targets \
  --service-namespace ecs
```

输出 :

```
{
  "ScalableTargets": [
    {
      "ServiceNamespace": "ecs",
      "ScalableDimension": "ecs:service:DesiredCount",
      "ResourceId": "service/default/web-app",
      "MinCapacity": 1,
      "MaxCapacity": 10,
      "RoleARN": "arn:aws:iam::123456789012:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService",
      "CreationTime": 1462558906.199,
    }
  ]
}
```

```
    "SuspendedState": {
      "DynamicScalingOutSuspended": false,
      "ScheduledScalingSuspended": false,
      "DynamicScalingInSuspended": false
    },
    "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
  }
]
}
```

有关更多信息，请参阅《[应用程序自动缩放用户指南](#)》中的[可用于 Application Auto Scaling 的 AWS 服务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeScalableTargets](#) 中的。

## describe-scaling-activities

以下代码示例演示了如何使用 describe-scaling-activities。

### AWS CLI

示例 1：描述指定 Amazon ECS 服务的扩展活动

以下 describe-scaling-activities 示例描述了 default 集群中运行的名为 web-app Amazon ECS 服务的扩展活动。输出显示了由扩展策略启动的扩展活动。

```
aws application-autoscaling describe-scaling-activities \
  --service-namespace ecs \
  --resource-id service/default/web-app
```

输出：

```
{
  "ScalingActivities": [
    {
      "ScalableDimension": "ecs:service:DesiredCount",
      "Description": "Setting desired count to 1.",
      "ResourceId": "service/default/web-app",
      "ActivityId": "e6c5f7d1-dbbb-4a3f-89b2-51f33e766399",
      "StartTime": 1462575838.171,
```

```

        "ServiceNamespace": "ecs",
        "EndTime": 1462575872.111,
        "Cause": "monitor alarm web-app-cpu-1t-25 in state ALARM triggered
policy web-app-cpu-1t-25",
        "StatusMessage": "Successfully set desired count to 1. Change
successfully fulfilled by ecs.",
        "StatusCode": "Successful"
    }
]
}

```

有关更多信息，请参阅《Auto [Scaling Auto Scaling 用户指南](#)》中的应用程序 [Auto Scaling 的扩展活动](#)。

### 示例 2：描述指定 DynamoDB 表的扩展活动

以下 describe-scaling-activities 示例描述了名为的 DynamoDB 表的扩展活动。TestTable 输出显示了由两个不同的计划操作启动的扩展活动。

```

aws application-autoscaling describe-scaling-activities \
  --service-namespace dynamodb \
  --resource-id table/TestTable

```

输出：

```

{
  "ScalingActivities": [
    {
      "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
      "Description": "Setting write capacity units to 10.",
      "ResourceId": "table/my-table",
      "ActivityId": "4d1308c0-bbcf-4514-a673-b0220ae38547",
      "StartTime": 1561574415.086,
      "ServiceNamespace": "dynamodb",
      "EndTime": 1561574449.51,
      "Cause": "maximum capacity was set to 10",
      "StatusMessage": "Successfully set write capacity units to 10. Change
successfully fulfilled by dynamodb.",
      "StatusCode": "Successful"
    },
    {
      "ScalableDimension": "dynamodb:table:WriteCapacityUnits",

```

```

    "Description": "Setting min capacity to 5 and max capacity to 10",
    "ResourceId": "table/my-table",
    "ActivityId": "f2b7847b-721d-4e01-8ef0-0c8d3bacc1c7",
    "StartTime": 1561574414.644,
    "ServiceNamespace": "dynamodb",
    "Cause": "scheduled action name my-second-scheduled-action was
triggered",
    "StatusMessage": "Successfully set min capacity to 5 and max capacity to
10",
    "StatusCode": "Successful"
  },
  {
    "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
    "Description": "Setting write capacity units to 15.",
    "ResourceId": "table/my-table",
    "ActivityId": "d8ea4de6-9eaa-499f-b466-2cc5e681ba8b",
    "StartTime": 1561574108.904,
    "ServiceNamespace": "dynamodb",
    "EndTime": 1561574140.255,
    "Cause": "minimum capacity was set to 15",
    "StatusMessage": "Successfully set write capacity units to 15. Change
successfully fulfilled by dynamodb.",
    "StatusCode": "Successful"
  },
  {
    "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
    "Description": "Setting min capacity to 15 and max capacity to 20",
    "ResourceId": "table/my-table",
    "ActivityId": "3250fd06-6940-4e8e-bb1f-d494db7554d2",
    "StartTime": 1561574108.512,
    "ServiceNamespace": "dynamodb",
    "Cause": "scheduled action name my-first-scheduled-action was
triggered",
    "StatusMessage": "Successfully set min capacity to 15 and max capacity
to 20",
    "StatusCode": "Successful"
  }
]
}

```

有关更多信息，请参阅《Auto [Scaling Auto Scaling 用户指南](#)》中的应用程序 [Auto Scaling 的扩展活动](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeScalingActivities](#) 中的。

## describe-scaling-policies

以下代码示例演示了如何使用 describe-scaling-policies。

### AWS CLI

#### 描述扩展策略

此示例命令描述了 ecs 服务命名空间的扩展策略。

命令:

```
aws application-autoscaling describe-scaling-policies --service-namespace ecs
```

输出:

```
{
  "ScalingPolicies": [
    {
      "PolicyName": "web-app-cpu-gt-75",
      "ScalableDimension": "ecs:service:DesiredCount",
      "ResourceId": "service/default/web-app",
      "CreationTime": 1462561899.23,
      "StepScalingPolicyConfiguration": {
        "Cooldown": 60,
        "StepAdjustments": [
          {
            "ScalingAdjustment": 200,
            "MetricIntervalLowerBound": 0.0
          }
        ],
        "AdjustmentType": "PercentChangeInCapacity"
      },
      "PolicyARN": "arn:aws:autoscaling:us-west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/ecs/service/default/web-app:policyName/web-app-cpu-gt-75",
      "PolicyType": "StepScaling",
      "Alarms": [
        {
          "AlarmName": "web-app-cpu-gt-75",
          "AlarmARN": "arn:aws:cloudwatch:us-west-2:012345678910:alarm:web-app-cpu-gt-75"
        }
      ]
    }
  ]
}
```

```

    ],
    "ServiceNamespace": "ecs"
  },
  {
    "PolicyName": "web-app-cpu-lt-25",
    "ScalableDimension": "ecs:service:DesiredCount",
    "ResourceId": "service/default/web-app",
    "CreationTime": 1462562575.099,
    "StepScalingPolicyConfiguration": {
      "Cooldown": 1,
      "StepAdjustments": [
        {
          "ScalingAdjustment": -50,
          "MetricIntervalUpperBound": 0.0
        }
      ],
      "AdjustmentType": "PercentChangeInCapacity"
    },
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/ecs/
service/default/web-app:policyName/web-app-cpu-lt-25",
    "PolicyType": "StepScaling",
    "Alarms": [
      {
        "AlarmName": "web-app-cpu-lt-25",
        "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:web-app-cpu-lt-25"
      }
    ],
    "ServiceNamespace": "ecs"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeScalingPolicies](#)中的。

## describe-scheduled-actions

以下代码示例演示了如何使用 describe-scheduled-actions。

### AWS CLI

#### 描述计划操作

以下describe-scheduled-actions示例显示了指定服务命名空间的计划操作的详细信息：

```
aws application-autoscaling describe-scheduled-actions \  
  --service-namespace dynamodb
```

输出：

```
{  
  "ScheduledActions": [  
    {  
      "ScalableDimension": "dynamodb:table:WriteCapacityUnits",  
      "Schedule": "at(2019-05-20T18:35:00)",  
      "ResourceId": "table/my-table",  
      "CreationTime": 1561571888.361,  
      "ScheduledActionARN": "arn:aws:autoscaling:us-  
west-2:123456789012:scheduledAction:2d36aa3b-cdf9-4565-b290-81db519b227d:resource/  
dynamodb/table/my-table:scheduledActionName/my-first-scheduled-action",  
      "ScalableTargetAction": {  
        "MinCapacity": 15,  
        "MaxCapacity": 20  
      },  
      "ScheduledActionName": "my-first-scheduled-action",  
      "ServiceNamespace": "dynamodb"  
    },  
    {  
      "ScalableDimension": "dynamodb:table:WriteCapacityUnits",  
      "Schedule": "at(2019-05-20T18:40:00)",  
      "ResourceId": "table/my-table",  
      "CreationTime": 1561571946.021,  
      "ScheduledActionARN": "arn:aws:autoscaling:us-  
west-2:123456789012:scheduledAction:2d36aa3b-cdf9-4565-b290-81db519b227d:resource/  
dynamodb/table/my-table:scheduledActionName/my-second-scheduled-action",  
      "ScalableTargetAction": {  
        "MinCapacity": 5,  
        "MaxCapacity": 10  
      },  
      "ScheduledActionName": "my-second-scheduled-action",  
      "ServiceNamespace": "dynamodb"  
    }  
  ]  
}
```

有关更多信息，请参阅《Application Auto Scaling 用户指南》中的[计划扩展](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeScheduledActions](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出可扩展目标的标签

以下`list-tags-for-resource`示例列出了附加到由其 ARN 指定的可扩展目标的标签键名称和值。

```
aws application-autoscaling list-tags-for-resource \  
  --resource-arn arn:aws:application-autoscaling:us-west-2:123456789012:scalable-  
  target/1234abcd56ab78cd901ef1234567890ab123
```

输出：

```
{  
  "Tags": {  
    "environment": "production"  
  }  
}
```

有关更多信息，请参阅《[Auto Scaling Auto Scaling 用户指南](#)》中的对应用程序 [Auto Scaling](#) 的[标记支持](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## put-scaling-policy

以下代码示例演示了如何使用 `put-scaling-policy`。

### AWS CLI

示例 1：应用具有预定义指标规范的目标跟踪扩展策略

以下`put-scaling-policy`示例将具有预定义指标规范的目标跟踪扩展策略应用于默认集群中名为 `web-app` 的 Amazon ECS 服务。该策略将服务的平均 CPU 利用率保持在 75%，横向扩展和缩减冷却时间为 60 秒。输出包含代表您创建的两个 CloudWatch 警报的 ARN 和名称。



```
aws application-autoscaling put-scaling-policy --service-namespace ecs \  
--scalable-dimension ecs:service:DesiredCount \  
--resource-id service/default/web-app \  
--policy-name cpu75-target-tracking-scaling-policy --policy-type  
TargetTrackingScaling \  
--target-tracking-scaling-policy-configuration file://config.json
```

此示例假设您在当前目录中有一个 config.json 文件，其中包含以下内容：

```
{  
  "TargetValue": 75.0,  
  "PredefinedMetricSpecification": {  
    "PredefinedMetricType": "ECSServiceAverageCPUUtilization"  
  },  
  "ScaleOutCooldown": 60,  
  "ScaleInCooldown": 60  
}
```

输出：

```
{  
  "PolicyARN": "arn:aws:autoscaling:us-west-2:012345678910:scalingPolicy:6d8972f3-  
efc8-437c-92d1-6270f29a66e7:resource/ecs/service/default/web-app:policyName/cpu75-  
target-tracking-scaling-policy",  
  "Alarms": [  
    {  
      "AlarmARN": "arn:aws:cloudwatch:us-  
west-2:012345678910:alarm:TargetTracking-service/default/web-app-AlarmHigh-d4f0770c-  
b46e-434a-a60f-3b36d653feca",  
      "AlarmName": "TargetTracking-service/default/web-app-AlarmHigh-d4f0770c-  
b46e-434a-a60f-3b36d653feca"  
    },  
    {  
      "AlarmARN": "arn:aws:cloudwatch:us-  
west-2:012345678910:alarm:TargetTracking-service/default/web-app-AlarmLow-1b437334-  
d19b-4a63-a812-6c67aaf2910d",  
      "AlarmName": "TargetTracking-service/default/web-app-AlarmLow-1b437334-  
d19b-4a63-a812-6c67aaf2910d"  
    }  
  ]  
}
```

## 示例 2：应用具有自定义指标规范的目标跟踪扩展策略

以下 `put-scaling-policy` 示例将带有自定义指标规范的目标跟踪扩展策略应用于默认集群中名为 `web-app` 的 Amazon ECS 服务。该策略将服务的平均利用率保持在 75%，横向扩展和缩减冷却时间为 60 秒。输出包含代表您创建的两个 CloudWatch 警报的 ARN 和名称。

```
aws application-autoscaling put-scaling-policy --service-namespace ecs \  
--scalable-dimension ecs:service:DesiredCount \  
--resource-id service/default/web-app \  
--policy-name cms75-target-tracking-scaling-policy \  
--policy-type TargetTrackingScaling \  
--target-tracking-scaling-policy-configuration file://config.json
```

此示例假设您在当前目录中有一个 `config.json` 文件，其中包含以下内容：

```
{  
  "TargetValue":75.0,  
  "CustomizedMetricSpecification":{  
    "MetricName":"MyUtilizationMetric",  
    "Namespace":"MyNamespace",  
    "Dimensions": [  
      {  
        "Name":"MyOptionalMetricDimensionName",  
        "Value":"MyOptionalMetricDimensionValue"  
      }  
    ],  
    "Statistic":"Average",  
    "Unit":"Percent"  
  },  
  "ScaleOutCooldown": 60,  
  "ScaleInCooldown": 60  
}
```

输出：

```
{  
  "PolicyARN": "arn:aws:autoscaling:us-west-2:012345678910:scalingPolicy:  
8784a896-b2ba-47a1-b08c-27301cc499a1:resource/ecs/service/default/web-  
app:policyName/cms75-target-tracking-scaling-policy",  
  "Alarms": [  
    {
```

```

        "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0",
        "AlarmName": "TargetTracking-service/default/web-app-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0"
    },
    {
        "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4",
        "AlarmName": "TargetTracking-service/default/web-app-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4"
    }
]
}

```

### 示例 3：为仅向外扩展应用目标跟踪扩展策略

以下 `put-scaling-policy` 示例将目标跟踪扩展策略应用于默认集群 `web-app` 中名为 `Amazon ECS 服务`。当来自 `Application Load Balancer` 的 `RequestCountPerTarget` 指标超过阈值时，该策略用于扩展 `ECS 服务`。输出包含代表您创建的 `CloudWatch 警报` 的 `ARN` 和名称。

```

aws application-autoscaling put-scaling-policy \
  --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/default/web-app \
  --policy-name alb-scale-out-target-tracking-scaling-policy \
  --policy-type TargetTrackingScaling \
  --target-tracking-scaling-policy-configuration file://config.json

```

`config.json` 的内容：

```

{
  "TargetValue": 1000.0,
  "PredefinedMetricSpecification": {
    "PredefinedMetricType": "ALBRequestCountPerTarget",
    "ResourceLabel": "app/EC2Co-EcsE1-1TKLTMITMM0E0/f37c06a68c1748aa/
targetgroup/EC2Co-Defau-LDNM7Q3ZH1ZN/6d4ea56ca2d6a18d"
  },
  "ScaleOutCooldown": 60,
  "ScaleInCooldown": 60,
  "DisableScaleIn": true
}

```

```
}

```

输出：

```
{
  "PolicyARN": "arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/ecs/service/default/web-app:policyName/alb-scale-out-target-tracking-scaling-policy",
  "Alarms": [
    {
      "AlarmName": "TargetTracking-service/default/web-app-AlarmHigh-d4f0770c-b46e-434a-a60f-3b36d653feca",
      "AlarmARN": "arn:aws:cloudwatch:us-west-2:123456789012:alarm:TargetTracking-service/default/web-app-AlarmHigh-d4f0770c-b46e-434a-a60f-3b36d653feca"
    }
  ]
}
```

有关更多信息，请参阅《[Auto Scaling Auto Scaling 用户指南](#)》中的[AWS 应用程序 Auto Scaling 的目标跟踪扩展策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutScalingPolicy](#) 中的。

## put-scheduled-action

以下代码示例演示了如何使用 `put-scheduled-action`。

### AWS CLI

向 DynamoDB 表添加计划操作

此示例将计划操作添加到 DynamoDB 表中，该表 `TestTable` 名为按周期性计划进行横向扩展。按照指定的计划（世界标准时间每天下午 12:15），如果当前容量低于为 `MinCapacity` 的指定值，Application Auto Scaling 会扩展到指定的值。 `MinCapacity`

命令：

```
aws application-autoscaling put-scheduled-action --service-namespace dynamodb
--scheduled-action-name my-recurring-action --schedule "cron(15 12 * * ? *)" --
resource-id table/TestTable --scalable-dimension dynamodb:table:WriteCapacityUnits
--scalable-target-action MinCapacity=6
```

有关更多信息，请参阅 [Application Auto Scaling 用户指南](#) 中的计划扩展。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [PutScheduledAction](#) 中的。

## register-scalable-target

以下代码示例演示了如何使用 `register-scalable-target`。

### AWS CLI

示例 1：将 ECS 服务注册为可扩展目标

以下 `register-scalable-target` 示例向 Application Auto Scaling 注册了一项 Amazon ECS 服务。它还向可扩展目标添加一个带有密钥名称 `environment` 和值 `production` 的标签。

```
aws application-autoscaling register-scalable-target \
  --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/default/web-app \
  --min-capacity 1 --max-capacity 10 \
  --tags environment=production
```

输出：

```
{
  "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

有关其他 AWS 服务和自定义资源的示例，请参阅《[Application Auto Scaling 用户指南](#)》中可与 [Application Auto Scaling 配合使用的](#) 服务中的主题。

示例 2：暂停可扩展目标的扩展活动

以下 `register-scalable-target` 示例暂停现有可扩展目标的扩展活动。

```
aws application-autoscaling register-scalable-target \
  --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:ReadCapacityUnits \
  --resource-id table/my-table \
  --suspended-state
DynamicScalingInSuspended=true,DynamicScalingOutSuspended=true,ScheduledScalingSuspended=true
```

输出：

```
{
  "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

有关更多信息，请参阅《Auto [Scaling 用户指南](#)》中的“[暂停和恢复应用程序 Auto Scaling 的缩放](#)”。

示例 3：恢复可扩展目标的扩展活动

以下 `register-scalable-target` 示例恢复现有可扩展目标的扩展活动。

```
aws application-autoscaling register-scalable-target \
  --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:ReadCapacityUnits \
  --resource-id table/my-table \
  --suspended-state
DynamicScalingInSuspended=false,DynamicScalingOutSuspended=false,ScheduledScalingSuspended=
```

输出：

```
{
  "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

有关更多信息，请参阅《Auto [Scaling 用户指南](#)》中的“[暂停和恢复应用程序 Auto Scaling 的缩放](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterScalableTarget](#) 中的。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

### AWS CLI

向可扩展目标添加标签

以下tag-resource示例将带有密钥名称environment和值production的标签添加到由其 ARN 指定的可扩展目标。

```
aws application-autoscaling tag-resource \  
  --resource-arn arn:aws:application-autoscaling:us-west-2:123456789012:scalable-  
target/1234abcd56ab78cd901ef1234567890ab123 \  
  --tags environment=production
```

此命令不生成任何输出。

有关更多信息，请参阅《[Auto Scaling Auto Scaling 用户指南](#)》中的对应用程序 Auto Scaling 的标记支持。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从可扩展目标中移除标签

以下untag-resource示例environment从其 ARN 指定的可扩展目标中删除带有密钥名称的标签对。

```
aws application-autoscaling untag-resource \  
  --resource-arn arn:aws:application-autoscaling:us-west-2:123456789012:scalable-  
target/1234abcd56ab78cd901ef1234567890ab123 \  
  --tag-keys "environment"
```

此命令不生成任何输出。

有关更多信息，请参阅《[Auto Scaling Auto Scaling 用户指南](#)》中的对应用程序 Auto Scaling 的标记支持。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## 使用 Application Discovery 服务示例 AWS CLI

以下代码示例向您展示了如何使用与 Application Discovery Service AWS Command Line Interface 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## describe-agents

以下代码示例演示了如何使用 describe-agents。

AWS CLI

描述具有指定收集状态的代理

此示例命令描述了收集状态为“已启动”或“已停止”的收集代理。

命令:

```
aws discovery describe-agents --filters
name="collectionStatus",values="STARTED","STOPPED",condition="EQUALS" --max-results
3
```

输出:

```
{
  "Snapshots": [
    {
      "version": "1.0.40.0",
      "agentType": "EC2",
      "hostName": "ip-172-31-40-234",
      "collectionStatus": "STOPPED",
      "agentNetworkInfoList": [
        {
          "macAddress": "06:b5:97:14:fc:0d",
          "ipAddress": "172.31.40.234"
        }
      ]
    }
  ]
}
```



```

    }
  ],
  "health": "UNKNOWN",
  "agentId": "i-003305c02a776e883",
  "registeredTime": "2016-12-09T19:05:06Z",
  "lastHealthPingTime": "2016-12-09T19:05:10Z"
},
{
  "version": "1.0.40.0",
  "agentType": "EC2",
  "hostName": "ip-172-31-39-64",
  "collectionStatus": "STARTED",
  "agentNetworkInfoList": [
    {
      "macAddress": "06:a1:0e:c7:b2:73",
      "ipAddress": "172.31.39.64"
    }
  ],
  "health": "SHUTDOWN",
  "agentId": "i-003a5e5e2b36cf8bd",
  "registeredTime": "2016-11-16T16:36:25Z",
  "lastHealthPingTime": "2016-11-16T16:47:37Z"
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAgents](#)中的。

## describe-configurations

以下代码示例演示了如何使用 describe-configurations。

### AWS CLI

#### 描述选定的资产配置

此示例命令描述了两台指定服务器的配置。该操作会从配置 ID 中检测资产的类型。每条命令只允许使用一种类型的资产。

#### 命令:

```
aws discovery describe-configurations --configuration-ids "d-server-099385097ef9fbcfb" "d-server-0c4f2dd1fee22c6c1"
```

输出：

```
{
  "configurations": [
    {
      "server.performance.maxCpuUsagePct": "0.0",
      "server.performance.maxDiskReadIOPS": "0.0",
      "server.performance.avgCpuUsagePct": "0.0",
      "server.type": "EC2",
      "server.performance.maxNetworkReadsPerSecondInKB": "0.19140625",
      "server.hostName": "ip-172-31-35-152",
      "server.configurationId": "d-server-0c4f2dd1fee22c6c1",
      "server.tags.hasMoreValues": "false",
      "server.performance.minFreeRAMInKB": "1543496.0",
      "server.osVersion": "3.14.48-33.39.amzn1.x86_64",
      "server.performance.maxDiskReadsPerSecondInKB": "0.0",
      "server.applications": "[]",
      "server.performance.numDisks": "1",
      "server.performance.numCpus": "1",
      "server.performance.numCores": "1",
      "server.performance.maxDiskWriteIOPS": "0.0",
      "server.performance.maxNetworkWritesPerSecondInKB": "0.82421875",
      "server.performance.avgDiskWritesPerSecondInKB": "0.0",
      "server.networkInterfaceInfo": "[{\"name\":\"eth0\",
\\\"macAddress\\\":\\\"06:A7:7D:3F:54:57\\\",\\\"ipAddress\\\":\\\"172.31.35.152\\\",\\\"netMask\\\":
\\\"255.255.240.0\\\"},{\"name\":\"lo\",\\\"macAddress\\\":\\\"00:00:00:00:00:00\\\",\\\"ipAddress
\\\":\\\"127.0.0.1\\\",\\\"netMask\\\":\\\"255.0.0.0\\\"},{\"name\":\"eth0\",\\\"macAddress\\\":
\\\"06:A7:7D:3F:54:57\\\",\\\"ipAddress\\\":\\\"fe80::4a7:7dff:fe3f:5457\\\"},{\"name\":\"lo\",
\\\"macAddress\\\":\\\"00:00:00:00:00:00\\\",\\\"ipAddress\\\":\\\":::1\\\"}]",
      "server.performance.avgNetworkReadsPerSecondInKB":
"0.049153645833333333",
      "server.tags": "[]",
      "server.applications.hasMoreValues": "false",
      "server.timeOfCreation": "2016-10-28 23:44:00.0",
      "server.agentId": "i-4447bc1b",
      "server.performance.maxDiskWritesPerSecondInKB": "0.0",
      "server.performance.avgDiskReadIOPS": "0.0",
      "server.performance.avgFreeRAMInKB": "1547210.1333333333",
      "server.performance.avgDiskReadsPerSecondInKB": "0.0",
      "server.performance.avgDiskWriteIOPS": "0.0",
      "server.performance.numNetworkCards": "2",
      "server.hypervisor": "xen",
      "server.networkInterfaceInfo.hasMoreValues": "false",
      "server.performance.avgNetworkWritesPerSecondInKB": "0.1380859375",
```

```

"server.osName": "Linux - Amazon Linux AMI release 2015.03",
"server.performance.totalRAMInKB": "1694732.0",
"server.cpuType": "x64"
},
{
  "server.performance.maxCpuUsagePct": "100.0",
  "server.performance.maxDiskReadIOPS": "0.0",
  "server.performance.avgCpuUsagePct": "14.733333333333338",
  "server.type": "EC2",
  "server.performance.maxNetworkReadsPerSecondInKB": "13.400390625",
  "server.hostName": "ip-172-31-42-208",
  "server.configurationId": "d-server-099385097ef9fbcb",
  "server.tags.hasMoreValues": "false",
  "server.performance.minFreeRAMInKB": "1531104.0",
  "server.osVersion": "3.14.48-33.39.amzn1.x86_64",
  "server.performance.maxDiskReadsPerSecondInKB": "0.0",
  "server.applications": "[]",
  "server.performance.numDisks": "1",
  "server.performance.numCpus": "1",
  "server.performance.numCores": "1",
  "server.performance.maxDiskWriteIOPS": "1.0",
  "server.performance.maxNetworkWritesPerSecondInKB": "12.271484375",
  "server.performance.avgDiskWritesPerSecondInKB":
"0.5333333333333334",
  "server.networkInterfaceInfo": "[{\\"name\\":\\"eth0\\",
\\"macAddress\\":\\"06:4A:79:60:75:61\\",\\"ipAddress\\":\\"172.31.42.208\\",\\"netMask
\\":\\"255.255.240.0\\"},{\\"name\\":\\"eth0\\",\\"macAddress\\":\\"06:4A:79:60:75:61\\",
\\"ipAddress\\":\\"fe80::44a:79ff:fe60:7561\\"},{\\"name\\":\\"lo\\",\\"macAddress\\":
\\"00:00:00:00:00:00\\",\\"ipAddress\\":\\"::1\\"},{\\"name\\":\\"lo\\",\\"macAddress\\":
\\"00:00:00:00:00:00\\",\\"ipAddress\\":\\"127.0.0.1\\",\\"netMask\\":\\"255.0.0.0\\"}]",
  "server.performance.avgNetworkReadsPerSecondInKB":
"2.8720052083333334",
  "server.tags": "[]",
  "server.applications.hasMoreValues": "false",
  "server.timeOfCreation": "2016-10-28 23:44:30.0",
  "server.agentId": "i-c142b99e",
  "server.performance.maxDiskWritesPerSecondInKB": "4.0",
  "server.performance.avgDiskReadIOPS": "0.0",
  "server.performance.avgFreeRAMInKB": "1534946.4",
  "server.performance.avgDiskReadsPerSecondInKB": "0.0",
  "server.performance.avgDiskWriteIOPS": "0.13333333333333336",
  "server.performance.numNetworkCards": "2",
  "server.hypervisor": "xen",
  "server.networkInterfaceInfo.hasMoreValues": "false",

```

```
        "server.performance.avgNetworkWritesPerSecondInKB":
"1.7977864583333332",
        "server.osName": "Linux - Amazon Linux AMI release 2015.03",
        "server.performance.totalRAMInKB": "1694732.0",
        "server.cpuType": "x64"
    }
]
}
```

## 描述选定的资产配置

此示例命令描述了两个指定应用程序的配置。该操作会从配置 ID 中检测资产的类型。每条命令只允许使用一种类型的资产。

### 命令:

```
aws discovery describe-configurations --configuration-ids "d-
application-0ac39bc0e4fad0e42" "d-application-02444a45288013764q"
```

### 输出:

```
{
  "configurations": [
    {
      "application.serverCount": "0",
      "application.name": "Application-12345",
      "application.lastModifiedTime": "2016-12-13 23:53:27.0",
      "application.description": "",
      "application.timeOfCreation": "2016-12-13 23:53:27.0",
      "application.configurationId": "d-application-0ac39bc0e4fad0e42"
    },
    {
      "application.serverCount": "0",
      "application.name": "Application-67890",
      "application.lastModifiedTime": "2016-12-13 23:53:33.0",
      "application.description": "",
      "application.timeOfCreation": "2016-12-13 23:53:33.0",
      "application.configurationId": "d-application-02444a45288013764"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeConfigurations](#)中的。

## list-configurations

以下代码示例演示了如何使用 list-configurations。

### AWS CLI

列出所有搜索到的符合一组筛选条件的服务器

此示例命令列出了已发现的与两种主机名模式中的任何一个匹配且未运行 Ubuntu 的服务器。

命令:

```
aws discovery list-configurations --configuration-type SERVER --filters
name="server.hostName",values="172-31-35","172-31-42",condition="CONTAINS"
name="server.osName",values="Ubuntu",condition="NOT_CONTAINS"
```

输出:

```
{
  "configurations": [
    {
      "server.osVersion": "3.14.48-33.39.amzn1.x86_64",
      "server.type": "EC2",
      "server.hostName": "ip-172-31-42-208",
      "server.timeOfCreation": "2016-10-28 23:44:30.0",
      "server.configurationId": "d-server-099385097ef9fbcfb",
      "server.osName": "Linux - Amazon Linux AMI release 2015.03",
      "server.agentId": "i-c142b99e"
    },
    {
      "server.osVersion": "3.14.48-33.39.amzn1.x86_64",
      "server.type": "EC2",
      "server.hostName": "ip-172-31-35-152",
      "server.timeOfCreation": "2016-10-28 23:44:00.0",
      "server.configurationId": "d-server-0c4f2dd1fee22c6c1",
      "server.osName": "Linux - Amazon Linux AMI release 2015.03",
      "server.agentId": "i-4447bc1b"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListConfigurations](#)中的。

## AppRegistry 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AppRegistry。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-attribute-group**

以下代码示例演示了如何使用 `associate-attribute-group`。

AWS CLI

关联属性组

以下 `associate-attribute-group` 示例将您 AWS 账户中的特定属性组与账户中的特定应用程序相关联。AWS

```
aws servicecatalog-appregistry associate-attribute-group \  
  --application "ExampleApplication" \  
  --attribute-group "ExampleAttributeGroup"
```

输出：

```
{  
  "applicationArn": "arn:aws:servicecatalog:us-west-2:813737243517:/  
applications/0ars38r6btoohvpvd9gqrptt91",  
  "attributeGroupArn": "arn:aws:servicecatalog:us-west-2:813737243517:/attribute-  
groups/01sj5xdwhbw54kejwnt09fnpcl"
```

```
}
```

有关更多信息，请参阅《S AWS ervice Catalog 管理员指南》中的[关联和取消关联属性组](#)。  
AppRegistry

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateAttributeGroup](#)中的。

## create-application

以下代码示例演示了如何使用 create-application。

### AWS CLI

#### 创建应用程序

以下create-application示例在您的 AWS 账户中创建了一个新应用程序。

```
aws servicecatalog-appregistry create-application \  
  --name "ExampleApplication"
```

输出：

```
{  
  "application": {  
    "id": "0ars38r6btoohvpvd9gqrptt91",  
    "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/  
applications/0ars38r6btoohvpvd9gqrptt91",  
    "name": "ExampleApplication",  
    "creationTime": "2023-02-28T21:10:10.820000+00:00",  
    "lastUpdateTime": "2023-02-28T21:10:10.820000+00:00",  
    "tags": {}  
  }  
}
```

有关更多信息，请参阅《S AWS ervice Catalog AppRegistry 管理员指南》中的[创建应用程序](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateApplication](#)中的。

## create-attribute-group

以下代码示例演示了如何使用 create-attribute-group。

## AWS CLI

### 创建属性组

以下 `create-attribute-group` 示例在您的 AWS 账户中创建了一个新的属性组。

```
aws servicecatalog-appregistry create-attribute-group \  
  --name "ExampleAttributeGroup" \  
  --attributes '{"SomeKey1":"SomeValue1","SomeKey2":"SomeValue2"}'
```

输出：

```
{  
  "attributeGroup": {  
    "id": "01sj5xdwhbw54kejwnt09fnpc1",  
    "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/attribute-  
groups/01sj5xdwhbw54kejwnt09fnpc1",  
    "name": "ExampleAttributeGroup",  
    "creationTime": "2023-02-28T20:38:01.389000+00:00",  
    "lastUpdateTime": "2023-02-28T20:38:01.389000+00:00",  
    "tags": {}  
  }  
}
```

有关更多信息，请参阅《S AWS ervice Catalog AppRegistry 管理员指南》中的[创建属性组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateAttributeGroup](#) 中的。

## delete-application

以下代码示例演示了如何使用 `delete-application`。

### AWS CLI

#### 删除应用程序

以下 `delete-application` 示例删除了您 AWS 账户中的特定应用程序。

```
aws servicecatalog-appregistry delete-application \  
  --application "ExampleApplication3"
```



输出：

```
{
  "application": {
    "id": "055gw7aynr1i5mbv7kjwtzx5945",
    "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/
applications/055gw7aynr1i5mbv7kjwtzx5945",
    "name": "ExampleApplication3",
    "creationTime": "2023-02-28T22:06:28.228000+00:00",
    "lastUpdateTime": "2023-02-28T22:06:28.228000+00:00"
  }
}
```

有关更多信息，请参阅《S AWS ervice Catalog AppRegistry 管理员指南》中的[删除应用程序](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteApplication](#)中的。

## delete-attribute-group

以下代码示例演示了如何使用 delete-attribute-group。

### AWS CLI

#### 示例 8：删除属性组

以下delete-attribute-group示例删除了您 AWS 账户中的特定属性组。

```
aws servicecatalog-appregistry delete-attribute-group \
  --attribute-group "ExampleAttributeGroup3"
```

输出：

```
{
  "attributeGroup": {
    "id": "011ge6y3emyjijt8dw8jn6r0hv",
    "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/attribute-
groups/011ge6y3emyjijt8dw8jn6r0hv",
    "name": "ExampleAttributeGroup3",
    "creationTime": "2023-02-28T22:05:35.224000+00:00",
    "lastUpdateTime": "2023-02-28T22:05:35.224000+00:00"
  }
}
```

有关更多信息，请参阅《S AWS ervice Catalog AppRegistry 管理员指南》中的[删除属性组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAttributeGroup](#)中的。

## get-application

以下代码示例演示了如何使用 get-application。

### AWS CLI

#### 获取申请

以下get-application示例检索有关您 AWS 账户中特定应用程序的元数据信息。

```
aws servicecatalog-appregistry get-application \  
  --application "ExampleApplication"
```

输出：

```
{  
  "id": "0ars38r6btoohvpvd9gqrptt91",  
  "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/  
applications/0ars38r6btoohvpvd9gqrptt91",  
  "name": "ExampleApplication",  
  "creationTime": "2023-02-28T21:10:10.820000+00:00",  
  "lastUpdateTime": "2023-02-28T21:10:10.820000+00:00",  
  "associatedResourceCount": 0,  
  "tags": {  
    "aws:servicecatalog:applicationName": "ExampleApplication"  
  },  
  "integrations": {  
    "resourceGroup": {  
      "state": "CREATE_COMPLETE",  
      "arn": "arn:aws:resource-groups:us-west-2:813737243517:group/  
AWS_AppRegistry_Application-ExampleApplication"  
    }  
  }  
}
```

有关更多信息，请参阅《S AWS ervice Catalog AppRegistry 管理员指南》中的[“使用应用程序”的详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetApplication](#)中的。

## get-attribute-group

以下代码示例演示了如何使用 `get-attribute-group`。

### AWS CLI

#### 获取属性组

以下`get-attribute-group`示例检索您 AWS 账户中的特定属性组。

```
aws servicecatalog-appregistry get-attribute-group \  
  --attribute-group "ExampleAttributeGroup"
```

输出：

```
{  
  "id": "01sj5xdwhbw54kejwnt09fnpc1",  
  "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/attribute-  
groups/01sj5xdwhbw54kejwnt09fnpc1",  
  "name": "ExampleAttributeGroup",  
  "attributes": "{\"SomeKey1\": \"SomeValue1\", \"SomeKey2\": \"SomeValue2\"}",  
  "creationTime": "2023-02-28T20:38:01.389000+00:00",  
  "lastUpdateTime": "2023-02-28T20:38:01.389000+00:00",  
  "tags": {  
    "aws:servicecatalog:attributeGroupName": "ExampleAttributeGroup"  
  }  
}
```

有关更多信息，请参阅《S AWS ervice Catalog AppRegistry 管理员指南》中的[管理属性组的元数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAttributeGroup](#)中的。

## list-applications

以下代码示例演示了如何使用 `list-applications`。

### AWS CLI

#### 列出应用程序

以下`list-applications`示例检索您 AWS 账户中所有应用程序的列表。

```
aws servicecatalog-appregistry list-applications
```

输出：

```
{
  "applications": [
    {
      "id": "03axw94pjfj3uan00tcgbrxnkw",
      "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/
applications/03axw94pjfj3uan00tcgbrxnkw",
      "name": "ExampleApplication2",
      "creationTime": "2023-02-28T21:59:34.094000+00:00",
      "lastUpdateTime": "2023-02-28T21:59:34.094000+00:00"
    },
    {
      "id": "055gw7aynr1i5mbv7kjwzx5945",
      "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/
applications/055gw7aynr1i5mbv7kjwzx5945",
      "name": "ExampleApplication3",
      "creationTime": "2023-02-28T22:06:28.228000+00:00",
      "lastUpdateTime": "2023-02-28T22:06:28.228000+00:00"
    },
    {
      "id": "0ars38r6btoohvpvd9gqrptt91",
      "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/
applications/0ars38r6btoohvpvd9gqrptt91",
      "name": "ExampleApplication",
      "description": "This is an example application",
      "creationTime": "2023-02-28T21:10:10.820000+00:00",
      "lastUpdateTime": "2023-02-28T21:24:19.729000+00:00"
    }
  ]
}
```

有关更多信息，请参阅《S AWS ervice Catalog AppRegistry 管理员指南》中的[查看应用程序详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListApplications](#)中的。

## list-associated-attribute-groups

以下代码示例演示了如何使用 list-associated-attribute-groups。

## AWS CLI

### 列出关联的属性组

以下 `list-associated-attribute-groups` 示例检索您的 AWS 账户中与您的账户中的特定应用程序关联的所有属性组的列表。AWS

```
aws servicecatalog-appregistry list-associated-attribute-groups \  
  --application "ExampleApplication"
```

输出：

```
{  
  "attributeGroups": [  
    "01sj5xdwhbw54kejwnt09fnpc1"  
  ]  
}
```

有关更多信息，请参阅《S AWS ervic e Catalog 管理员指南》中的[关联和取消关联属性组](#)。AppRegistry

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAssociatedAttributeGroups](#) 中的。

## list-attribute-groups-for-application

以下代码示例演示了如何使用 `list-attribute-groups-for-application`。

## AWS CLI

### 列出应用程序的属性组

以下 `list-attribute-groups-for-application` 示例列出了您 AWS 账户中与账户中特定应用程序关联的所有属性组的详细信息。AWS

```
aws servicecatalog-appregistry list-attribute-groups-for-application \  
  --application "ExampleApplication"
```

输出：

```
{  
  "attributeGroupsDetails": [  
    {
```

```
    "id": "01sj5xdwhbw54kejwnt09fnpc1",
    "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/attribute-
groups/01sj5xdwhbw54kejwnt09fnpc1",
    "name": "ExampleAttributeGroup"
  }
]
}
```

有关更多信息，请参阅《S AWS ervice Catalog AppRegistry 管理员指南》中的[查看属性组详情](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAttributeGroupsForApplication](#)中的。

## list-attribute-groups

以下代码示例演示了如何使用 list-attribute-groups。

### AWS CLI

#### 列出属性组

以下list-attribute-groups示例检索您的 AWS 账户中所有属性组的列表。

```
aws servicecatalog-appregistry list-attribute-groups
```

输出：

```
{
  "attributeGroups": [
    {
      "id": "011ge6y3emyjijt8dw8jn6r0hv",
      "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/attribute-
groups/011ge6y3emyjijt8dw8jn6r0hv",
      "name": "ExampleAttributeGroup3",
      "creationTime": "2023-02-28T22:05:35.224000+00:00",
      "lastUpdateTime": "2023-02-28T22:05:35.224000+00:00"
    },
    {
      "id": "01sj5xdwhbw54kejwnt09fnpc1",
      "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/attribute-
groups/01sj5xdwhbw54kejwnt09fnpc1",
      "name": "ExampleAttributeGroup",
      "description": "This is an example attribute group",
      "creationTime": "2023-02-28T20:38:01.389000+00:00",
```

```

        "lastUpdateTime": "2023-02-28T21:02:04.559000+00:00"
    },
    {
        "id": "03n1yffgq6d18vwrzxf0c70nm3",
        "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/attribute-
groups/03n1yffgq6d18vwrzxf0c70nm3",
        "name": "ExampleAttributeGroup2",
        "creationTime": "2023-02-28T21:57:30.687000+00:00",
        "lastUpdateTime": "2023-02-28T21:57:30.687000+00:00"
    }
]
}

```

有关更多信息，请参阅《S AWS ervice Catalog AppRegistry 管理员指南》中的[查看属性组详情](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAttributeGroups](#)中的。

## update-application

以下代码示例演示了如何使用 update-application。

### AWS CLI

#### 更新应用程序

以下update-application示例更新了您 AWS 账户中的特定应用程序，使其包含描述。

```

aws servicecatalog-appregistry update-application \
  --application "ExampleApplication" \
  --description "This is an example application"

```

输出：

```

{
  "application": {
    "id": "0ars38r6btoohvpvd9gqrptt91",
    "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/
applications/0ars38r6btoohvpvd9gqrptt91",
    "name": "ExampleApplication",
    "description": "This is an example application",
    "creationTime": "2023-02-28T21:10:10.820000+00:00",
    "lastUpdateTime": "2023-02-28T21:24:19.729000+00:00",
    "tags": {

```

```
        "aws:servicecatalog:applicationName": "ExampleApplication"
      }
    }
  }
```

有关更多信息，请参阅《S AWS ervice Catalog AppRegistry 管理员指南》中的[编辑应用程序](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateApplication](#)中的。

## update-attribute-group

以下代码示例演示了如何使用 update-attribute-group。

### AWS CLI

#### 更新属性组

以下update-attribute-group示例更新了您 AWS 账户中的特定属性组，使其包含描述。

```
aws servicecatalog-appregistry update-attribute-group \
  --attribute-group "ExampleAttributeGroup" \
  --description "This is an example attribute group"
```

输出：

```
{
  "attributeGroup": {
    "id": "01sj5xdwhbw54kejwnt09fnpc1",
    "arn": "arn:aws:servicecatalog:us-west-2:813737243517:/attribute-
groups/01sj5xdwhbw54kejwnt09fnpc1",
    "name": "ExampleAttributeGroup",
    "description": "This is an example attribute group",
    "creationTime": "2023-02-28T20:38:01.389000+00:00",
    "lastUpdateTime": "2023-02-28T21:02:04.559000+00:00",
    "tags": {
      "aws:servicecatalog:attributeGroupName": "ExampleAttributeGroup"
    }
  }
}
```

有关更多信息，请参阅《S AWS ervice Catalog AppRegistry 管理员指南》中的[编辑属性组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAttributeGroup](#)中的。



## 使用 Athena 的示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Athena 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### batch-get-named-query

以下代码示例演示了如何使用 batch-get-named-query。

AWS CLI

返回有关多个查询的信息

以下batch-get-named-query示例返回有关具有指定 ID 的命名查询的信息。

```
aws athena batch-get-named-query \  
  --named-query-ids a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 a1b2c3d4-5678-90ab-cdef-  
  EXAMPLE22222 a1b2c3d4-5678-90ab-cdef-EXAMPLE33333
```

输出：

```
{  
  "NamedQueries": [  
    {  
      "Name": "Flights Select Query",  
      "Description": "Sample query to get the top 10 airports with the most  
number of departures since 2000",
```

```

        "Database": "sampledb",
        "QueryString": "SELECT origin, count(*) AS total_departures\nFROM
\nflights_parquet\nWHERE year >= '2000'\nGROUP BY origin\nORDER BY total_departures
DESC\nLIMIT 10;",
        "NamedQueryId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "WorkGroup": "primary"
    },
    {
        "Name": "Load flights table partitions",
        "Description": "Sample query to load flights table partitions using MSCK
REPAIR TABLE statement",
        "Database": "sampledb",
        "QueryString": "MSCK REPAIR TABLE flights_parquet;",
        "NamedQueryId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
        "WorkGroup": "primary"
    },
    {
        "Name": "CloudFront Select Query",
        "Description": "Sample query to view requests per operating system
during a particular time frame",
        "Database": "sampledb",
        "QueryString": "SELECT os, COUNT(*) count FROM cloudfront_logs WHERE
date BETWEEN date '2014-07-05' AND date '2014-08-05' GROUP BY os;",
        "NamedQueryId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
        "WorkGroup": "primary"
    }
],
    "UnprocessedNamedQueryIds": []
}

```

有关更多信息，请参阅 Amazon Athena 用户指南中的[使用 Amazon Athena 运行 SQL 查询](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[BatchGetNamedQuery](#)中的。

## batch-get-query-execution

以下代码示例演示了如何使用 batch-get-query-execution。

### AWS CLI

返回有关一个或多个查询执行的信息

以下 batch-get-query-execution 示例返回具有指定查询 ID 的查询的查询执行信息。

```
aws athena batch-get-query-execution \
  --query-execution-ids a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 a1b2c3d4-5678-90ab-
  cdef-EXAMPLE22222
```

输出：

```
{
  "QueryExecutions": [
    {
      "QueryExecutionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "Query": "create database if not exists webdata",
      "StatementType": "DDL",
      "ResultConfiguration": {
        "OutputLocation": "s3://awsdoc-example-bucket/a1b2c3d4-5678-90ab-
        cdef-EXAMPLE11111.txt"
      },
      "QueryExecutionContext": {},
      "Status": {
        "State": "SUCCEEDED",
        "SubmissionDateTime": 1593470720.592,
        "CompletionDateTime": 1593470720.902
      },
      "Statistics": {
        "EngineExecutionTimeInMillis": 232,
        "DataScannedInBytes": 0,
        "TotalExecutionTimeInMillis": 310,
        "ResultConfiguration": {
          "QueryQueueTimeInMillis": 50,
          "ServiceProcessingTimeInMillis": 28
        },
      },
      "WorkGroup": "AthenaAdmin"
    },
    {
      "QueryExecutionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "Query": "select date, location, browser, uri, status from
      cloudfront_logs where method = 'GET' and status = 200 and location like 'SF0%'
      limit 10",
      "StatementType": "DML",
      "ResultConfiguration": {
        "OutputLocation": "s3://awsdoc-example-bucket/a1b2c3d4-5678-90ab-
        cdef-EXAMPLE22222.csv"
      },
    },
  ]
}
```

```

    "QueryExecutionContext": {
      "Database": "mydatabase",
      "Catalog": "awsdatacatalog"
    },
    "Status": {
      "State": "SUCCEEDED",
      "SubmissionDateTime": 1593469842.665,
      "CompletionDateTime": 1593469846.486
    },
    "Statistics": {
      "EngineExecutionTimeInMillis": 3600,
      "DataScannedInBytes": 203089,
      "TotalExecutionTimeInMillis": 3821,
      "QueryQueueTimeInMillis": 267,
      "QueryPlanningTimeInMillis": 1175
    },
    "WorkGroup": "AthenaAdmin"
  }
],
"UnprocessedQueryExecutionIds": []
}

```

有关更多信息，请参阅 Amazon Athena 用户指南中的[使用 Amazon Athena 运行 SQL 查询](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[BatchGetQueryExecution](#)中的。

## create-data-catalog

以下代码示例演示了如何使用 create-data-catalog。

### AWS CLI

#### 创建数据目录

以下 create-data-catalog 示例创建了 dynamo\_db\_catalog 数据目录。

```

aws athena create-data-catalog \
  --name dynamo_db_catalog \
  --type LAMBDA \
  --description "DynamoDB Catalog" \
  --parameters function=arn:aws:lambda:us-
west-2:111122223333:function:dynamo_db_lambda

```

此命令不生成任何输出。要查看结果，请使用 `aws athena get-data-catalog --name dynamo_db_catalog`。

有关更多信息，请参阅 Amazon Athena 用户指南 [create-data-catalog 中的注册目录](#) 。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDataCatalog](#) 中的。

## create-named-query

以下代码示例演示了如何使用 `create-named-query`。

### AWS CLI

#### 创建命名查询

以下 `create-named-query` 示例在 AthenaAdmin 工作组中创建了一个已保存的查询，该查询 `flights_parquet` 表中列出了 2016 年 1 月从西雅图飞往纽约的航班，这些航班的起飞和到达时间都延误了十分钟以上。由于表中的机场代码值是包含双引号的字符串（例如，“SEA”），因此它们被反斜杠转义并用单引号包围。

```
aws athena create-named-query \  
  --name "SEA to JFK delayed flights Jan 2016" \  
  --description "Both arrival and departure delayed more than 10 minutes." \  
  --database sampledb \  
  --query-string "SELECT flightdate, carrier, flightnum, origin, dest,  
depdelayminutes, arrdelayminutes FROM sampledb.flights_parquet WHERE yr = 2016 AND  
month = 1 AND origin = '\"SEA\"' AND dest = '\"JFK\"' AND depdelayminutes > 10 AND  
arrdelayminutes > 10" \  
  --work-group AthenaAdmin
```

输出：

```
{  
  "NamedQueryId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"  
}
```

有关更多信息，请参阅 Amazon Athena 用户指南中的 [使用 Amazon Athena 运行 SQL 查询](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateNamedQuery](#) 中的。

## create-work-group

以下代码示例演示了如何使用 create-work-group。

### AWS CLI

#### 创建工作组

以下create-work-group示例创建了一个名为的工作组Data\_Analyst\_Group，该工作组具有查询结果的输出位置s3://awsdoc-example-bucket。该命令创建一个覆盖客户端配置设置的工作组，其中包括查询结果的输出位置。该命令还启用 CloudWatch 指标，并向工作组添加三个键值标签对，以将其与其他工作组区分开来。请注意，--configuration参数在分隔其选项的逗号前没有空格。

```
aws athena create-work-group \  
  --name Data_Analyst_Group \  
  --configuration ResultConfiguration={OutputLocation="s3://awsdoc-example-  
bucket"},EnforceWorkGroupConfiguration="true",PublishCloudWatchMetricsEnabled="true"  
 \  
  --description "Workgroup for data analysts" \  
  --tags Key=Division,Value=West Key=Location,Value=Seattle Key=Team,Value="Big  
Data"
```

此命令不生成任何输出。要查看结果，请使用aws athena get-work-group --work-group Data\_Analyst\_Group。

有关更多信息，请参阅 Amazon Athena 用户指南中的[管理工作组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateWorkGroup](#)中的。

## delete-data-catalog

以下代码示例演示了如何使用 delete-data-catalog。

### AWS CLI

#### 删除数据目录

以下delete-data-catalog示例删除了UnusedDataCatalog数据目录。

```
aws athena delete-data-catalog \  
  --name UnusedDataCatalog
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Athena 用户指南 [delete-data-catalog](#) 中的 [删除目录](#) 。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteDataCatalog](#) 中的。

## delete-named-query

以下代码示例演示了如何使用 delete-named-query。

### AWS CLI

#### 删除命名查询

以下 delete-named-query 示例删除具有指定 ID 的命名查询。

```
aws athena delete-named-query \  
  --named-query-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Athena 用户指南中的 [使用 Amazon Athena 运行 SQL 查询](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteNamedQuery](#) 中的。

## delete-work-group

以下代码示例演示了如何使用 delete-work-group。

### AWS CLI

#### 删除工作组

以下 delete-work-group 示例删除了 TeamB 工作组。

```
aws athena delete-work-group \  
  --work-group TeamB
```

此命令不生成任何输出。要确认删除，请使用 `aws athena list-work-groups`。

有关更多信息，请参阅 Amazon Athena 用户指南中的 [管理工作组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteWorkGroup](#) 中的。

## get-data-catalog

以下代码示例演示了如何使用 `get-data-catalog`。

### AWS CLI

返回有关数据目录的信息

以下`get-data-catalog`示例返回有关`dynamo_db_catalog`数据目录的信息。

```
aws athena get-data-catalog \  
  --name dynamo_db_catalog
```

输出：

```
{  
  "DataCatalog": {  
    "Name": "dynamo_db_catalog",  
    "Description": "DynamoDB Catalog",  
    "Type": "LAMBDA",  
    "Parameters": {  
      "catalog": "dynamo_db_catalog",  
      "metadata-function": "arn:aws:lambda:us-  
west-2:111122223333:function:dynamo_db_lambda",  
      "record-function": "arn:aws:lambda:us-  
west-2:111122223333:function:dynamo_db_lambda"  
    }  
  }  
}
```

有关更多信息，请参阅 Amazon Athena 用户指南 `get-data-catalog` 中的 [显示目录详情](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDataCatalog](#) 中的。

## get-database

以下代码示例演示了如何使用 `get-database`。

### AWS CLI

返回有关数据目录中数据库的信息

以下`get-database`示例返回有关`AwsDataCatalog`数据目录中`sampleddb`数据库的信息。



```
aws athena get-database \  
  --catalog-name AwsDataCatalog \  
  --database-name sampledb
```

输出：

```
{  
  "Database": {  
    "Name": "sampledb",  
    "Description": "Sample database",  
    "Parameters": {  
      "CreatedBy": "Athena",  
      "EXTERNAL": "TRUE"  
    }  
  }  
}
```

有关更多信息，请参阅 Amazon Athena [用户指南中的显示数据库详情：获取数据库](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDatabase](#) 中的。

## get-named-query

以下代码示例演示了如何使用 get-named-query。

### AWS CLI

返回命名查询

以下 get-named-query 示例返回有关具有指定 ID 的查询的信息。

```
aws athena get-named-query \  
  --named-query-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "NamedQuery": {  
    "Name": "CloudFront Logs - SF0",  
    "Description": "Shows successful GET request data for SF0",  
    "Database": "default",
```

```
    "QueryString": "select date, location, browser, uri, status from
cloudfront_logs where method = 'GET' and status = 200 and location like 'SF0%'
limit 10",
    "NamedQueryId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "WorkGroup": "AthenaAdmin"
  }
}
```

有关更多信息，请参阅 Amazon Athena 用户指南中的[使用 Amazon Athena 运行 SQL 查询](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetNamedQuery](#)中的。

## get-query-execution

以下代码示例演示了如何使用 get-query-execution。

### AWS CLI

返回有关查询执行的信息

以下 get-query-execution 示例返回有关具有指定查询 ID 的查询的信息。

```
aws athena get-query-execution \
  --query-execution-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
  "QueryExecution": {
    "QueryExecutionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Query": "select date, location, browser, uri, status from cloudfront_logs
where method = 'GET
' and status = 200 and location like 'SF0%' limit 10",
    "StatementType": "DML",
    "ResultConfiguration": {
      "OutputLocation": "s3://awsdoc-example-bucket/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111.csv"
    },
    "QueryExecutionContext": {
      "Database": "mydatabase",
      "Catalog": "awsdatacatalog"
    },
    "Status": {
```

```

        "State": "SUCCEEDED",
        "SubmissionDateTime": 1593469842.665,
        "CompletionDateTime": 1593469846.486
    },
    "Statistics": {
        "EngineExecutionTimeInMillis": 3600,
        "DataScannedInBytes": 203089,
        "TotalExecutionTimeInMillis": 3821,
        "QueryQueueTimeInMillis": 267,
        "QueryPlanningTimeInMillis": 1175
    },
    "WorkGroup": "AthenaAdmin"
}
}

```

有关更多信息，请参阅 Amazon Athena 用户指南中的[使用 Amazon Athena 运行 SQL 查询](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetQueryExecution](#)中的。

## get-query-results

以下代码示例演示了如何使用 get-query-results。

### AWS CLI

#### 返回查询结果

以下 get-query-results 示例返回具有指定查询 ID 的查询的结果。

```

aws athena get-query-results \
  --query-execution-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111

```

#### 输出：

```

{
  "ResultSet": {
    "Rows": [
      {
        "Data": [
          {
            "VarCharValue": "date"
          },
          {

```

```
        "VarCharValue": "location"
      },
      {
        "VarCharValue": "browser"
      },
      {
        "VarCharValue": "uri"
      },
      {
        "VarCharValue": "status"
      }
    ]
  },
  {
    "Data": [
      {
        "VarCharValue": "2014-07-05"
      },
      {
        "VarCharValue": "SF04"
      },
      {
        "VarCharValue": "Safari"
      },
      {
        "VarCharValue": "/test-image-2.jpeg"
      },
      {
        "VarCharValue": "200"
      }
    ]
  },
  {
    "Data": [
      {
        "VarCharValue": "2014-07-05"
      },
      {
        "VarCharValue": "SF04"
      },
      {
        "VarCharValue": "Opera"
      }
    ]
  }
}
```

```
        "VarCharValue": "/test-image-2.jpeg"
      },
      {
        "VarCharValue": "200"
      }
    ]
  },
  {
    "Data": [
      {
        "VarCharValue": "2014-07-05"
      },
      {
        "VarCharValue": "SF04"
      },
      {
        "VarCharValue": "Firefox"
      },
      {
        "VarCharValue": "/test-image-3.jpeg"
      },
      {
        "VarCharValue": "200"
      }
    ]
  },
  {
    "Data": [
      {
        "VarCharValue": "2014-07-05"
      },
      {
        "VarCharValue": "SF04"
      },
      {
        "VarCharValue": "Lynx"
      },
      {
        "VarCharValue": "/test-image-3.jpeg"
      },
      {
        "VarCharValue": "200"
      }
    ]
  }
]
```

```
    },
    {
      "Data": [
        {
          "VarCharValue": "2014-07-05"
        },
        {
          "VarCharValue": "SF04"
        },
        {
          "VarCharValue": "IE"
        },
        {
          "VarCharValue": "/test-image-2.jpeg"
        },
        {
          "VarCharValue": "200"
        }
      ]
    },
    {
      "Data": [
        {
          "VarCharValue": "2014-07-05"
        },
        {
          "VarCharValue": "SF04"
        },
        {
          "VarCharValue": "Opera"
        },
        {
          "VarCharValue": "/test-image-1.jpeg"
        },
        {
          "VarCharValue": "200"
        }
      ]
    },
    {
      "Data": [
        {
          "VarCharValue": "2014-07-05"
        },
```

```
        {
          "VarCharValue": "SF04"
        },
        {
          "VarCharValue": "Chrome"
        },
        {
          "VarCharValue": "/test-image-3.jpeg"
        },
        {
          "VarCharValue": "200"
        }
      ]
    },
    {
      "Data": [
        {
          "VarCharValue": "2014-07-05"
        },
        {
          "VarCharValue": "SF04"
        },
        {
          "VarCharValue": "Firefox"
        },
        {
          "VarCharValue": "/test-image-2.jpeg"
        },
        {
          "VarCharValue": "200"
        }
      ]
    },
    {
      "Data": [
        {
          "VarCharValue": "2014-07-05"
        },
        {
          "VarCharValue": "SF04"
        },
        {
          "VarCharValue": "Chrome"
        }
      ]
    }
  ]
}
```

```
        {
          "VarCharValue": "/test-image-3.jpeg"
        },
        {
          "VarCharValue": "200"
        }
      ]
    },
    {
      "Data": [
        {
          "VarCharValue": "2014-07-05"
        },
        {
          "VarCharValue": "SF04"
        },
        {
          "VarCharValue": "IE"
        },
        {
          "VarCharValue": "/test-image-2.jpeg"
        },
        {
          "VarCharValue": "200"
        }
      ]
    }
  ],
  "ResultSetMetadata": {
    "ColumnInfo": [
      {
        "CatalogName": "hive",
        "SchemaName": "",
        "TableName": "",
        "Name": "date",
        "Label": "date",
        "Type": "date",
        "Precision": 0,
        "Scale": 0,
        "Nullable": "UNKNOWN",
        "CaseSensitive": false
      },
      {
        "CatalogName": "hive",
```



```
    "SchemaName": "",
    "TableName": "",
    "Name": "location",
    "Label": "location",
    "Type": "varchar",
    "Precision": 2147483647,
  "Data": [
    {
      "Scale": 0,
      "Nullable": "UNKNOWN",
      "CaseSensitive": true
    },
    {
      "CatalogName": "hive",
      "SchemaName": "",
      "TableName": "",
      "Name": "browser",
      "Label": "browser",
      "Type": "varchar",
      "Precision": 2147483647,
      "Scale": 0,
      "Nullable": "UNKNOWN",
      "CaseSensitive": true
    },
    {
      "CatalogName": "hive",
      "SchemaName": "",
      "TableName": "",
      "Name": "uri",
      "Label": "uri",
      "Type": "varchar",
      "Precision": 2147483647,
      "Scale": 0,
      "Nullable": "UNKNOWN",
      "CaseSensitive": true
    },
    {
      "CatalogName": "hive",
      "SchemaName": "",
      "TableName": "",
      "Name": "status",
      "Label": "status",
      "Type": "integer",
      "Precision": 10,
```

```

        "Scale": 0,
        "Nullable": "UNKNOWN",
        "CaseSensitive": false
      }
    ]
  },
  "UpdateCount": 0
}

```

有关更多信息，请参阅 Amazon Athena 用户指南中的[使用查询结果、输出文件和查询历史记录](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetQueryResults](#)中的。

## get-table-metadata

以下代码示例演示了如何使用 get-table-metadata。

### AWS CLI

返回有关表的元数据信息

以下 get-table-metadata 示例从数据目录的 sampledb 数据库返回有关 counties 表的元数据信息，包括列名及其 AwsDataCatalog 数据类型。

```

aws athena get-table-metadata \
  --catalog-name AwsDataCatalog \
  --database-name sampledb \
  --table-name counties

```

输出：

```

{
  "TableMetadata": {
    "Name": "counties",
    "CreateTime": 1593559968.0,
    "LastAccessTime": 0.0,
    "TableType": "EXTERNAL_TABLE",
    "Columns": [
      {
        "Name": "name",
        "Type": "string",
        "Comment": "from deserializer"
      }
    ]
  }
}

```

```
    },
    {
      "Name": "boundaryshape",
      "Type": "binary",
      "Comment": "from deserializer"
    },
    {
      "Name": "motto",
      "Type": "string",
      "Comment": "from deserializer"
    },
    {
      "Name": "population",
      "Type": "int",
      "Comment": "from deserializer"
    }
  ],
  "PartitionKeys": [],
  "Parameters": {
    "EXTERNAL": "TRUE",
    "inputformat": "com.esri.json.hadoop.EnclosedJsonInputFormat",
    "location": "s3://awsdoc-example-bucket/json",
    "outputformat":
"org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat",
    "serde.param.serialization.format": "1",
    "serde.serialization.lib": "com.esri.hadoop.hive.serde.JsonSerde",
    "transient_lastDdlTime": "1593559968"
  }
}
```

有关更多信息，请参阅 Amazon Athena 用户指南 `get-table-metadata` 中的 [显示表格详情](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetTableMetadata](#) 中的。

## get-work-group

以下代码示例演示了如何使用 `get-work-group`。

### AWS CLI

返回有关工作组的信息

以下 `get-work-group` 示例返回有关 AthenaAdmin 工作组的信息。

```
aws athena get-work-group \  
  --work-group AthenaAdmin
```

输出：

```
{  
  "WorkGroup": {  
    "Name": "AthenaAdmin",  
    "State": "ENABLED",  
    "Configuration": {  
      "ResultConfiguration": {  
        "OutputLocation": "s3://awsdoc-example-bucket/"  
      },  
      "EnforceWorkGroupConfiguration": false,  
      "PublishCloudWatchMetricsEnabled": true,  
      "RequesterPaysEnabled": false  
    },  
    "Description": "Workgroup for Athena administrators",  
    "CreationTime": 1573677174.105  
  }  
}
```

有关更多信息，请参阅 Amazon Athena 用户指南中的[管理工作组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetWorkGroup](#)中的。

## list-data-catalogs

以下代码示例演示了如何使用 list-data-catalogs。

AWS CLI

列出在 Athena 注册的数据目录

以下 list-data-catalogs 示例列出了在 Athena 中注册的数据目录。

```
aws athena list-data-catalogs
```

输出：

```
{  
  "DataCatalogsSummary": [  
    {  
      "Name": "AthenaAdmin",  
      "State": "ENABLED",  
      "Configuration": {  
        "ResultConfiguration": {  
          "OutputLocation": "s3://awsdoc-example-bucket/"  
        },  
        "EnforceWorkGroupConfiguration": false,  
        "PublishCloudWatchMetricsEnabled": true,  
        "RequesterPaysEnabled": false  
      },  
      "Description": "Workgroup for Athena administrators",  
      "CreationTime": 1573677174.105  
    }  
  ]  
}
```

```
{
  "CatalogName": "AwsDataCatalog",
  "Type": "GLUE"
},
{
  "CatalogName": "cw_logs_catalog",
  "Type": "LAMBDA"
},
{
  "CatalogName": "cw_metrics_catalog",
  "Type": "LAMBDA"
}
]
```

有关更多信息，请参阅 Amazon Athena [用户 list-data-catalogs 指南中的列出注册目录](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDataCatalogs](#) 中的。

## list-databases

以下代码示例演示了如何使用 list-databases。

### AWS CLI

列出数据目录中的数据库

以下 list-databases 示例列出了 AwsDataCatalog 数据目录中的数据库。

```
aws athena list-databases \
  --catalog-name AwsDataCatalog
```

输出：

```
{
  "DatabaseList": [
    {
      "Name": "default"
    },
    {
      "Name": "mydatabase"
    }
  ]
}
```

```
    "Name": "newdb"
  },
  {
    "Name": "sampledb",
    "Description": "Sample database",
    "Parameters": {
      "CreatedBy": "Athena",
      "EXTERNAL": "TRUE"
    }
  },
  {
    "Name": "webdata"
  }
]
```

有关更多信息，请参阅 Amazon Athena [用户指南中的在目录中列出数据库：列表数据库](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDatabases](#) 中的。

## list-named-queries

以下代码示例演示了如何使用 `list-named-queries`。

### AWS CLI

列出工作组的命名查询

以下 `list-named-queries` 示例列出了 AthenaAdmin 工作组的命名查询。

```
aws athena list-named-queries \
  --work-group AthenaAdmin
```

输出：

```
{
  "NamedQueryIds": [
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
  ]
}
```

有关更多信息，请参阅 Amazon Athena 用户指南中的[使用 Amazon Athena 运行 SQL 查询](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListNamedQueries](#)中的。

## list-query-executions

以下代码示例演示了如何使用 list-query-executions。

### AWS CLI

列出指定工作组中查询的查询 ID

以下 list-query-executions 示例列出了 AthenaAdmin 工作组中最多十个查询 ID。

```
aws athena list-query-executions \  
  --work-group AthenaAdmin \  
  --max-items 10
```

输出：

```
{  
  "QueryExecutionIds": [  
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE11110",  
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",  
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE11114",  
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE11115",  
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE11116",  
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE11117",  
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE11118",  
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE11119"  
  ],  
  "NextToken": "eyJ0ZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxMH0="
```

有关更多信息，请参阅 Amazon Athena 用户指南中的[使用查询结果、输出文件和查询历史记录](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListQueryExecutions](#)中的。

## list-table-metadata

以下代码示例演示了如何使用 list-table-metadata。

## AWS CLI

列出数据目录的指定数据库中表的元数据

以下`list-table-metadata`示例返回数据目录`geography`数据库中最多两个表的元`AwsDataCatalog`数据信息。

```
aws athena list-table-metadata \  
  --catalog-name AwsDataCatalog \  
  --database-name geography \  
  --max-items 2
```

输出：

```
{  
  "TableMetadataList": [  
    {  
      "Name": "country_codes",  
      "CreateTime": 1586553454.0,  
      "TableType": "EXTERNAL_TABLE",  
      "Columns": [  
        {  
          "Name": "country",  
          "Type": "string",  
          "Comment": "geo id"  
        },  
        {  
          "Name": "alpha-2 code",  
          "Type": "string",  
          "Comment": "geo id2"  
        },  
        {  
          "Name": "alpha-3 code",  
          "Type": "string",  
          "Comment": "state name"  
        },  
        {  
          "Name": "numeric code",  
          "Type": "bigint",  
          "Comment": ""  
        },  
        {  
          "Name": "latitude",
```



```

        "Type": "bigint",
        "Comment": "location (latitude)"
    },
    {
        "Name": "longitude",
        "Type": "bigint",
        "Comment": "location (longitude)"
    }
],
"Parameters": {
    "areColumnsQuoted": "false",
    "classification": "csv",
    "columnsOrdered": "true",
    "delimiter": ",",
    "has_encrypted_data": "false",
    "inputformat": "org.apache.hadoop.mapred.TextInputFormat",
    "location": "s3://awsdoc-example-bucket/csv/countrycode",
    "outputformat":
"org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat",
    "serde.param.field.delim": ",",
    "serde.serialization.lib":
"org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe",
    "skip.header.line.count": "1",
    "typeOfData": "file"
}
},
{
    "Name": "county_populations",
    "CreateTime": 1586553446.0,
    "TableType": "EXTERNAL_TABLE",
    "Columns": [
        {
            "Name": "id",
            "Type": "string",
            "Comment": "geo id"
        },
        {
            "Name": "country",

            "Name": "id2",
            "Type": "string",
            "Comment": "geo id2"
        },
    ],
}

```

```

        "Name": "county",
        "Type": "string",
        "Comment": "county name"
    },
    {
        "Name": "state",
        "Type": "string",
        "Comment": "state name"
    },
    {
        "Name": "population estimate 2018",
        "Type": "string",
        "Comment": ""
    }
],
"Parameters": {
    "areColumnsQuoted": "false",
    "classification": "csv",
    "columnsOrdered": "true",
    "delimiter": ",",
    "has_encrypted_data": "false",
    "inputformat": "org.apache.hadoop.mapred.TextInputFormat",
    "location": "s3://awsdoc-example-bucket/csv/CountyPopulation",
    "outputformat":
"org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat",
    "serde.param.field.delim": ",",
    "serde.serialization.lib":
"org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe",
    "skip.header.line.count": "1",
    "typeOfData": "file"
}
}
],
"NextToken": "eyJ0ZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ=="
}

```

有关更多信息，请参阅 Amazon Athena [a 用户指南 list-table-metadata](#) 中的 [显示数据库中所有表的元数据](#) 。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTableMetadata](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

#### 示例 1：列出工作组的标签

以下 `list-tags-for-resource` 示例列出了 `Data_Analyst_Group` 工作组的标签。

```
aws athena list-tags-for-resource \  
  --resource-arn arn:aws:athena:us-west-2:111122223333:workgroup/  
Data_Analyst_Group
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "Division",  
      "Value": "West"  
    },  
    {  
      "Key": "Team",  
      "Value": "Big Data"  
    },  
    {  
      "Key": "Location",  
      "Value": "Seattle"  
    }  
  ]  
}
```

#### 示例 2：列出数据目录的标签

以下 `list-tags-for-resource` 示例列出了 `dynamo_db_catalog` 数据目录的标签。

```
aws athena list-tags-for-resource \  
  --resource-arn arn:aws:athena:us-west-2:111122223333:datacatalog/  
dynamo_db_catalog
```

输出：

```
{
  "Tags": [
    {
      "Key": "Division",
      "Value": "Mountain"
    },
    {
      "Key": "Organization",
      "Value": "Retail"
    },
    {
      "Key": "Product_Line",
      "Value": "Shoes"
    },
    {
      "Key": "Location",
      "Value": "Denver"
    }
  ]
}
```

有关更多信息，请参阅 Amazon Athena [a 用户 list-tags-for-resource 指南中的列出资源的标签](#) 。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## list-work-groups

以下代码示例演示了如何使用 list-work-groups。

### AWS CLI

列出工作组

以下 list-work-groups 示例列出了当前账户中的工作组。

```
aws athena list-work-groups
```

输出：

```
{
  "WorkGroups": [
    {
```

```
    "Name": "Data_Analyst_Group",
    "State": "ENABLED",
    "Description": "",
    "CreationTime": 1578006683.016
  },
  {
    "Name": "AthenaAdmin",
    "State": "ENABLED",
    "Description": "",
    "CreationTime": 1573677174.105
  },
  {
    "Name": "primary",
    "State": "ENABLED",
    "Description": "",
    "CreationTime": 1567465222.723
  }
]
```

有关更多信息，请参阅 Amazon Athena 用户指南中的[管理工作组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListWorkGroups](#)中的。

## start-query-execution

以下代码示例演示了如何使用 start-query-execution。

### AWS CLI

示例 1：在工作组中对指定数据库和数据目录中的指定表运行查询

以下 start-query-execution 示例使用 AthenaAdmin 工作组对 AwsDataCatalog 数据目录 cflogsdatabase 中的 cloudfront\_logs 表运行查询。

```
aws athena start-query-execution \  
  --query-string "select date, location, browser, uri, status from cloudfront_logs \  
  where method = 'GET' and status = 200 and location like 'SF0%' limit 10" \  
  --work-group "AthenaAdmin" \  
  --query-execution-context Database=cflogsdatabase,Catalog=AwsDataCatalog
```

输出：

```
{
  "QueryExecutionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

有关更多信息，请参阅 Amazon Athena 用户指南中的[使用 Amazon Athena 运行 SQL 查询](#)。

示例 2：运行使用指定工作组在指定数据目录中创建数据库的查询

以下start-query-execution示例使用AthenaAdmin工作组在默认数据目录newdbAwsDataCatalog中创建数据库。

```
aws athena start-query-execution \
  --query-string "create database if not exists newdb" \
  --work-group "AthenaAdmin"
```

输出：

```
{
  "QueryExecutionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11112"
}
```

有关更多信息，请参阅 Amazon Athena 用户指南中的[使用 Amazon Athena 运行 SQL 查询](#)。

示例 3：运行查询以在指定数据库和数据目录中的表上创建视图

以下start-query-execution示例使用对cloudfront\_logs表的SELECT语句cflogsdatabase来创建视图cf10。

```
aws athena start-query-execution \
  --query-string "CREATE OR REPLACE VIEW cf10 AS SELECT * FROM cloudfront_logs
  limit 10" \
  --query-execution-context Database=cflogsdatabase
```

输出：

```
{
  "QueryExecutionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11113"
}
```

有关更多信息，请参阅 Amazon Athena 用户指南中的[使用 Amazon Athena 运行 SQL 查询](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartQueryExecution](#)中的。

## stop-query-execution

以下代码示例演示了如何使用 stop-query-execution。

### AWS CLI

停止正在运行的查询

以下stop-query-execution示例停止具有指定查询 ID 的查询。

```
aws athena stop-query-execution \  
  --query-execution-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Athena 用户指南中的[使用 Amazon Athena 运行 SQL 查询](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopQueryExecution](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

要将标签添加到资源中

以下tag-resource示例向dynamo\_db\_catalog数据目录添加了三个标签。

```
aws athena tag-resource \  
  --resource-arn arn:aws:athena:us-west-2:111122223333:datacatalog/  
dynamo_db_catalog \  
  --tags Key=Organization,Value=Retail Key=Division,Value=Mountain  
Key=Product_Line,Value=Shoes Key=Location,Value=Denver
```

此命令不生成任何输出。要查看结果，请使用aws athena list-tags-for-resource --resource-arn arn:aws:athena:us-west-2:111122223333:datacatalog/dynamo\_db\_catalog。

有关更多信息，请参阅 Amazon Athena [用户指南中的为资源添加标签](#)：标签资源。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从资源中移除标签

以下untag-resource示例从dynamo\_db\_catalog数据目录资源中移除Specialization和Focus键及其关联值。

```
aws athena untag-resource \  
  --resource-arn arn:aws:athena:us-west-2:111122223333:datacatalog/  
dynamo_db_catalog \  
  --tag-keys Specialization Focus
```

此命令不生成任何输出。要查看结果，请使用list-tags-for-resource命令。

有关更多信息，请参阅 Amazon Athena [a 用户指南中的从资源中移除标签：取消标记资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-data-catalog

以下代码示例演示了如何使用 update-data-catalog。

### AWS CLI

更新数据目录

以下update-data-catalog示例更新了 Lambda 函数和cw\_logs\_catalog数据目录的描述。

```
aws athena update-data-catalog \  
  --name cw_logs_catalog \  
  --type LAMBDA \  
  --description "New CloudWatch Logs Catalog" \  
  --function=arn:aws:lambda:us-west-2:111122223333:function:new_cw_logs_lambda
```

此命令不生成任何输出。要查看结果，请使用aws athena get-data-catalog --name cw\_logs\_catalog。



有关更多信息，请参阅 Amazon Athena 用户指南 [update-data-catalog](#) 中的 [更新目录](#) 。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateDataCatalog](#) 中的。

## update-work-group

以下代码示例演示了如何使用 `update-work-group`。

### AWS CLI

#### 更新工作组

以下 `update-work-group` 示例禁用了 `Data_Analyst_Group` 工作组。用户无法在禁用的工作组中运行或创建查询，但仍可以查看指标、数据使用限制控制、工作组设置、查询历史记录和保存的查询。

```
aws athena update-work-group \  
  --work-group Data_Analyst_Group \  
  --state DISABLED
```

此命令不生成任何输出。要验证状态的变化，请使用 `aws athena get-work-group --work-group Data_Analyst_Group` 并检查输出中的 `State` 属性。

有关更多信息，请参阅 Amazon Athena 用户指南中的 [管理工作组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateWorkGroup](#) 中的。

## 使用的 Auto Scaling 示例 AWS CLI

以下代码示例向您展示了如何使用与 Auto Scaling AWS Command Line Interface 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### **attach-instances**

以下代码示例演示了如何使用 `attach-instances`。

#### AWS CLI

将实例附加到 Auto Scaling 组

此示例将指定的实例附加到指定的 Auto Scaling 组。

```
aws autoscaling attach-instances \  
  --instance-ids i-061c63c5eb45f0416 \  
  --auto-scaling-group-name my-asg
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachInstances](#)中的。

### **attach-load-balancer-target-groups**

以下代码示例演示了如何使用 `attach-load-balancer-target-groups`。

#### AWS CLI

将目标组附加到自动扩缩组

此示例将指定目标组附加到指定的自动扩缩组。

```
aws autoscaling attach-load-balancer-target-groups \  
  --auto-scaling-group-name my-asg \  
  --target-group-arns arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的 [Elastic Load Balancing 和 Amazon EC2 Auto Scaling](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachLoadBalancerTargetGroups](#)中的。

## attach-load-balancers

以下代码示例演示了如何使用 attach-load-balancers。

### AWS CLI

将 Classic Load Balancer 连接到 Auto Scaling 组

此示例将指定的 Classic Load Balancer 附加到指定的 Auto Scaling 组。

```
aws autoscaling attach-load-balancers \  
  --load-balancer-names my-load-balancer \  
  --auto-scaling-group-name my-asg
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的 [Elastic Load Balancing 和 Amazon EC2 Auto Scaling](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachLoadBalancers](#)中的。

## cancel-instance-refresh

以下代码示例演示了如何使用 cancel-instance-refresh。

### AWS CLI

取消实例刷新

以下cancel-instance-refresh示例取消了指定 Auto Scaling 组正在进行的实例刷新。

```
aws autoscaling cancel-instance-refresh \  
  --auto-scaling-group-name my-asg
```

输出：

```
{  
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"  
}
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[取消实例刷新](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelInstanceRefresh](#)中的。

## complete-lifecycle-action

以下代码示例演示了如何使用 `complete-lifecycle-action`。

### AWS CLI

#### 完成生命周期操作

此示例通知 Amazon EC2 Auto Scaling 指定的生命周期操作已完成，因此它可以完成实例的启动或终止。

```
aws autoscaling complete-lifecycle-action \  
  --lifecycle-hook-name my-launch-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-action-result CONTINUE \  
  --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 [Amazon EC2 Auto Scaling 生命周期挂钩](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CompleteLifecycleAction](#) 中的。

## create-auto-scaling-group

以下代码示例演示了如何使用 `create-auto-scaling-group`。

### AWS CLI

#### 示例 1：创建自动扩缩组

以下 `create-auto-scaling-group` 示例在区域内多个可用区中的子网中创建自动扩缩组。实例以指定启动模板的默认版本启动。请注意，大多数其他设置都使用默认值，例如，终止策略和运行状况检查配置。

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateId=lt-1234567890abcde12 \  
  --min-size 1 \  
  --max-size 5 \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[自动扩缩组](#)。

示例 2：附加应用程序负载均衡器、网络负载均衡器或网关负载均衡器

此示例为支持预期流量的负载均衡器指定目标组的 ARN。运行状况检查类型指定 ELB，以便在 Elastic Load Balancing 报告实例运行状况不佳时，自动扩缩组将取代它。该命令还定义了以 600 秒为单位的运行状况检查宽限期。宽限期有助于防止新启动的实例过早终止。

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateId=lt-1234567890abcde12 \  
  --target-group-arns arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/943f017f100becff \  
  --health-check-type ELB \  
  --health-check-grace-period 600 \  
  --min-size 1 \  
  --max-size 5 \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[Elastic Load Balancing 和 Amazon EC2 Auto Scaling](#)。

示例 3：指定置放群组，并使用最新版本的启动模板

此示例将实例启动到单个可用区中的置放群组。这对于具有 HPC 工作负载的低延迟群组很有用。此示例还将指定群组的最小大小、最大大小和所需容量。

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest' \  
  --min-size 1 \  
  --max-size 5 \  
  --desired-capacity 3 \  
  --placement-group my-placement-group \  
  --vpc-zone-identifier "subnet-6194ea3b"
```

此命令不生成任何输出。

有关更多信息，请参阅《适用于 Linux 实例的 Amazon EC2 用户指南》中的[放置组](#)。

示例 4：指定单个实例自动扩缩组，并使用特定版本的启动模板

此示例将创建一个自动扩缩组，并将其最小和最大容量均设置为 1 以强制运行一个实例。该命令还指定了启动模板的 v1，其中指定了现有 ENI 的 ID。当您使用为 eth0 指定现有 ENI 的启动模板时，必须为自动扩缩组指定与网络接口匹配的可用区，而无需在请求中同时指定子网 ID。

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg-single-instance \  
  --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='1' \  
  --min-size 1 \  
  --max-size 1 \  
  --availability-zones us-west-2a
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[自动扩缩组](#)。

示例 5：指定不同的终止策略

此示例使用启动配置创建自动扩缩组，并将终止策略设置为首先终止最旧的实例。该命令还将标签应用于该组及其实例，其密钥为 Role，值为 WebServer。

```
aws autoscaling create-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-configuration-name my-lc \  
  --min-size 1 \  
  --max-size 5 \  
  --termination-policies "OldestInstance" \  
  --tags "ResourceId=my-asg,ResourceType=auto-scaling-  
group,Key=Role,Value=WebServer,PropagateAtLaunch=true" \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[使用 Amazon EC2 Auto Scaling 终止策略](#)。

示例 6：指定启动生命周期挂钩

此示例将使用生命周期挂钩创建一个自动扩缩组，该挂钩支持在实例启动时的自定义操作。

```
aws autoscaling create-auto-scaling-group \  
  --cli-input-json file://~/config.json
```

config.json 文件的内容：

```
{  
  "AutoScalingGroupName": "my-asg",  
  "LaunchTemplate": {  
    "LaunchTemplateId": "lt-1234567890abcde12"  
  },  
  "LifecycleHookSpecificationList": [{  
    "LifecycleHookName": "my-launch-hook",  
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",  
    "NotificationTargetARN": "arn:aws:sqs:us-west-2:123456789012:my-sqs-queue",  
    "RoleARN": "arn:aws:iam::123456789012:role/my-notification-role",  
    "NotificationMetadata": "SQS message metadata",  
    "HeartbeatTimeout": 4800,  
    "DefaultResult": "ABANDON"  
  }],  
  "MinSize": 1,  
  "MaxSize": 5,  
  "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",  
  "Tags": [{  
    "ResourceType": "auto-scaling-group",  
    "ResourceId": "my-asg",  
    "PropagateAtLaunch": true,  
    "Value": "test",  
    "Key": "environment"  
  }]  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 [Amazon EC2 Auto Scaling 生命周期挂钩](#)。

#### 示例 7：指定终止生命周期挂钩

此示例将使用生命周期挂钩创建一个自动扩缩组，该挂钩支持在实例终止时的自定义操作。

```
aws autoscaling create-auto-scaling-group \  
  --cli-input-json file://~/config.json
```

config.json 的内容：

```
{
  "AutoScalingGroupName": "my-asg",
  "LaunchTemplate": {
    "LaunchTemplateId": "lt-1234567890abcde12"
  },
  "LifecycleHookSpecificationList": [{
    "LifecycleHookName": "my-termination-hook",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
    "HeartbeatTimeout": 120,
    "DefaultResult": "CONTINUE"
  }],
  "MinSize": 1,
  "MaxSize": 5,
  "TargetGroupARNs": [
    "arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-
    targets/73e2d6bc24d8a067"
  ],
  "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 [Amazon EC2 Auto Scaling 生命周期挂钩](#)。

示例 8：指定自定义终止策略

此示例将创建一个自动扩缩组，该组指定自定义 Lambda 函数终止策略，以指示 Amazon EC2 Auto Scaling 哪些实例可以安全地在横向缩减时终止。

```
aws autoscaling create-auto-scaling-group \
  --auto-scaling-group-name my-asg-single-instance \
  --launch-template LaunchTemplateName=my-template-for-auto-scaling \
  --min-size 1 \
  --max-size 5 \
  --termination-policies "arn:aws:lambda:us-
  west-2:123456789012:function:HelloFunction:prod" \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不生成任何输出。



有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[通过 Lambda 创建自定义终止策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateAutoScalingGroup](#) 中的。

## create-launch-configuration

以下代码示例演示了如何使用 create-launch-configuration。

### AWS CLI

#### 示例 1：创建启动配置

此示例创建了一个简单的启动配置。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[创建启动配置](#)。

#### 示例 2：使用安全组、key pair 和 bootstrapping 脚本创建启动配置

此示例使用用户数据中包含的安全组、key pair 和引导脚本创建启动配置。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --security-groups sg-eb2af88example \  
  --key-name my-key-pair \  
  --user-data file://myuserdata.txt
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[创建启动配置](#)。

#### 示例 3：使用 IAM 角色创建启动配置

此示例使用 IAM 角色的实例配置文件名称创建启动配置。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --iam-instance-profile my-autoscaling-role
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[在 Amazon EC2 实例上运行的应用程序的 IAM 角色](#)。

#### 示例 4：创建启用详细监控的启动配置

此示例创建了启用 EC2 详细监控的启动配置，该配置会 CloudWatch 在 1 分钟内将 EC2 指标发送到。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --instance-monitoring Enabled=true
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的配置自动扩展[实例的监控](#)。

#### 示例 5：创建启动竞价型实例的启动配置

此示例创建了使用竞价型实例作为唯一购买选项的启动配置。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --spot-price "0.50"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[请求竞价型实例](#)。

#### 示例 6：使用 EC2 实例创建启动配置

此示例基于现有实例的属性创建启动配置。它会覆盖放置租约以及是否通过包含 `--placement-tenancy` 和 `--no-associate-public-ip-address` 选项来设置公有 IP 地址。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc-from-instance \  
  --instance-id i-0123a456700123456 \  
  --instance-type m5.large \  
  --no-associate-public-ip-address \  
  --placement-tenancy dedicated
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的使用 EC2 [实例创建启动配置](#)。

示例 7：使用块储存设备映射为 Amazon EBS 卷创建启动配置

此示例为设备名称 `/dev/sdh` 和卷大小为 20 的 Amazon EBS gp3 卷创建了带有块储存设备映射的启动配置。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --block-device-mappings '[{"DeviceName":"/dev/sdh","Ebs":  
{ "VolumeSize":20,"VolumeType":"gp3" } } ]'
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling API 参考 [中的 EBS](#)。

有关引用 JSON 格式参数值的语法的消息，请参阅《AWS 命令行界面用户 [指南](#)》中的 [CL AWS I 中对字符串使用引号](#)。

示例 8：为实例存储卷创建带有块储存设备映射的启动配置

此示例使用设备名称 `ephemeral1` 作为实例存储卷创建启动配置 `/dev/sdc`。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --block-device-mappings '[{"DeviceName":"/dev/sdc","VirtualName":"ephemeral1"}]'
```

此命令不生成任何输出。

有关更多信息，请参阅 [BlockDeviceMapping](#) Amazon EC2 Auto Scaling API 参考中的。

有关引用 JSON 格式参数值的语法的消息，请参阅《AWS 命令行界面用户指南》中的 [CL AWS I 中对字符串使用引号](#)。

示例 9：创建启动配置并在启动时禁止连接块储存设备

此示例创建了一个启动配置，用于抑制由 AMI 的块储存设备映射指定的块储存设备（例如，`/dev/sdf`）。

```
aws autoscaling create-launch-configuration \  
  --launch-configuration-name my-lc \  
  --image-id ami-04d5cc9b88example \  
  --instance-type m5.large \  
  --block-device-mappings '[{"DeviceName":"/dev/sdf","NoDevice":""}]'
```

此命令不生成任何输出。

有关更多信息，请参阅 [BlockDeviceMapping](#) Amazon EC2 Auto Scaling API 参考中的。

有关引用 JSON 格式参数值的语法的消息，请参阅《AWS 命令行界面用户指南》中的 [CL AWS I 中对字符串使用引号](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateLaunchConfiguration](#) 中的。

## create-or-update-tags

以下代码示例演示了如何使用 `create-or-update-tags`。

### AWS CLI

为 Auto Scaling 组创建或更新标签

此示例向指定的 Auto Scaling 组添加了两个标签。

```
aws autoscaling create-or-update-tags \  
  --tags ResourceId=my-asg,ResourceType=auto-scaling-  
group,Key=Role,Value=WebServer,PropagateAtLaunch=true ResourceId=my-  
asg,ResourceType=auto-scaling-group,Key=Dept,Value=Research,PropagateAtLaunch=true
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的为 Auto Scaling [组和实例添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateOrUpdateTags](#)中的。

## delete-auto-scaling-group

以下代码示例演示了如何使用 delete-auto-scaling-group。

### AWS CLI

示例 1：删除指定的自动扩缩组

此示例将删除指定的自动扩缩组。

```
aws autoscaling delete-auto-scaling-group \  
  --auto-scaling-group-name my-asg
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[删除自动扩缩基础设施](#)。

示例 2：强制删除指定的自动扩缩组

要在不等待自动扩缩组中的实例终止的情况下删除该组，请使用 --force-delete 选项。

```
aws autoscaling delete-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --force-delete
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[删除自动扩缩基础设施](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAutoScalingGroup](#)中的。

## delete-launch-configuration

以下代码示例演示了如何使用 delete-launch-configuration。

### AWS CLI

删除启动配置

此示例删除了指定的启动配置。

```
aws autoscaling delete-launch-configuration \  
  --launch-configuration-name my-launch-config
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[删除自动扩缩基础设施](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLaunchConfiguration](#)中的。

## delete-lifecycle-hook

以下代码示例演示了如何使用 delete-lifecycle-hook。

### AWS CLI

#### 删除生命周期挂钩

此示例删除了指定的生命周期挂钩。

```
aws autoscaling delete-lifecycle-hook \  
  --lifecycle-hook-name my-lifecycle-hook \  
  --auto-scaling-group-name my-asg
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLifecycleHook](#)中的。

## delete-notification-configuration

以下代码示例演示了如何使用 delete-notification-configuration。

### AWS CLI

#### 删除 Auto Scaling 通知

此示例从指定的 Auto Scaling 组中删除指定的通知。

```
aws autoscaling delete-notification-configuration \  
  --auto-scaling-group-name my-asg \  
  --notification-configuration-name my-notification-configuration
```

```
--topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[删除通知配置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNotificationConfiguration](#)中的。

## delete-policy

以下代码示例演示了如何使用 delete-policy。

### AWS CLI

删除扩展策略

此示例删除了指定的扩展策略。

```
aws autoscaling delete-policy \  
  --auto-scaling-group-name my-asg \  
  --policy-name alb1000-target-tracking-scaling-policy
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePolicy](#)中的。

## delete-scheduled-action

以下代码示例演示了如何使用 delete-scheduled-action。

### AWS CLI

从 Auto Scaling 组中删除计划操作

此示例从指定的 Auto Scaling 组中删除指定的计划操作。

```
aws autoscaling delete-scheduled-action \  
  --auto-scaling-group-name my-asg \  
  --scheduled-action-name my-scheduled-action
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteScheduledAction](#)中的。

## delete-tags

以下代码示例演示了如何使用 delete-tags。

### AWS CLI

从 Auto Scaling 组中删除标签

此示例从指定的 Auto Scaling 组中删除指定的标签。

```
aws autoscaling delete-tags \  
  --tags ResourceId=my-asg,ResourceType=auto-scaling-group,Key=Dept,Value=Research
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的为 Auto Scaling [组和实例添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteTags](#) 中的。

## delete-warm-pool

以下代码示例演示了如何使用 delete-warm-pool。

### AWS CLI

示例 1：删除温池

以下示例删除了指定 Auto Scaling 组的温池。

```
aws autoscaling delete-warm-pool \  
  --auto-scaling-group-name my-asg
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 [Amazon EC2 Auto Scaling 的温水池](#)。

示例 2：强制删除温池

要在不等待其实例终止的情况下删除温池，请使用 --force-delete 选项。



```
aws autoscaling delete-warm-pool \  
  --auto-scaling-group-name my-asg \  
  --force-delete
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 [Amazon EC2 Auto Scaling 的温水池](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteWarmPool](#)中的。

## describe-account-limits

以下代码示例演示了如何使用 describe-account-limits。

### AWS CLI

描述您的 Amazon EC2 Auto Scaling 账户限制

此示例描述了您的 AWS 账户的 Amazon EC2 Auto Scaling 限制。

```
aws autoscaling describe-account-limits
```

输出：

```
{  
  "NumberOfLaunchConfigurations": 5,  
  "MaxNumberOfLaunchConfigurations": 100,  
  "NumberOfAutoScalingGroups": 3,  
  "MaxNumberOfAutoScalingGroups": 20  
}
```

有关更多信息，请参阅 [Amazon EC2 Auto Scaling 用户指南中的 Amazon EC2 Auto Scaling 服务配额](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAccountLimits](#)中的。

## describe-adjustment-types

以下代码示例演示了如何使用 describe-adjustment-types。

## AWS CLI

### 描述可用的缩放调整类型

此示例描述了可用的调整类型。

```
aws autoscaling describe-adjustment-types
```

输出：

```
{
  "AdjustmentTypes": [
    {
      "AdjustmentType": "ChangeInCapacity"
    },
    {
      "AdjustmentType": "ExactCapacity"
    },
    {
      "AdjustmentType": "PercentChangeInCapacity"
    }
  ]
}
```

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[扩展调整类型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAdjustmentTypes](#)中的。

## describe-auto-scaling-groups

以下代码示例演示了如何使用 describe-auto-scaling-groups。

### AWS CLI

#### 示例 1：描述指定的自动扩缩组

此示例将描述指定的自动扩缩组。

```
aws autoscaling describe-auto-scaling-groups \
  --auto-scaling-group-name my-asg
```

输出：

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "my-asg",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-
a11a-7b0acd480f03:autoScalingGroupName/my-asg",
      "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "1",
        "LaunchTemplateId": "lt-1234567890abcde12"
      },
      "MinSize": 0,
      "MaxSize": 1,
      "DesiredCapacity": 1,
      "DefaultCooldown": 300,
      "AvailabilityZones": [
        "us-west-2a",
        "us-west-2b",
        "us-west-2c"
      ],
      "LoadBalancerNames": [],
      "TargetGroupARNs": [],
      "HealthCheckType": "EC2",
      "HealthCheckGracePeriod": 0,
      "Instances": [
        {
          "InstanceId": "i-06905f55584de02da",
          "InstanceType": "t2.micro",
          "AvailabilityZone": "us-west-2a",
          "HealthStatus": "Healthy",
          "LifecycleState": "InService",
          "ProtectedFromScaleIn": false,
          "LaunchTemplate": {
            "LaunchTemplateName": "my-launch-template",
            "Version": "1",
            "LaunchTemplateId": "lt-1234567890abcde12"
          }
        }
      ],
      "CreatedTime": "2023-10-28T02:39:22.152Z",
      "SuspendedProcesses": [],
    }
  ]
}
```

```
        "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
        "EnabledMetrics": [],
        "Tags": [],
        "TerminationPolicies": [
            "Default"
        ],
        "NewInstancesProtectedFromScaleIn": false,
        "ServiceLinkedRoleARN": "arn",
        "TrafficSources": []
    }
]
}
```

### 示例 2：描述前 100 个指定的自动扩缩组

此示例将描述指定的自动扩缩组。它允许您指定最多 100 个组名称。

```
aws autoscaling describe-auto-scaling-groups \
  --max-items 100 \
  --auto-scaling-group-name "group1" "group2" "group3" "group4"
```

有关输出示例，请参阅示例 1。

### 示例 3：描述指定区域中的自动扩缩组

此示例将描述指定区域中的自动扩缩组（最多 75 个组）。

```
aws autoscaling describe-auto-scaling-groups \
  --max-items 75 \
  --region us-east-1
```

有关输出示例，请参阅示例 1。

### 示例 4：描述指定数量的自动扩缩组

要返回特定数量的自动扩缩组，请使用 `--max-items` 选项。

```
aws autoscaling describe-auto-scaling-groups \
  --max-items 1
```

有关输出示例，请参阅示例 1。

如果输出包含 `NextToken` 字段，则可描述更多组。要获取其他组，请在后续调用中使用此字段的值和 `--starting-token` 选项，如下所示。

```
aws autoscaling describe-auto-scaling-groups \  
  --starting-token Z3M3LMPEXAMPLE
```

有关输出示例，请参阅示例 1。

#### 示例 5：描述使用启动配置的 Auto Scaling 群组

此示例使用 `--query` 选项来描述使用启动配置的 Auto Scaling 组。

```
aws autoscaling describe-auto-scaling-groups \  
  --query 'AutoScalingGroups[?LaunchConfigurationName!=`null`]'
```

输出：

```
[  
  {  
    "AutoScalingGroupName": "my-asg",  
    "AutoScalingGroupARN": "arn:aws:autoscaling:us-  
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-  
a11a-7b0acd480f03:autoScalingGroupName/my-asg",  
    "LaunchConfigurationName": "my-lc",  
    "MinSize": 0,  
    "MaxSize": 1,  
    "DesiredCapacity": 1,  
    "DefaultCooldown": 300,  
    "AvailabilityZones": [  
      "us-west-2a",  
      "us-west-2b",  
      "us-west-2c"  
    ],  
    "LoadBalancerNames": [],  
    "TargetGroupARNs": [],  
    "HealthCheckType": "EC2",  
    "HealthCheckGracePeriod": 0,  
    "Instances": [  
      {  
        "InstanceId": "i-088c57934a6449037",  
        "InstanceType": "t2.micro",  
        "AvailabilityZone": "us-west-2c",
```

```
        "HealthStatus": "Healthy",
        "LifecycleState": "InService",
        "LaunchConfigurationName": "my-lc",
        "ProtectedFromScaleIn": false
    }
],
"CreatedTime": "2023-10-28T02:39:22.152Z",
"SuspendedProcesses": [],
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
"EnabledMetrics": [],
"Tags": [],
"TerminationPolicies": [
    "Default"
],
"NewInstancesProtectedFromScaleIn": false,
"ServiceLinkedRoleARN": "arn",
"TrafficSources": []
}
]
```

有关更多信息，请参阅《AWS 命令行界面用户指南》中的[筛选 AWS CLI 输出](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAutoScalingGroups](#) 中的。

## describe-auto-scaling-instances

以下代码示例演示了如何使用 describe-auto-scaling-instances。

### AWS CLI

示例 1：描述一个或多个实例

此示例将描述指定的实例。

```
aws autoscaling describe-auto-scaling-instances \
  --instance-ids i-06905f55584de02da
```

输出：

```
{
  "AutoScalingInstances": [
    {
      "InstanceId": "i-06905f55584de02da",
```

```

    "InstanceType": "t2.micro",
    "AutoScalingGroupName": "my-asg",
    "AvailabilityZone": "us-west-2b",
    "LifecycleState": "InService",
    "HealthStatus": "HEALTHY",
    "ProtectedFromScaleIn": false,
    "LaunchTemplate": {
      "LaunchTemplateId": "lt-1234567890abcde12",
      "LaunchTemplateName": "my-launch-template",
      "Version": "1"
    }
  ]
}

```

### 示例 2：描述一个或多个实例

此示例使用 `--max-items` 选项来指定通过此调用返回多少个实例。

```
aws autoscaling describe-auto-scaling-instances \
  --max-items 1
```

如果输出包含 `NextToken` 字段，可返回更多实例。要获取其他实例，请在后续调用中使用此字段的值和 `--starting-token` 选项，如下所示。

```
aws autoscaling describe-auto-scaling-instances \
  --starting-token Z3M3LMPEXAMPLE
```

有关输出示例，请参阅示例 1。

### 示例 3：描述使用启动配置的实例

此示例使用 `--query` 选项来描述使用启动配置的实例。

```
aws autoscaling describe-auto-scaling-instances \
  --query 'AutoScalingInstances[?LaunchConfigurationName!=`null`]'
```

输出：

```
[
  {
```

```
    "InstanceId": "i-088c57934a6449037",
    "InstanceType": "t2.micro",
    "AutoScalingGroupName": "my-asg",
    "AvailabilityZone": "us-west-2c",
    "LifecycleState": "InService",
    "HealthStatus": "HEALTHY",
    "LaunchConfigurationName": "my-lc",
    "ProtectedFromScaleIn": false
  }
]
```

有关更多信息，请参阅《AWS 命令行界面用户指南》中的[筛选 AWS CLI 输出](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAutoScalingInstances](#)中的。

## describe-auto-scaling-notification-types

以下代码示例演示了如何使用 describe-auto-scaling-notification-types。

### AWS CLI

描述可用的通知类型

此示例描述了可用的通知类型。

```
aws autoscaling describe-auto-scaling-notification-types
```

输出：

```
{
  "AutoScalingNotificationTypes": [
    "autoscaling:EC2_INSTANCE_LAUNCH",
    "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
    "autoscaling:EC2_INSTANCE_TERMINATE",
    "autoscaling:EC2_INSTANCE_TERMINATE_ERROR",
    "autoscaling:TEST_NOTIFICATION"
  ]
}
```

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的 [Auto Scaling 组扩展时获取 Amazon SNS 通知](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAutoScalingNotificationTypes](#)中的。



## describe-instance-refreshes

以下代码示例演示了如何使用 describe-instance-refreshes。

### AWS CLI

#### 描述实例刷新

以下 describe-instance-refreshes 示例返回指定 Auto Scaling 组的所有实例刷新请求的描述，包括状态消息和（如果有）状态原因。

```
aws autoscaling describe-instance-refreshes \  
  --auto-scaling-group-name my-asg
```

输出：

```
{  
  "InstanceRefreshes": [  
    {  
      "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b",  
      "AutoScalingGroupName": "my-asg",  
      "Status": "InProgress",  
      "StatusReason": "Waiting for instances to warm up before continuing. For  
example: 0e69cc3f05f825f4f is warming up.",  
      "EndTime": "2023-03-23T16:42:55Z",  
      "PercentageComplete": 0,  
      "InstancesToUpdate": 0,  
      "Preferences": {  
        "MinHealthyPercentage": 100,  
        "InstanceWarmup": 300,  
        "CheckpointPercentages": [  
          50  
        ],  
        "CheckpointDelay": 3600,  
        "SkipMatching": false,  
        "AutoRollback": true,  
        "ScaleInProtectedInstances": "Ignore",  
        "StandbyInstances": "Ignore"  
      }  
    },  
    {  
      "InstanceRefreshId": "dd7728d0-5bc4-4575-96a3-1b2c52bf8bb1",  
      "AutoScalingGroupName": "my-asg",
```

```

        "Status": "Successful",
        "EndTime": "2022-06-02T16:53:37Z",
        "PercentageComplete": 100,
        "InstancesToUpdate": 0,
    "Preferences": {
        "MinHealthyPercentage": 90,
        "InstanceWarmup": 300,
        "SkipMatching": true,
        "AutoRollback": true,
        "ScaleInProtectedInstances": "Ignore",
        "StandbyInstances": "Ignore"
    }
}
]
}

```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[检查实例刷新状态](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeInstanceRefreshes](#) 中的。

## describe-launch-configurations

以下代码示例演示了如何使用 describe-launch-configurations。

### AWS CLI

示例 1：描述指定的启动配置

此示例描述了指定的启动配置。

```
aws autoscaling describe-launch-configurations \
  --launch-configuration-names my-launch-config
```

输出：

```

{
  "LaunchConfigurations": [
    {
      "LaunchConfigurationName": "my-launch-config",
      "LaunchConfigurationARN": "arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:98d3b196-4cf9-4e88-8ca1-8547c24ced8b:launchConfigura
my-launch-config",

```

```
    "ImageId": "ami-0528a5175983e7f28",
    "KeyName": "my-key-pair-uswest2",
    "SecurityGroups": [
      "sg-05eaec502fcdadc2e"
    ],
    "ClassicLinkVPCSecurityGroups": [],
    "UserData": "",
    "InstanceType": "t2.micro",
    "KernelId": "",
    "RamdiskId": "",
    "BlockDeviceMappings": [
      {
        "DeviceName": "/dev/xvda",
        "Ebs": {
          "SnapshotId": "snap-06c1606ba5ca274b1",
          "VolumeSize": 8,
          "VolumeType": "gp2",
          "DeleteOnTermination": true,
          "Encrypted": false
        }
      }
    ],
    "InstanceMonitoring": {
      "Enabled": true
    },
    "CreatedTime": "2020-10-28T02:39:22.321Z",
    "EbsOptimized": false,
    "AssociatePublicIpAddress": true,
    "MetadataOptions": {
      "HttpTokens": "required",
      "HttpPutResponseHopLimit": 1,
      "HttpEndpoint": "disabled"
    }
  }
]
}
```

## 示例 2：描述指定数量的启动配置

要返回特定数量的启动配置，请使用 `--max-items` 选项。

```
aws autoscaling describe-launch-configurations \
  --max-items 1
```

如果输出包含字NextToken段，则还有更多启动配置。要获取其他启动配置，请在后续调用中使用此字段的值和--starting-token选项，如下所示。

```
aws autoscaling describe-launch-configurations \  
  --starting-token Z3M3LMPEXAMPLE
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLaunchConfigurations](#)中的。

## describe-lifecycle-hook-types

以下代码示例演示了如何使用 describe-lifecycle-hook-types。

### AWS CLI

描述可用的生命周期挂钩类型

此示例描述了可用的生命周期挂钩类型。

```
aws autoscaling describe-lifecycle-hook-types
```

输出：

```
{  
  "LifecycleHookTypes": [  
    "autoscaling:EC2_INSTANCE_LAUNCHING",  
    "autoscaling:EC2_INSTANCE_TERMINATING"  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLifecycleHookTypes](#)中的。

## describe-lifecycle-hooks

以下代码示例演示了如何使用 describe-lifecycle-hooks。

### AWS CLI

描述您的生命周期挂钩

此示例描述了指定 Auto Scaling 组的生命周期挂钩。

```
aws autoscaling describe-lifecycle-hooks \  
  --auto-scaling-group-name my-asg
```

输出：

```
{  
  "LifecycleHooks": [  
    {  
      "GlobalTimeout": 3000,  
      "HeartbeatTimeout": 30,  
      "AutoScalingGroupName": "my-asg",  
      "LifecycleHookName": "my-launch-hook",  
      "DefaultResult": "ABANDON",  
      "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING"  
    },  
    {  
      "GlobalTimeout": 6000,  
      "HeartbeatTimeout": 60,  
      "AutoScalingGroupName": "my-asg",  
      "LifecycleHookName": "my-termination-hook",  
      "DefaultResult": "CONTINUE",  
      "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLifecycleHooks](#)中的。

## describe-load-balancer-target-groups

以下代码示例演示了如何使用 `describe-load-balancer-target-groups`。

### AWS CLI

描述 Auto Scaling 组的负载均衡器目标组

此示例描述了附加到指定 Auto Scaling 组的负载均衡器目标组。

```
aws autoscaling describe-load-balancer-target-groups \  
  --auto-scaling-group-name my-asg
```

输出：

```
{
  "LoadBalancerTargetGroups": [
    {
      "LoadBalancerTargetGroupARN": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
      "State": "Added"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoadBalancerTargetGroups](#)中的。

## describe-load-balancers

以下代码示例演示了如何使用 describe-load-balancers。

### AWS CLI

描述 Auto Scaling 组的经典负载均衡器

此示例描述了指定 Auto Scaling 组的经典负载均衡器。

```
aws autoscaling describe-load-balancers \
  --auto-scaling-group-name my-asg
```

输出：

```
{
  "LoadBalancers": [
    {
      "State": "Added",
      "LoadBalancerName": "my-load-balancer"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoadBalancers](#)中的。

## describe-metric-collection-types

以下代码示例演示了如何使用 describe-metric-collection-types。

## AWS CLI

### 描述可用的指标收集类型

此示例描述了可用的指标收集类型。

```
aws autoscaling describe-metric-collection-types
```

输出：

```
{
  "Metrics": [
    {
      "Metric": "GroupMinSize"
    },
    {
      "Metric": "GroupMaxSize"
    },
    {
      "Metric": "GroupDesiredCapacity"
    },
    {
      "Metric": "GroupInServiceInstances"
    },
    {
      "Metric": "GroupInServiceCapacity"
    },
    {
      "Metric": "GroupPendingInstances"
    },
    {
      "Metric": "GroupPendingCapacity"
    },
    {
      "Metric": "GroupTerminatingInstances"
    },
    {
      "Metric": "GroupTerminatingCapacity"
    },
    {
      "Metric": "GroupStandbyInstances"
    },
    {
```

```

        "Metric": "GroupStandbyCapacity"
    },
    {
        "Metric": "GroupTotalInstances"
    },
    {
        "Metric": "GroupTotalCapacity"
    }
],
"Granularities": [
    {
        "Granularity": "1Minute"
    }
]
}

```

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[自动扩缩组指标](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeMetricCollectionTypes](#)中的。

## describe-notification-configurations

以下代码示例演示了如何使用 describe-notification-configurations。

### AWS CLI

#### 示例 1：描述指定群组的通知配置

此示例描述了指定 Auto Scaling 组的通知配置。

```
aws autoscaling describe-notification-configurations \
  --auto-scaling-group-name my-asg
```

输出：

```

{
  "NotificationConfigurations": [
    {
      "AutoScalingGroupName": "my-asg",
      "NotificationType": "autoscaling:TEST_NOTIFICATION",
      "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
    },
    {

```



```
        "AutoScalingGroupName": "my-asg",
        "NotificationType": "autoscaling:TEST_NOTIFICATION",
        "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
    }
]
}
```

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的 [Auto Scaling 组扩展时获取 Amazon SNS 通知](#)。

示例 1：描述指定数量的通知配置

要返回特定数量的通知配置，请使用 `max-items` 参数。

```
aws autoscaling describe-notification-configurations \
  --auto-scaling-group-name my-auto-scaling-group \
  --max-items 1
```

输出：

```
{
  "NotificationConfigurations": [
    {
      "AutoScalingGroupName": "my-asg",
      "NotificationType": "autoscaling:TEST_NOTIFICATION",
      "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
    },
    {
      "AutoScalingGroupName": "my-asg",
      "NotificationType": "autoscaling:TEST_NOTIFICATION",
      "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
    }
  ]
}
```

如果输出包含 `NextToken` 段，则还有更多通知配置。要获取其他通知配置，请在后续调用中使用此字段的值和 `starting-token` 参数，如下所示。

```
aws autoscaling describe-notification-configurations \
  --auto-scaling-group-name my-asg \
  --starting-token Z3M3LMPEXAMPLE
```

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的 [Auto Scaling 组扩展时获取 Amazon SNS 通知](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeNotificationConfigurations](#) 中的。

## describe-policies

以下代码示例演示了如何使用 describe-policies。

### AWS CLI

示例 1：描述指定群组的伸缩策略

此示例描述了指定 Auto Scaling 组的扩展策略。

```
aws autoscaling describe-policies \  
  --auto-scaling-group-name my-asg
```

输出：

```
{  
  "ScalingPolicies": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "PolicyName": "alb1000-target-tracking-scaling-policy",  
      "PolicyARN": "arn:aws:autoscaling:us-  
west-2:123456789012:scalingPolicy:3065d9c8-9969-4bec-  
bb6a-3fbe5550fde6:autoScalingGroupName/my-asg:policyName/alb1000-target-tracking-  
scaling-policy",  
      "PolicyType": "TargetTrackingScaling",  
      "StepAdjustments": [],  
      "Alarms": [  
        {  
          "AlarmName": "TargetTracking-my-asg-  
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196",  
          "AlarmARN": "arn:aws:cloudwatch:us-  
west-2:123456789012:alarm:TargetTracking-my-asg-  
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196"  
        },  
        {  
          "AlarmName": "TargetTracking-my-asg-AlarmLow-f96f899d-b8e7-4d09-  
a010-c1aaa35da296",
```

```

        "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-f96f899d-b8e7-4d09-a010-
c1aaa35da296"
    }
  ],
  "TargetTrackingConfiguration": {
    "PredefinedMetricSpecification": {
      "PredefinedMetricType": "ALBRequestCountPerTarget",
      "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-
alb-target-group/943f017f100becff"
    },
    "TargetValue": 1000.0,
    "DisableScaleIn": false
  },
  "Enabled": true
},
{
  "AutoScalingGroupName": "my-asg",
  "PolicyName": "cpu40-target-tracking-scaling-policy",
  "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:5fd26f71-39d4-4690-82a9-
b8515c45cdde:autoScalingGroupName/my-asg:policyName/cpu40-target-tracking-scaling-
policy",
  "PolicyType": "TargetTrackingScaling",
  "StepAdjustments": [],
  "Alarms": [
    {
      "AlarmName": "TargetTracking-my-asg-
AlarmHigh-139f9789-37b9-42ad-bea5-b5b147d7f473",
      "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-139f9789-37b9-42ad-bea5-
b5b147d7f473"
    },
    {
      "AlarmName": "TargetTracking-my-asg-AlarmLow-bd681c67-
fc18-4c56-8468-fb8e413009c9",
      "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-bd681c67-fc18-4c56-8468-
fb8e413009c9"
    }
  ],
  "TargetTrackingConfiguration": {
    "PredefinedMetricSpecification": {
      "PredefinedMetricType": "ASGAverageCPUUtilization"
    }
  }
}

```

```
        },
        "TargetValue": 40.0,
        "DisableScaleIn": false
      },
      "Enabled": true
    ]
  ]
}
```

有关更多信息，请参阅 Amazon EC2 Auto [Scaling 用户指南中的动态扩展](#)。

### 示例 2：描述指定名称的扩展策略

要返回特定的扩展策略，请使用 `--policy-names` 选项。

```
aws autoscaling describe-policies \  
  --auto-scaling-group-name my-asg \  
  --policy-names cpu40-target-tracking-scaling-policy
```

有关输出示例，请参阅示例 1。

有关更多信息，请参阅 Amazon EC2 Auto [Scaling 用户指南中的动态扩展](#)。

### 示例 3：描述一些扩展策略

要返回特定数量的策略，请使用 `--max-items` 选项。

```
aws autoscaling describe-policies \  
  --auto-scaling-group-name my-asg \  
  --max-items 1
```

有关输出示例，请参阅示例 1。

如果输出包含一个 `NextToken` 字段，则在后续调用中使用该字段的值和 `--starting-token` 选项来获取其他策略。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg --starting-token  
Z3M3LMPEXAMPLE
```

有关更多信息，请参阅 Amazon EC2 Auto [Scaling 用户指南中的动态扩展](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePolicies](#)中的。

## describe-scaling-activities

以下代码示例演示了如何使用 describe-scaling-activities。

### AWS CLI

#### 示例 1：描述指定组的扩展活动

此示例描述指定自动扩缩组的扩展活动。

```
aws autoscaling describe-scaling-activities \  
  --auto-scaling-group-name my-asg
```

输出：

```
{  
  "Activities": [  
    {  
      "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",  
      "Description": "Launching a new EC2 instance: i-0d44425630326060f",  
      "AutoScalingGroupName": "my-asg",  
      "Cause": "At 2020-10-30T19:35:51Z a user request update of  
AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the desired  
capacity from 0 to 16. At 2020-10-30T19:36:07Z an instance was started in response  
to a difference between desired and actual capacity, increasing the capacity from 0  
to 16.",  
      "StartTime": "2020-10-30T19:36:09.766Z",  
      "EndTime": "2020-10-30T19:36:41Z",  
      "StatusCode": "Successful",  
      "Progress": 100,  
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":  
\"us-west-2b\"}"  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[验证自动扩缩组的扩展活动](#)。

#### 示例 2：描述已删除组的扩展活动

要在删除自动扩缩组后描述扩展活动，请添加 `--include-deleted-groups` 选项。

```
aws autoscaling describe-scaling-activities \  
  --auto-scaling-group-name my-asg \  
  --include-deleted-groups
```

输出：

```
{  
  "Activities": [  
    {  
      "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",  
      "Description": "Launching a new EC2 instance. Status Reason: Your Spot  
request price of 0.001 is lower than the minimum required Spot request fulfillment  
price of 0.0031. Launching EC2 instance failed.",  
      "AutoScalingGroupName": "my-asg",  
      "Cause": "At 2021-01-13T20:47:24Z a user request update of  
AutoScalingGroup constraints to min: 1, max: 5, desired: 3 changing the desired  
capacity from 0 to 3. At 2021-01-13T20:47:27Z an instance was started in response  
to a difference between desired and actual capacity, increasing the capacity from 0  
to 3.",  
      "StartTime": "2021-01-13T20:47:30.094Z",  
      "EndTime": "2021-01-13T20:47:30Z",  
      "StatusCode": "Failed",  
      "StatusMessage": "Your Spot request price of 0.001 is lower than the  
minimum required Spot request fulfillment price of 0.0031. Launching EC2 instance  
failed.",  
      "Progress": 100,  
      "Details": "{\"Subnet ID\": \"subnet-5ea0c127\", \"Availability Zone\":  
\\\"us-west-2b\\\"}",  
      "AutoScalingGroupState": "Deleted",  
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-  
west-2:123456789012:autoScalingGroup:283179a2-  
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[排除 Amazon EC2 Auto Scaling 问题](#)。

示例 3：描述指定数量的扩展活动

要返回特定数量的活动，请使用 `--max-items` 选项。

```
aws autoscaling describe-scaling-activities \  
  --max-items 1
```

输出：

```
{  
  "Activities": [  
    {  
      "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",  
      "Description": "Launching a new EC2 instance: i-0d44425630326060f",  
      "AutoScalingGroupName": "my-asg",  
      "Cause": "At 2020-10-30T19:35:51Z a user request update of  
AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the desired  
capacity from 0 to 16. At 2020-10-30T19:36:07Z an instance was started in response  
to a difference between desired and actual capacity, increasing the capacity from 0  
to 16.",  
      "StartTime": "2020-10-30T19:36:09.766Z",  
      "EndTime": "2020-10-30T19:36:41Z",  
      "StatusCode": "Successful",  
      "Progress": 100,  
      "Details": "{\"Subnet ID\": \"subnet-5ea0c127\", \"Availability Zone\":  
\"us-west-2b\"}"  
    }  
  ]  
}
```

如果输出包含 `NextToken` 字段，可返回更多活动。要获取其他活动，请在后续调用中使用此字段的值和 `--starting-token` 选项，如下所示。

```
aws autoscaling describe-scaling-activities \  
  --starting-token Z3M3LMPEXAMPLE
```

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[验证自动扩缩组的扩展活动](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeScalingActivities](#)中的。

## describe-scaling-process-types

以下代码示例演示了如何使用 `describe-scaling-process-types`。

## AWS CLI

### 描述可用的流程类型

此示例描述了可用的流程类型。

```
aws autoscaling describe-scaling-process-types
```

输出：

```
{
  "Processes": [
    {
      "ProcessName": "AZRebalance"
    },
    {
      "ProcessName": "AddToLoadBalancer"
    },
    {
      "ProcessName": "AlarmNotification"
    },
    {
      "ProcessName": "HealthCheck"
    },
    {
      "ProcessName": "InstanceRefresh"
    },
    {
      "ProcessName": "Launch"
    },
    {
      "ProcessName": "ReplaceUnhealthy"
    },
    {
      "ProcessName": "ScheduledActions"
    },
    {
      "ProcessName": "Terminate"
    }
  ]
}
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的暂停和恢复扩展[流程](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeScalingProcessTypes](#)中的。

## describe-scheduled-actions

以下代码示例演示了如何使用 describe-scheduled-actions。

### AWS CLI

#### 示例 1：描述所有计划操作

此示例描述了您的所有计划操作。

```
aws autoscaling describe-scheduled-actions
```

输出：

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",
      "StartTime": "2023-12-01T04:00:00Z",
      "Time": "2023-12-01T04:00:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4,
      "TimeZone": "America/New_York"
    }
  ]
}
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[计划扩展](#)。

#### 示例 2：描述指定组的计划操作

要描述特定 Auto Scaling 组的计划操作，请使用 `--auto-scaling-group-name` 选项。

```
aws autoscaling describe-scheduled-actions \
```

```
--auto-scaling-group-name my-asg
```

输出：

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",
      "StartTime": "2023-12-01T04:00:00Z",
      "Time": "2023-12-01T04:00:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4,
      "TimeZone": "America/New_York"
    }
  ]
}
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[计划扩展](#)。

示例 3：描述指定的计划操作

要描述特定的计划操作，请使用 `--scheduled-action-names` 选项。

```
aws autoscaling describe-scheduled-actions \
  --scheduled-action-names my-recurring-action
```

输出：

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",

```

```
        "StartTime": "2023-12-01T04:00:00Z",
        "Time": "2023-12-01T04:00:00Z",
        "MinSize": 1,
        "MaxSize": 6,
        "DesiredCapacity": 4,
        "TimeZone": "America/New_York"
    }
]
}
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[计划扩展](#)。

示例 4：描述具有指定开始时间的计划操作

要描述在特定时间开始的计划操作，请使用 `--start-time` 选项。

```
aws autoscaling describe-scheduled-actions \
  --start-time "2023-12-01T04:00:00Z"
```

输出：

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",
      "StartTime": "2023-12-01T04:00:00Z",
      "Time": "2023-12-01T04:00:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4,
      "TimeZone": "America/New_York"
    }
  ]
}
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[计划扩展](#)。

示例 5：描述在指定时间结束的计划操作

要描述在特定时间结束的计划操作，请使用 `--end-time` 选项。

```
aws autoscaling describe-scheduled-actions \  
  --end-time "2023-12-01T04:00:00Z"
```

输出：

```
{  
  "ScheduledUpdateGroupActions": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "ScheduledActionName": "my-recurring-action",  
      "Recurrence": "30 0 1 1,6,12 *",  
      "ScheduledActionARN": "arn:aws:autoscaling:us-  
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-  
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",  
      "StartTime": "2023-12-01T04:00:00Z",  
      "Time": "2023-12-01T04:00:00Z",  
      "MinSize": 1,  
      "MaxSize": 6,  
      "DesiredCapacity": 4,  
      "TimeZone": "America/New_York"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 [计划扩展](#)。

示例 6：描述指定数量的计划操作

要返回特定数量的计划操作，请使用 `--max-items` 选项。

```
aws autoscaling describe-scheduled-actions \  
  --auto-scaling-group-name my-asg \  
  --max-items 1
```

输出：

```
{  
  "ScheduledUpdateGroupActions": [  
    {  
      "AutoScalingGroupName": "my-asg",
```

```

        "ScheduledActionName": "my-recurring-action",
        "Recurrence": "30 0 1 1,6,12 *",
        "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",
        "StartTime": "2023-12-01T04:00:00Z",
        "Time": "2023-12-01T04:00:00Z",
        "MinSize": 1,
        "MaxSize": 6,
        "DesiredCapacity": 4,
        "TimeZone": "America/New_York"
    }
]
}

```

如果输出包含字NextToken段，则还有更多计划操作。要获取其他计划操作，请在后续调用中使用此字段的值和--starting-token选项，如下所示。

```

aws autoscaling describe-scheduled-actions \
  --auto-scaling-group-name my-asg \
  --starting-token Z3M3LMPEXAMPLE

```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[计划扩展](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeScheduledActions](#)中的。

## describe-tags

以下代码示例演示了如何使用 describe-tags。

### AWS CLI

描述所有标签

此示例描述了您的所有标签。

```
aws autoscaling describe-tags
```

输出：

```

{
  "Tags": [

```

```
{
  "ResourceType": "auto-scaling-group",
  "ResourceId": "my-asg",
  "PropagateAtLaunch": true,
  "Value": "Research",
  "Key": "Dept"
},
{
  "ResourceType": "auto-scaling-group",
  "ResourceId": "my-asg",
  "PropagateAtLaunch": true,
  "Value": "WebServer",
  "Key": "Role"
}
]
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的为 Auto Scaling [组和实例添加标签](#)。

#### 示例 2：描述指定群组的标签

要描述特定 Auto Scaling 组的标签，请使用 `--filters` 选项。

```
aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的为 Auto Scaling [组和实例添加标签](#)。

#### 示例 3：描述指定数量的标签

要返回特定数量的标签，请使用 `--max-items` 选项。

```
aws autoscaling describe-tags \
  --max-items 1
```

如果输出包含 `NextToken` 段，则会有更多标签。要获取其他标签，请在后续调用中使用此字段的值和 `--starting-token` 选项，如下所示。

```
aws autoscaling describe-tags \
  --filters Name=auto-scaling-group,Values=my-asg \
  --starting-token Z3M3LMPEXAMPLE
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的为 Auto Scaling [组和实例添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTags](#)中的。

## describe-termination-policy-types

以下代码示例演示了如何使用 describe-termination-policy-types。

### AWS CLI

描述可用的终止政策类型

此示例描述了可用的终止策略类型。

```
aws autoscaling describe-termination-policy-types
```

输出：

```
{
  "TerminationPolicyTypes": [
    "AllocationStrategy",
    "ClosestToNextInstanceHour",
    "Default",
    "NewestInstance",
    "OldestInstance",
    "OldestLaunchConfiguration",
    "OldestLaunchTemplate"
  ]
}
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的控制哪些 Auto Scaling 实例在[缩容期间终止](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTerminationPolicyTypes](#)中的。

## describe-warm-pool

以下代码示例演示了如何使用 describe-warm-pool。

### AWS CLI

描述一个温暖的泳池

此示例描述了指定 Auto Scaling 组的温池。

```
aws autoscaling describe-warm-pool \  
  --auto-scaling-group-name my-asg
```

输出：

```
{  
  "WarmPoolConfiguration": {  
    "MinSize": 2,  
    "PoolState": "Stopped"  
  },  
  "Instances": [  
    {  
      "InstanceId": "i-070a5bbc7e7f40dc5",  
      "InstanceType": "t2.micro",  
      "AvailabilityZone": "us-west-2c",  
      "LifecycleState": "Warmed:Pending",  
      "HealthStatus": "Healthy",  
      "LaunchTemplate": {  
        "LaunchTemplateId": "lt-00a731f6e9fa48610",  
        "LaunchTemplateName": "my-template-for-auto-scaling",  
        "Version": "6"  
      }  
    },  
    {  
      "InstanceId": "i-0b52f061814d3bd2d",  
      "InstanceType": "t2.micro",  
      "AvailabilityZone": "us-west-2b",  
      "LifecycleState": "Warmed:Pending",  
      "HealthStatus": "Healthy",  
      "LaunchTemplate": {  
        "LaunchTemplateId": "lt-00a731f6e9fa48610",  
        "LaunchTemplateName": "my-template-for-auto-scaling",  
        "Version": "6"  
      }  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 [Amazon EC2 Auto Scaling 的温水池](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeWarmPool](#)中的。

## detach-instances

以下代码示例演示了如何使用 detach-instances。

### AWS CLI

将实例与 Auto Scaling 组分离

此示例将指定实例与指定的 Auto Scaling 组分离。

```
aws autoscaling detach-instances \  
  --instance-ids i-030017cfa84b20135 \  
  --auto-scaling-group-name my-asg \  
  --should-decrement-desired-capacity
```

输出：

```
{  
  "Activities": [  
    {  
      "ActivityId": "5091cb52-547a-47ce-a236-c9ccbc2cb2c9",  
      "AutoScalingGroupName": "my-asg",  
      "Description": "Detaching EC2 instance: i-030017cfa84b20135",  
      "Cause": "At 2020-10-31T17:35:04Z instance i-030017cfa84b20135 was  
detached in response to a user request, shrinking the capacity from 2 to 1.",  
      "StartTime": "2020-04-12T15:02:16.179Z",  
      "StatusCode": "InProgress",  
      "Progress": 50,  
      "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":  
\"us-west-2c\"}"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachInstances](#)中的。

## detach-load-balancer-target-groups

以下代码示例演示了如何使用 detach-load-balancer-target-groups。

## AWS CLI

将负载均衡器目标组与 Auto Scaling 组分离

此示例将指定的负载均衡器目标组与指定的 Auto Scaling 组分离。

```
aws autoscaling detach-load-balancer-target-groups \  
  --auto-scaling-group-name my-asg \  
  --target-group-arns arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

此命令不产生任何输出

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[将负载均衡器挂接到 Auto Scaling 组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachLoadBalancerTargetGroups](#)中的。

## detach-load-balancers

以下代码示例演示了如何使用 detach-load-balancers。

### AWS CLI

将 Classic Load Balancer 与 Auto Scaling 组分离

此示例将指定的 Classic Load Balancer 与指定的 Auto Scaling 组分离。

```
aws autoscaling detach-load-balancers \  
  --load-balancer-names my-load-balancer \  
  --auto-scaling-group-name my-asg
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[将负载均衡器挂接到 Auto Scaling 组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachLoadBalancers](#)中的。

## disable-metrics-collection

以下代码示例演示了如何使用 disable-metrics-collection。

## AWS CLI

### 禁用自动扩缩组指标收集

此示例将禁用指定自动扩缩组的 GroupDesiredCapacity 指标的收集。

```
aws autoscaling disable-metrics-collection \  
  --auto-scaling-group-name my-asg \  
  --metrics GroupDesiredCapacity
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 Auto Scaling [组和实例的监控 CloudWatch 指标](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisableMetricsCollection](#) 中的。

## enable-metrics-collection

以下代码示例演示了如何使用 enable-metrics-collection。

## AWS CLI

### 示例 1：启用自动扩缩组的指标收集

此示例将启用指定自动扩缩组的数据收集。

```
aws autoscaling enable-metrics-collection \  
  --auto-scaling-group-name my-asg \  
  --granularity "1Minute"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 Auto Scaling [组和实例的监控 CloudWatch 指标](#)。

### 示例 2：收集自动扩缩组指定指标的相关数据

要收集特定指标的相关数据，请使用 --metrics 选项。

```
aws autoscaling enable-metrics-collection \  
  --metrics GroupDesiredCapacity
```

```
--auto-scaling-group-name my-asg \  
--metrics GroupDesiredCapacity --granularity "1Minute"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 Auto Scaling [组和实例的监控 CloudWatch 指标](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [EnableMetricsCollection](#) 中的。

## enter-standby

以下代码示例演示了如何使用 enter-standby。

### AWS CLI

将实例移至待机模式

此示例将指定实例置于待机模式。这对于更新当前正在运行的实例或对其进行故障排除非常有用。

```
aws autoscaling enter-standby \  
--instance-ids i-061c63c5eb45f0416 \  
--auto-scaling-group-name my-asg \  
--should-decrement-desired-capacity
```

输出：

```
{  
  "Activities": [  
    {  
      "ActivityId": "ffa056b4-6ed3-41ba-ae7c-249dfae6eba1",  
      "AutoScalingGroupName": "my-asg",  
      "Description": "Moving EC2 instance to Standby: i-061c63c5eb45f0416",  
      "Cause": "At 2020-10-31T20:31:00Z instance i-061c63c5eb45f0416 was moved  
to standby in response to a user request, shrinking the capacity from 1 to 0.",  
      "StartTime": "2020-10-31T20:31:00.949Z",  
      "StatusCode": "InProgress",  
      "Progress": 50,  
      "Details": "{\"Subnet ID\": \"subnet-6194ea3b\", \"Availability Zone\":  
\"us-west-2c\"}"  
    }  
  ]  
}
```

```
}
```

有关更多信息，请参阅 [Amazon EC2 Auto Scaling 用户指南中的 Amazon EC2 Auto Scaling 实例生命周期](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnterStandby](#)中的。

## execute-policy

以下代码示例演示了如何使用 `execute-policy`。

### AWS CLI

执行扩展策略

此示例执行`my-step-scale-out-policy`为指定的 Auto Scaling 组命名的扩展策略。

```
aws autoscaling execute-policy \  
  --auto-scaling-group-name my-asg \  
  --policy-name my-step-scale-out-policy \  
  --metric-value 95 \  
  --breach-threshold 80
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的步骤和简单扩展[策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ExecutePolicy](#)中的。

## exit-standby

以下代码示例演示了如何使用 `exit-standby`。

### AWS CLI

将实例移出待机模式

此示例将指定实例移出待机模式。

```
aws autoscaling exit-standby \  
  --instance-ids i-061c63c5eb45f0416 \  
  --auto-scaling-group-name my-asg
```

输出：

```
{
  "Activities": [
    {
      "ActivityId": "142928e1-a2dc-453a-9b24-b85ad6735928",
      "AutoScalingGroupName": "my-asg",
      "Description": "Moving EC2 instance out of Standby:
i-061c63c5eb45f0416",
      "Cause": "At 2020-10-31T20:32:50Z instance i-061c63c5eb45f0416 was moved
out of standby in response to a user request, increasing the capacity from 0 to
1.",
      "StartTime": "2020-10-31T20:32:50.222Z",
      "StatusCode": "PreInService",
      "Progress": 30,
      "Details": "{\"Subnet ID\": \"subnet-6194ea3b\", \"Availability Zone\":
\\\"us-west-2c\\\"}"
    }
  ]
}
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的临时从 Auto Scaling [组中移除实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ExitStandby](#) 中的。

## put-lifecycle-hook

以下代码示例演示了如何使用 put-lifecycle-hook。

### AWS CLI

#### 示例 1：创建生命周期挂钩

此示例创建了一个生命周期挂钩，该挂钩将在任何新启动的实例上调用，超时时间为 4800 秒。这对于在用户数据脚本完成之前保持实例处于等待状态或使用调用 Lambda AWS a 函数非常有用。

#### EventBridge

```
aws autoscaling put-lifecycle-hook \
  --auto-scaling-group-name my-asg \
  --lifecycle-hook-name my-launch-hook \
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
```

```
--heartbeat-timeout 4800
```

此命令不生成任何输出。如果已存在同名的生命周期挂钩，则新的生命周期挂钩将被新的生命周期挂钩覆盖。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 [Amazon EC2 Auto Scaling 生命周期挂钩](#)。

示例 2：发送 Amazon SNS 电子邮件通知您实例状态的转换

此示例创建了一个生命周期挂钩，其中包含 Amazon SNS 主题和 IAM 角色，用于在实例启动时接收通知。

```
aws autoscaling put-lifecycle-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-hook-name my-launch-hook \  
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \  
  --notification-target-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \  
  --role-arn arn:aws:iam::123456789012:role/my-auto-scaling-role
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 [Amazon EC2 Auto Scaling 生命周期挂钩](#)。

示例 3：向 Amazon SQS 队列发布消息

此示例创建了一个生命周期挂钩，该钩子将包含元数据的消息发布到指定的 Amazon SQS 队列。

```
aws autoscaling put-lifecycle-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-hook-name my-launch-hook \  
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \  
  --notification-target-arn arn:aws:sqs:us-west-2:123456789012:my-sqs-queue \  
  --role-arn arn:aws:iam::123456789012:role/my-notification-role \  
  --notification-metadata "SQS message metadata"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 [Amazon EC2 Auto Scaling 生命周期挂钩](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutLifecycleHook](#) 中的。

## put-notification-configuration

以下代码示例演示了如何使用 put-notification-configuration。

### AWS CLI

#### 添加通知

此示例将指定的通知添加到指定的 Auto Scaling 组。

```
aws autoscaling put-notification-configuration \  
  --auto-scaling-group-name my-asg \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \  
  --notification-type autoscaling:TEST_NOTIFICATION
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的 [Auto Scaling 组扩展时获取 Amazon SNS 通知](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutNotificationConfiguration](#) 中的。

## put-scaling-policy

以下代码示例演示了如何使用 put-scaling-policy。

### AWS CLI

#### 向 Auto Scaling 组添加目标跟踪扩展策略

以下 put-scaling-policy 示例将目标跟踪扩展策略应用于指定的 Auto Scaling 组。输出包含代表您创建的两个 CloudWatch 警报的 ARN 和名称。如果已存在同名的扩展策略，则该策略将被新的扩展策略覆盖。

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \  
  --policy-name alb1000-target-tracking-scaling-policy \  
  --policy-type TargetTrackingScaling \  
  --target-tracking-configuration file://config.json
```

config.json 的内容：

```
{
```



```
"TargetValue": 1000.0,
"PredefinedMetricSpecification": {
  "PredefinedMetricType": "ALBRequestCountPerTarget",
  "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-target-
group/943f017f100becff"
}
}
```

输出：

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:228f02c2-
c665-4bfd-aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/alb1000-target-
tracking-scaling-policy",
  "Alarms": [
    {
      "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-
my-asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
      "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
    },
    {
      "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-
my-asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",
      "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
    }
  ]
}
```

有关更多示例，请参阅 Amazon EC2 Auto Scaling 用户指南中的 AWS 命令行界面 (AWS CLI) 扩展策略示例。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutScalingPolicy](#) 中的。

## put-scheduled-update-group-action

以下代码示例演示了如何使用 put-scheduled-update-group-action。

### AWS CLI

示例 1：向 Auto Scaling 组添加计划操作

此示例将指定的计划操作添加到指定的 Auto Scaling 组中。

```
aws autoscaling put-scheduled-update-group-action \  
  --auto-scaling-group-name my-asg \  
  --scheduled-action-name my-scheduled-action \  
  --start-time "2023-05-12T08:00:00Z" \  
  --min-size 2 \  
  --max-size 6 \  
  --desired-capacity 4
```

此命令不生成任何输出。如果已存在同名的预定操作，则该操作将被新的计划操作覆盖。

有关更多示例，请参阅 Amazon EC2 Auto [Scaling 用户指南中的计划扩展](#)。

### 示例 2：指定期计划

此示例创建了一个计划操作，以按定期计划进行扩展，该计划计划在每年一月、六月和十二月的第一天 00:30 执行。

```
aws autoscaling put-scheduled-update-group-action \  
  --auto-scaling-group-name my-asg \  
  --scheduled-action-name my-recurring-action \  
  --recurrence "30 0 1 1,6,12 *" \  
  --min-size 2 \  
  --max-size 6 \  
  --desired-capacity 4
```

此命令不生成任何输出。如果已存在同名的预定操作，则该操作将被新的计划操作覆盖。

有关更多示例，请参阅 Amazon EC2 Auto [Scaling 用户指南中的计划扩展](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutScheduledUpdateGroupAction](#) 中的。

## put-warm-pool

以下代码示例演示了如何使用 put-warm-pool。

### AWS CLI

#### 创建温水池

以下示例为指定的 Auto Scaling 组创建了一个温池。

```
aws autoscaling put-warm-pool \  
  --auto-scaling-group-name my-asg \  
  --min-size 2
```

此命令不生成任何输出。如果温池已经存在，则会对其进行更新。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 [Amazon EC2 Auto Scaling 的温水池](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutWarmPool](#) 中的。

## record-lifecycle-action-heartbeat

以下代码示例演示了如何使用 record-lifecycle-action-heartbeat。

### AWS CLI

#### 记录生命周期操作心跳

此示例记录生命周期操作心跳，以使实例保持待处理状态。

```
aws autoscaling record-lifecycle-action-heartbeat \  
  --lifecycle-hook-name my-launch-hook \  
  --auto-scaling-group-name my-asg \  
  --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 [Amazon EC2 Auto Scaling 生命周期挂钩](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RecordLifecycleActionHeartbeat](#) 中的。

## resume-processes

以下代码示例演示了如何使用 resume-processes。

### AWS CLI

#### 恢复暂停的进程

此示例恢复指定的 Auto Scaling 组的指定已暂停的扩展过程。

```
aws autoscaling resume-processes \  
  --auto-scaling-group-name my-asg \  
  --scaling-processes AlarmNotification
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的暂停和恢复扩展[流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ResumeProcesses](#)中的。

## rollback-instance-refresh

以下代码示例演示了如何使用 rollback-instance-refresh。

### AWS CLI

要回滚实例，请刷新

以下rollback-instance-refresh示例回滚指定的 Auto Scaling 组正在进行的实例刷新。

```
aws autoscaling rollback-instance-refresh \  
  --auto-scaling-group-name my-asg
```

输出：

```
{  
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"  
}
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[通过回滚撤销更改](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RollbackInstanceRefresh](#)中的。

## set-desired-capacity

以下代码示例演示了如何使用 set-desired-capacity。

### AWS CLI

为自动扩缩组设置所需容量

此示例将为指定的自动扩缩组设置所需容量。

```
aws autoscaling set-desired-capacity \  
  --auto-scaling-group-name my-asg \  
  --desired-capacity 2 \  
  --honor-cooldown
```

如果成功，该命令将返回到提示符。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetDesiredCapacity](#)中的。

## set-instance-health

以下代码示例演示了如何使用 set-instance-health。

### AWS CLI

设置实例的运行状况

此示例将指定实例的运行状况设置为Unhealthy。

```
aws autoscaling set-instance-health \  
  --instance-id i-061c63c5eb45f0416 \  
  --health-status Unhealthy
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetInstanceHealth](#)中的。

## set-instance-protection

以下代码示例演示了如何使用 set-instance-protection。

### AWS CLI

示例 1：为实例启用实例保护设置

此示例为指定实例启用实例保护。

```
aws autoscaling set-instance-protection \  
  --instance-ids i-061c63c5eb45f0416 \  
  --instance-ids i-061c63c5eb45f0416
```

```
--auto-scaling-group-name my-asg --protected-from-scale-in
```

此命令不生成任何输出。

### 示例 2：禁用实例的实例保护设置

此示例禁用指定实例的实例保护。

```
aws autoscaling set-instance-protection \  
  --instance-ids i-061c63c5eb45f0416 \  
  --auto-scaling-group-name my-asg \  
  --no-protected-from-scale-in
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetInstanceProtection](#)中的。

## start-instance-refresh

以下代码示例演示了如何使用 start-instance-refresh。

### AWS CLI

#### 示例 1：使用命令行参数启动实例刷新

以下start-instance-refresh示例使用命令行参数启动实例刷新。可选preferences参数指定 a InstanceWarmup 为60秒，a MinHealthyPercentage 为50百分比。

```
aws autoscaling start-instance-refresh \  
  --auto-scaling-group-name my-asg \  
  --preferences '{"InstanceWarmup": 60, "MinHealthyPercentage": 50}'
```

输出：

```
{  
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"  
}
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[启动实例刷新](#)。

#### 示例 2：使用 JSON 文件启动实例刷新

以下 `start-instance-refresh` 示例使用 JSON 文件启动实例刷新。您可以指定 Auto Scaling 组并在 JSON 文件中定义所需的配置和首选项，如以下示例所示。

```
aws autoscaling start-instance-refresh \  
  --cli-input-json file://config.json
```

`config.json` 的内容：

```
{  
  "AutoScalingGroupName": "my-asg",  
  "DesiredConfiguration": {  
    "LaunchTemplate": {  
      "LaunchTemplateId": "lt-068f72b729example",  
      "Version": "$Default"  
    }  
  },  
  "Preferences": {  
    "InstanceWarmup": 60,  
    "MinHealthyPercentage": 50,  
    "AutoRollback": true,  
    "ScaleInProtectedInstances": Ignore,  
    "StandbyInstances": Terminate  
  }  
}
```

输出：

```
{  
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"  
}
```

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 [启动实例刷新](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartInstanceRefresh](#) 中的。

## suspend-processes

以下代码示例演示了如何使用 `suspend-processes`。

### AWS CLI

暂停 Auto Scaling 进程

此示例暂停指定 Auto Scaling 组的指定伸缩过程。

```
aws autoscaling suspend-processes \  
  --auto-scaling-group-name my-asg \  
  --scaling-processes AlarmNotification
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的暂停和恢复扩展[流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[SuspendProcesses](#)中的。

## terminate-instance-in-auto-scaling-group

以下代码示例演示了如何使用 terminate-instance-in-auto-scaling-group。

### AWS CLI

终止自动扩缩组中的实例

此示例将在不更新指定自动扩缩组大小的情况下终止该组中的指定实例。Amazon EC2 Auto Scaling 会在指定实例终止后启动替代实例。

```
aws autoscaling terminate-instance-in-auto-scaling-group \  
  --instance-id i-061c63c5eb45f0416 \  
  --no-should-decrement-desired-capacity
```

输出：

```
{  
  "Activities": [  
    {  
      "ActivityId": "8c35d601-793c-400c-fcd0-f64a27530df7",  
      "AutoScalingGroupName": "my-asg",  
      "Description": "Terminating EC2 instance: i-061c63c5eb45f0416",  
      "Cause": "",  
      "StartTime": "2020-10-31T20:34:25.680Z",  
      "StatusCode": "InProgress",  
      "Progress": 0,  
      "Details": "{\"Subnet ID\": \"subnet-6194ea3b\", \"Availability Zone\":  
        \"us-west-2c\"}"
```



```
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TerminateInstanceInAutoScalingGroup](#)中的。

## update-auto-scaling-group

以下代码示例演示了如何使用 update-auto-scaling-group。

### AWS CLI

#### 示例 1：更新自动扩缩组的大小限制

该示例将更新指定的自动扩缩组，该组的最小大小为 1，最大大小为 10。

```
aws autoscaling update-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --min-size 2 \  
  --max-size 10
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[设置自动扩缩组的容量限制](#)。

#### 示例 2：添加 Elastic Load Balancing 运行状况检查并指定要使用的可用区和子网

此示例将更新指定的自动扩缩组以添加 Elastic Load Balancing 运行状况检查。此命令还会使用多个可用区中的子网 ID 列表更新 --vpc-zone-identifier 的值。

```
aws autoscaling update-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --health-check-type ELB \  
  --health-check-grace-period 600 \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的 [Elastic Load Balancing 和 Amazon EC2 Auto Scaling](#)。

### 示例 3：更新置放群组 and 终止策略

此示例将更新要使用的置放群组和终止策略。

```
aws autoscaling update-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --placement-group my-placement-group \  
  --termination-policies "OldestInstance"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的[自动扩缩组](#)。

### 示例 4：使用最新版本的启动模板

此示例会将指定的自动扩缩组更新为使用最新版本的指定启动模板。

```
aws autoscaling update-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest'
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[启动模板](#)。

### 示例 5：使用特定版本的启动模板

此示例会将指定的自动扩缩组更新为使用特定版本的启动模板，而不是最新或默认版本。

```
aws autoscaling update-auto-scaling-group \  
  --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='2'
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[启动模板](#)。

### 示例 6：定义混合实例策略并启用容量再平衡

此示例将指定的自动扩缩组更新为使用混合实例策略并启用容量再平衡。此结构允许您指定具有竞价和按需容量的组，并针对不同的架构使用不同的启动模板。

```
aws autoscaling update-auto-scaling-group \  
  --cli-input-json file://~/config.json
```

config.json 的内容：

```
{  
  "AutoScalingGroupName": "my-asg",  
  "CapacityRebalance": true,  
  "MixedInstancesPolicy": {  
    "LaunchTemplate": {  
      "LaunchTemplateSpecification": {  
        "LaunchTemplateName": "my-launch-template-for-x86",  
        "Version": "$Latest"  
      },  
      "Overrides": [  
        {  
          "InstanceType": "c6g.large",  
          "LaunchTemplateSpecification": {  
            "LaunchTemplateName": "my-launch-template-for-arm",  
            "Version": "$Latest"  
          }  
        },  
        {  
          "InstanceType": "c5.large"  
        },  
        {  
          "InstanceType": "c5a.large"  
        }  
      ]  
    },  
    "InstancesDistribution": {  
      "OnDemandPercentageAboveBaseCapacity": 50,  
      "SpotAllocationStrategy": "capacity-optimized"  
    }  
  }  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[具有多个实例类型和购买选项的自动扩缩组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateAutoScalingGroup](#)中的。

## 使用 Auto Scaling Plans 示例 AWS CLI

以下代码示例向您展示了如何使用与 Auto Scaling Plans AWS Command Line Interface 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### create-scaling-plan

以下代码示例演示了如何使用 create-scaling-plan。

AWS CLI

创建扩展计划

以下 create-scaling-plan 示例 my-scaling-plan 使用已创建的 JSON 文件（名为 config.json）创建了一个名为的扩展计划。扩展计划的结构包括名为的 Auto Scaling 组的扩展指令 my-asg。它指定 TagFilters 属性作为应用程序源，并启用预测式扩展和动态扩展。

```
aws autoscaling-plans create-scaling-plan \
  --scaling-plan-name my-scaling-plan \
  --cli-input-json file://~/config.json
```

config.json 文件的内容：

```
{
  "ApplicationSource": {
    "TagFilters": [
      {
```



## delete-scaling-plan

以下代码示例演示了如何使用 delete-scaling-plan。

### AWS CLI

删除扩展计划

以下delete-scaling-plan示例删除了指定的扩展计划。

```
aws autoscaling-plans delete-scaling-plan \  
  --scaling-plan-name my-scaling-plan \  
  --scaling-plan-version 1
```

此命令不生成任何输出。

有关更多信息，请参阅 [AWS Auto Scaling 用户指南](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteScalingPlan](#)中的。

## describe-scaling-plan-resources

以下代码示例演示了如何使用 describe-scaling-plan-resources。

### AWS CLI

描述扩展计划的可扩展资源

以下describe-scaling-plan-resources示例显示与指定扩展计划关联的单个可扩展资源（Auto Scaling 组）的详细信息。

```
aws autoscaling-plans describe-scaling-plan-resources \  
  --scaling-plan-name my-scaling-plan \  
  --scaling-plan-version 1
```

输出：

```
{  
  "ScalingPlanResources": [  
    {  
      "ScalableDimension": "autoscaling:autoScalingGroup:DesiredCapacity",
```

```

    "ScalingPlanVersion": 1,
    "ResourceId": "autoScalingGroup/my-asg",
    "ScalingStatusCode": "Active",
    "ScalingStatusMessage": "Target tracking scaling policies have been
applied to the resource.",
    "ScalingPolicies": [
      {
        "PolicyName": "AutoScaling-my-asg-b1ab65ae-4be3-4634-bd64-
c7471662b251",
        "PolicyType": "TargetTrackingScaling",
        "TargetTrackingConfiguration": {
          "PredefinedScalingMetricSpecification": {
            "PredefinedScalingMetricType":
"ALBRequestCountPerTarget",
            "ResourceLabel": "app/my-alb/f37c06a68c1748aa/
targetgroup/my-target-group/6d4ea56ca2d6a18d"
          },
          "TargetValue": 40.0
        }
      }
    ],
    "ServiceNamespace": "autoscaling",
    "ScalingPlanName": "my-scaling-plan"
  }
]
}

```

有关更多信息，请参阅[什么是 AWS Auto Scaling？](#) 在 AWS Auto Scaling 用户指南中。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeScalingPlanResources](#)中的。

## describe-scaling-plans

以下代码示例演示了如何使用 describe-scaling-plans。

### AWS CLI

#### 描述扩展计划

以下 describe-scaling-plans 示例显示了指定扩展计划的详细信息。

```

aws autoscaling-plans describe-scaling-plans \
  --scaling-plan-names scaling-plan-with-asg-and-ddb

```

输出：

```
{
  "ScalingPlans": [
    {
      "LastMutatingRequestTime": 1565388443.963,
      "ScalingPlanVersion": 1,
      "CreationTime": 1565388443.963,
      "ScalingInstructions": [
        {
          "ScalingPolicyUpdateBehavior": "ReplaceExternalPolicies",
          "ScalableDimension":
"autoscaling:autoScalingGroup:DesiredCapacity",
          "TargetTrackingConfigurations": [
            {
              "PredefinedScalingMetricSpecification": {
                "PredefinedScalingMetricType":
"ASGAverageCPUUtilization"
              },
              "TargetValue": 50.0,
              "EstimatedInstanceWarmup": 300,
              "DisableScaleIn": false
            }
          ],
          "ResourceId": "autoScalingGroup/my-asg",
          "DisableDynamicScaling": false,
          "MinCapacity": 1,
          "ServiceNamespace": "autoscaling",
          "MaxCapacity": 10
        },
        {
          "ScalingPolicyUpdateBehavior": "ReplaceExternalPolicies",
          "ScalableDimension": "dynamodb:table:ReadCapacityUnits",
          "TargetTrackingConfigurations": [
            {
              "PredefinedScalingMetricSpecification": {
                "PredefinedScalingMetricType":
"DynamoDBReadCapacityUtilization"
              },
              "TargetValue": 50.0,
              "ScaleInCooldown": 60,
              "DisableScaleIn": false,
              "ScaleOutCooldown": 60
            }
          ]
        }
      ]
    }
  ]
}
```



```

    ],
    "ResourceId": "table/my-table",
    "DisableDynamicScaling": false,
    "MinCapacity": 5,
    "ServiceNamespace": "dynamodb",
    "MaxCapacity": 10000
  },
  {
    "ScalingPolicyUpdateBehavior": "ReplaceExternalPolicies",
    "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
    "TargetTrackingConfigurations": [
      {
        "PredefinedScalingMetricSpecification": {
          "PredefinedScalingMetricType":
"DynamoDBWriteCapacityUtilization"
        },
        "TargetValue": 50.0,
        "ScaleInCooldown": 60,
        "DisableScaleIn": false,
        "ScaleOutCooldown": 60
      }
    ],
    "ResourceId": "table/my-table",
    "DisableDynamicScaling": false,
    "MinCapacity": 5,
    "ServiceNamespace": "dynamodb",
    "MaxCapacity": 10000
  }
],
"ApplicationSource": {
  "TagFilters": [
    {
      "Values": [
        "my-application-id"
      ],
      "Key": "application"
    }
  ]
},
"StatusStartTime": 1565388455.836,
"ScalingPlanName": "scaling-plan-with-asg-and-ddb",
"StatusMessage": "Scaling plan has been created and applied to all
resources.",
"StatusCode": "Active"

```

```
    }  
  ]  
}
```

有关更多信息，请参阅[什么是 AWS Auto Scaling ?](#) 在 AWS Auto Scaling 用户指南中。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeScalingPlans](#)中的。

## get-scaling-plan-resource-forecast-data

以下代码示例演示了如何使用 `get-scaling-plan-resource-forecast-data`。

### AWS CLI

#### 检索负荷预测数据

此示例检索与指定扩展计划关联的可扩展资源（Auto Scaling 组）的负载预测数据。

```
aws autoscaling-plans get-scaling-plan-resource-forecast-data \  
  --scaling-plan-name my-scaling-plan \  
  --scaling-plan-version 1 \  
  --service-namespace "autoscaling" \  
  --resource-id autoScalingGroup/my-asg \  
  --scalable-dimension "autoscaling:autoScalingGroup:DesiredCapacity" \  
  --forecast-data-type "LoadForecast" \  
  --start-time "2019-08-30T00:00:00Z" \  
  --end-time "2019-09-06T00:00:00Z"
```

输出：

```
{  
  "Datapoints": [...]  
}
```

有关更多信息，请参阅《[AWS Auto Scaling 用户指南](#)》中的“什么是AWS自动缩放”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetScalingPlanResourceForecastData](#)中的。

## update-scaling-plan

以下代码示例演示了如何使用 `update-scaling-plan`。

## AWS CLI

### 更新扩展计划

以下update-scaling-plan示例修改了指定扩展计划中 Auto Scaling 组的扩展指标。

```
aws autoscaling-plans update-scaling-plan \  
  --scaling-plan-name my-scaling-plan \  
  --scaling-plan-version 1 \  
  --scaling-instructions \  
  '{"ScalableDimension":"autoscaling:autoScalingGroup:DesiredCapacity","ResourceId":"autoScal  
my-asg","ServiceNamespace":"autoscaling","TargetTrackingConfigurations":  
 [{"PredefinedScalingMetricSpecification":  
   {"PredefinedScalingMetricType":"ALBRequestCountPerTarget","ResourceLabel":"app/my-  
alb/f37c06a68c1748aa/targetgroup/my-target-  
group/6d4ea56ca2d6a18d"},"TargetValue":40.0}], "MinCapacity": 1, "MaxCapacity": 10}'
```

此命令不生成任何输出。

有关更多信息，请参阅[什么是 AWS Auto Scaling ?](#) 在 AWS Auto Scaling 用户指南中。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateScalingPlan](#)中的。

## AWS Backup 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Backup。 AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### create-backup-plan

以下代码示例演示了如何使用 create-backup-plan。

#### AWS CLI

##### 创建备份计划

以下create-backup-plan示例创建了保留期为 35 天的指定备份计划。

```
aws backup create-backup-plan \  
--backup-plan "{\"BackupPlanName\": \"Example-Backup-Plan\", \"Rules\": [{\"RuleName\": \  
\"DailyBackups\", \"ScheduleExpression\": \"cron(0 5 ? * * *)\", \"StartWindowMinutes \  
\": 480, \"TargetBackupVaultName\": \"Default\", \"Lifecycle\": {\"DeleteAfterDays \  
\": 35}}]}"
```

输出：

```
{  
  "BackupPlanId": "1fa3895c-a7f5-484a-a371-2dd6a1a9f729",  
  "BackupPlanArn": "arn:aws:backup:us-west-2:123456789012:backup-plan:1fa3895c-  
a7f5-484a-a371-2dd6a1a9f729",  
  "CreationDate": 1568928754.747,  
  "VersionId": "ZjQ2ZTI5YWQtZDg5Yi00MzYzLWJmZTAzMDE1Mzh1MDhjYjEz"  
}
```

有关更多信息，请参阅《Backup 开发者指南》中的创建AWS 备份[计划](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateBackupPlan](#)中的。

### create-backup-vault

以下代码示例演示了如何使用 create-backup-vault。

#### AWS CLI

##### 创建备份保管库

以下create-backup-vault示例创建了具有指定名称的备份存储库。

```
aws backup create-backup-vault
```

```
--backup-vault-name sample-vault
```

此命令不生成任何输出。输出：

```
{
  "BackupVaultName": "sample-vault",
  "BackupVaultArn": "arn:aws:backup:us-west-2:123456789012:backup-vault:sample-
vault",
  "CreationDate": 1568928338.385
}
```

有关更多信息，请参阅《[备份开发者指南](#)》中的[创建AWS 备份保管库](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateBackupVault](#)中的。

## get-backup-plan-from-template

以下代码示例演示了如何使用 `get-backup-plan-from-template`。

### AWS CLI

从模板中获取现有备份计划

以下`get-backup-plan-from-template`示例从模板中获取现有备份计划，该模板指定了保留期为 35 天的每日备份。

```
aws backup get-backup-plan-from-template \
  --backup-plan-template-id "87c0c1ef-254d-4180-8fef-2e76a2c38aaa"
```

输出：

```
{
  "BackupPlanDocument": {
    "Rules": [
      {
        "RuleName": "DailyBackups",
        "ScheduleExpression": "cron(0 5 ? * * *)",
        "StartWindowMinutes": 480,
        "Lifecycle": {
          "DeleteAfterDays": 35
        }
      }
    ]
  }
}
```

```

    }
  ]
}
}

```

有关更多信息，请参阅《Backup 开发者指南》中的创建AWS 备份[计划](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBackupPlanFromTemplate](#)中的。

## get-backup-plan

以下代码示例演示了如何使用 get-backup-plan。

### AWS CLI

获取备份计划的详细信息

以下get-backup-plan示例显示了指定备份计划的详细信息。

```

aws backup get-backup-plan \
  --backup-plan-id "fcbf5d8f-bd77-4f3a-9c97-f24fb3d373a5"

```

输出：

```

{
  "BackupPlan": {
    "BackupPlanName": "Example-Backup-Plan",
    "Rules": [
      {
        "RuleName": "DailyBackups",
        "TargetBackupVaultName": "Default",
        "ScheduleExpression": "cron(0 5 ? * * *)",
        "StartWindowMinutes": 480,
        "CompletionWindowMinutes": 10080,
        "Lifecycle": {
          "DeleteAfterDays": 35
        },
        "RuleId": "70e0ccdc-e9df-4e83-82ad-c1e5a9471cc3"
      }
    ]
  },
  "BackupPlanId": "fcbf5d8f-bd77-4f3a-9c97-f24fb3d373a5",

```

```
"BackupPlanArn": "arn:aws:backup:us-west-2:123456789012:backup-plan:fcfb5d8f-  
bd77-4f3a-9c97-f24fb3d373a5",  
  "VersionId": "NjQ2ZTZkODktMGVhNy00MmQ0LWE4YjktZTkWNTQ3OTkyYTcw",  
  "CreationDate": 1568926091.57  
}
```

有关更多信息，请参阅《Backup 开发者指南》中的创建AWS 备份[计划](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBackupPlan](#)中的。

## list-backup-jobs

以下代码示例演示了如何使用 `list-backup-jobs`。

### AWS CLI

示例 1：列出所有备份任务

以下`list-backup-jobs`示例返回有关您 AWS 账户中备份任务的元数据。

```
aws backup list-backup-jobs
```

输出：

```
{  
  "BackupJobs": [  
    {  
      "BackupJobId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "BackupVaultName": "Default",  
      "BackupVaultArn": "arn:aws:backup:us-west-2:123456789012:backup-  
vault:Default",  
      "ResourceArn": "arn:aws:ec2:us-west-2:123456789012:instance/  
i-12345678901234567",  
      "CreationDate": 1600721892.929,  
      "State": "CREATED",  
      "PercentDone": "0.0",  
      "IamRoleArn": "arn:aws:iam::123456789012:role/service-role/  
AWSBackupDefaultServiceRole",  
      "StartBy": 1600725492.929,  
      "ResourceType": "EC2"  
    },  
    {  
      "BackupJobId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
```

```

        "BackupVaultName": "Default",
        "BackupVaultArn": "arn:aws:backup:us-west-2:123456789012:backup-
vault:Default",
        "RecoveryPointArn": "arn:aws:backup:us-west-2:123456789012:recovery-
point:a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
        "ResourceArn": "arn:aws:elasticfilesystem:us-west-2:123456789012:file-
system/fs-12345678",
        "CreationDate": 1600721724.77,
        "CompletionDate": 1600721744.488,
        "State": "COMPLETED",
        "PercentDone": "100.0",
        "BackupSizeInBytes": 71,
        "IamRoleArn": "arn:aws:iam::123456789012:role/service-role/
AWSBackupDefaultServiceRole",
        "StartBy": 1600725324.77,
        "ResourceType": "EFS"
    }
]
}

```

有关更多信息，请参阅 [《Backup 开发者指南》中的创建AWS备份](#)。

## 示例 2：列出已完成的备份任务

以下 `list-backup-jobs` 示例返回有关您在 AWS 账户中已完成的备份任务的元数据。

```
aws backup list-backup-jobs \
  --by-state COMPLETED
```

输出：

```

{
  "BackupJobs": [
    {
      "BackupJobId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "BackupVaultName": "Default",
      "BackupVaultArn": "arn:aws:backup:us-west-2:123456789012:backup-
vault:Default",
      "RecoveryPointArn": "arn:aws:backup:us-west-2:123456789012:recovery-
point:a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
      "ResourceArn": "arn:aws:elasticfilesystem:us-west-2:123456789012:file-
system/fs-12345678",
      "CreationDate": 1600721724.77,

```



```
        "CompletionDate": 1600721744.488,
        "State": "COMPLETED",
        "PercentDone": "100.0",
        "BackupSizeInBytes": 71,
        "IamRoleArn": "arn:aws:iam::123456789012:role/service-role/
AWSBackupDefaultServiceRole",
        "StartBy": 1600725324.77,
        "ResourceType": "EFS"
    }
]
}
```

有关更多信息，请参阅《[Backup 开发者指南](#)》中的[创建AWS备份](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBackupJobs](#)中的。

## AWS Batch 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Batch。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### cancel-job

以下代码示例演示了如何使用 cancel-job。

AWS CLI

取消作业

此示例取消具有指定作业 ID 的作业。

命令:

```
aws batch cancel-job --job-id bcf0b186-a532-4122-842e-2ccab8d54efb --reason
"Cancelling job."
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelJob](#)中的。

## create-compute-environment

以下代码示例演示了如何使用 create-compute-environment。

AWS CLI

使用按需实例创建托管计算环境

此示例创建了一个托管计算环境，该环境具有按需启动的特定 C4 实例类型。计算环境称为 C4 OnDemand。

命令:

```
aws batch create-compute-environment --cli-input-json file://<path_to_json_file>/
C4OnDemand.json
```

JSON 文件格式：

```
{
  "computeEnvironmentName": "C4OnDemand",
  "type": "MANAGED",
  "state": "ENABLED",
  "computeResources": {
    "type": "EC2",
    "minvCpus": 0,
    "maxvCpus": 128,
    "desiredvCpus": 48,
    "instanceTypes": [
      "c4.large",
      "c4.xlarge",
      "c4.2xlarge",
      "c4.4xlarge",
```

```
    "c4.8xlarge"
  ],
  "subnets": [
    "subnet-220c0e0a",
    "subnet-1a95556d",
    "subnet-978f6dce"
  ],
  "securityGroupIds": [
    "sg-cf5093b2"
  ],
  "ec2KeyPair": "id_rsa",
  "instanceRole": "ecsInstanceRole",
  "tags": {
    "Name": "Batch Instance - C4OnDemand"
  }
},
"serviceRole": "arn:aws:iam::012345678910:role/AWSBatchServiceRole"
}
```

输出：

```
{
  "computeEnvironmentName": "C4OnDemand",
  "computeEnvironmentArn": "arn:aws:batch:us-east-1:012345678910:compute-
environment/C4OnDemand"
}
```

### 使用 Spot 实例创建托管计算环境

此示例创建了一个具有 M4 实例类型的托管计算环境，该环境在竞价出价等于或低于该实例类型的按需价格的 20% 时启动。计算环境被称为 M4Spot。

命令：

```
aws batch create-compute-environment --cli-input-json file://<path_to_json_file>/
M4Spot.json
```

JSON 文件格式：

```
{
  "computeEnvironmentName": "M4Spot",
```

```
"type": "MANAGED",
"state": "ENABLED",
"computeResources": {
  "type": "SPOT",
  "spotIamFleetRole": "arn:aws:iam::012345678910:role/aws-ec2-spot-fleet-role",
  "minvCpus": 0,
  "maxvCpus": 128,
  "desiredvCpus": 4,
  "instanceTypes": [
    "m4"
  ],
  "bidPercentage": 20,
  "subnets": [
    "subnet-220c0e0a",
    "subnet-1a95556d",
    "subnet-978f6dce"
  ],
  "securityGroupIds": [
    "sg-cf5093b2"
  ],
  "ec2KeyPair": "id_rsa",
  "instanceRole": "ecsInstanceRole",
  "tags": {
    "Name": "Batch Instance - M4Spot"
  }
},
"serviceRole": "arn:aws:iam::012345678910:role/AWSBatchServiceRole"
}
```

输出：

```
{
  "computeEnvironmentName": "M4Spot",
  "computeEnvironmentArn": "arn:aws:batch:us-east-1:012345678910:compute-
environment/M4Spot"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateComputeEnvironment](#)中的。

## create-job-queue

以下代码示例演示了如何使用 create-job-queue。

## AWS CLI

### 使用单一计算环境创建低优先级作业队列

此示例创建了一个名为的作业队列 `LowPriority`，该队列使用 `M4Spot` 计算环境。

命令:

```
aws batch create-job-queue --cli-input-json file://<path_to_json_file>/  
LowPriority.json
```

JSON 文件格式：

```
{  
  "jobQueueName": "LowPriority",  
  "state": "ENABLED",  
  "priority": 10,  
  "computeEnvironmentOrder": [  
    {  
      "order": 1,  
      "computeEnvironment": "M4Spot"  
    }  
  ]  
}
```

输出：

```
{  
  "jobQueueArn": "arn:aws:batch:us-east-1:012345678910:job-queue/LowPriority",  
  "jobQueueName": "LowPriority"  
}
```

### 创建包含两个计算环境的高优先级作业队列

此示例创建了一个名为的作业队列 `HighPriority`，该队列使用阶数为 1 的 `C4 OnDemand` 计算环境和阶数为 2 的 `M4Spot` 计算环境。调度器将首先尝试在 `C4 OnDemand` 计算环境中放置作业。

命令:

```
aws batch create-job-queue --cli-input-json file://<path_to_json_file>/  
HighPriority.json
```

## JSON 文件格式：

```
{
  "jobQueueName": "HighPriority",
  "state": "ENABLED",
  "priority": 1,
  "computeEnvironmentOrder": [
    {
      "order": 1,
      "computeEnvironment": "C4OnDemand"
    },
    {
      "order": 2,
      "computeEnvironment": "M4Spot"
    }
  ]
}
```

## 输出：

```
{
  "jobQueueArn": "arn:aws:batch:us-east-1:012345678910:job-queue/HighPriority",
  "jobQueueName": "HighPriority"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateJobQueue](#)中的。

## delete-compute-environment

以下代码示例演示了如何使用 delete-compute-environment。

### AWS CLI

#### 删除计算环境

此示例删除了 P2 OnDemand 计算环境。

#### 命令：

```
aws batch delete-compute-environment --compute-environment P2OnDemand
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteComputeEnvironment](#)中的。

## delete-job-queue

以下代码示例演示了如何使用 delete-job-queue。

### AWS CLI

删除作业队列

此示例删除了 GPGPU 作业队列。

命令:

```
aws batch delete-job-queue --job-queue GPGPU
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteJobQueue](#)中的。

## deregister-job-definition

以下代码示例演示了如何使用 deregister-job-definition。

### AWS CLI

取消注册作业定义

此示例取消注册名为 sleep10 的作业定义。

命令:

```
aws batch deregister-job-definition --job-definition sleep10
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterJobDefinition](#)中的。

## describe-compute-environments

以下代码示例演示了如何使用 describe-compute-environments。

### AWS CLI

描述计算环境

此示例描述了 P2 OnDemand 计算环境。

## 命令:

```
aws batch describe-compute-environments --compute-environments P2OnDemand
```

## 输出:

```
{
  "computeEnvironments": [
    {
      "status": "VALID",
      "serviceRole": "arn:aws:iam::012345678910:role/AWSBatchServiceRole",
      "computeEnvironmentArn": "arn:aws:batch:us-east-1:012345678910:compute-environment/P2OnDemand",
      "computeResources": {
        "subnets": [
          "subnet-220c0e0a",
          "subnet-1a95556d",
          "subnet-978f6dce"
        ],
        "tags": {
          "Name": "Batch Instance - P2OnDemand"
        },
        "desiredvCpus": 48,
        "minvCpus": 0,
        "instanceTypes": [
          "p2"
        ],
        "securityGroupIds": [
          "sg-cf5093b2"
        ],
        "instanceRole": "ecsInstanceRole",
        "maxvCpus": 128,
        "type": "EC2",
        "ec2KeyPair": "id_rsa"
      },
      "statusReason": "ComputeEnvironment Healthy",
      "ecsClusterArn": "arn:aws:ecs:us-east-1:012345678910:cluster/P2OnDemand_Batch_2c06f29d-d1fe-3a49-879d-42394c86effc",
      "state": "ENABLED",
      "computeEnvironmentName": "P2OnDemand",
      "type": "MANAGED"
    }
  ]
}
```



```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeComputeEnvironments](#)中的。

## describe-job-definitions

以下代码示例演示了如何使用 describe-job-definitions。

### AWS CLI

描述有效的作业定义

此示例描述了您的所有活动任务定义。

命令:

```
aws batch describe-job-definitions --status ACTIVE
```

输出:

```
{
  "jobDefinitions": [
    {
      "status": "ACTIVE",
      "jobDefinitionArn": "arn:aws:batch:us-east-1:012345678910:job-
definition/sleep60:1",
      "containerProperties": {
        "mountPoints": [],
        "parameters": {},
        "image": "busybox",
        "environment": {},
        "vcpus": 1,
        "command": [
          "sleep",
          "60"
        ],
        "volumes": [],
        "memory": 128,
        "ulimits": []
      },
      "type": "container",
      "jobDefinitionName": "sleep60",
      "revision": 1
    }
  ]
}
```

```
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeJobDefinitions](#)中的。

## describe-job-queues

以下代码示例演示了如何使用 `describe-job-queues`。

### AWS CLI

#### 描述任务队列

此示例描述了 HighPriority 任务队列。

命令:

```
aws batch describe-job-queues --job-queues HighPriority
```

输出:

```
{  
  "jobQueues": [  
    {  
      "status": "VALID",  
      "jobQueueArn": "arn:aws:batch:us-east-1:012345678910:job-queue/  
HighPriority",  
      "computeEnvironmentOrder": [  
        {  
          "computeEnvironment": "arn:aws:batch:us-  
east-1:012345678910:compute-environment/C4OnDemand",  
          "order": 1  
        }  
      ],  
      "statusReason": "JobQueue Healthy",  
      "priority": 1,  
      "state": "ENABLED",  
      "jobQueueName": "HighPriority"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeJobQueues](#)中的。

## describe-jobs

以下代码示例演示了如何使用 `describe-jobs`。

### AWS CLI

描述一份工作

以下`describe-jobs`示例描述了具有指定作业 ID 的作业。

```
aws batch describe-jobs \  
  --jobs bcf0b186-a532-4122-842e-2ccab8d54efb
```

输出：

```
{  
  "jobs": [  
    {  
      "status": "SUBMITTED",  
      "container": {  
        "mountPoints": [],  
        "image": "busybox",  
        "environment": [],  
        "vcpus": 1,  
        "command": [  
          "sleep",  
          "60"  
        ],  
        "volumes": [],  
        "memory": 128,  
        "ulimits": []  
      },  
      "parameters": {},  
      "jobDefinition": "arn:aws:batch:us-east-1:012345678910:job-definition/  
sleep60:1",  
      "jobQueue": "arn:aws:batch:us-east-1:012345678910:job-queue/  
HighPriority",  
      "jobId": "bcf0b186-a532-4122-842e-2ccab8d54efb",  
      "dependsOn": [],  
      "jobName": "example",
```

```
        "createdAt": 1480483387803
      }
    ]
  }
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeJobs](#)中的。

## list-jobs

以下代码示例演示了如何使用 `list-jobs`。

### AWS CLI

列出正在运行的作业

此示例列出了 HighPriority 作业队列中正在运行的作业。

命令:

```
aws batch list-jobs --job-queue HighPriority
```

输出:

```
{
  "jobSummaryList": [
    {
      "jobName": "example",
      "jobId": "e66ff5fd-a1ff-4640-b1a2-0b0a142f49bb"
    }
  ]
}
```

列出已提交的工作

此示例列出了 HighPriority 作业队列中处于已提交作业状态的作业。

命令:

```
aws batch list-jobs --job-queue HighPriority --job-status SUBMITTED
```

输出:

```
{
  "jobSummaryList": [
    {
      "jobName": "example",
      "jobId": "68f0c163-fbd4-44e6-9fd1-25b14a434786"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListJobs](#)中的。

## register-job-definition

以下代码示例演示了如何使用 `register-job-definition`。

### AWS CLI

#### 注册作业定义

此示例为一个简单的容器作业注册了一个作业定义。

#### 命令:

```
aws batch register-job-definition --job-definition-name sleep30 --type container --
container-properties '{ "image": "busybox", "vcpus": 1, "memory": 128, "command":
  [ "sleep", "30"]}'
```

#### 输出:

```
{
  "jobDefinitionArn": "arn:aws:batch:us-east-1:012345678910:job-definition/
sleep30:1",
  "jobDefinitionName": "sleep30",
  "revision": 1
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterJobDefinition](#)中的。

## submit-job

以下代码示例演示了如何使用 `submit-job`。

## AWS CLI

### 提交作业

此示例向作业队列提交了一个名为 `example` 的简单容器 `HighPriority` 作业。

命令:

```
aws batch submit-job --job-name example --job-queue HighPriority --job-definition
sleep60
```

输出 :

```
{
  "jobName": "example",
  "jobId": "876da822-4198-45f2-a252-6cea32512ea8"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SubmitJob](#)中的。

### terminate-job

以下代码示例演示了如何使用 `terminate-job`。

## AWS CLI

### 终止作业

此示例终止具有指定作业 ID 的作业。

命令:

```
aws batch terminate-job --job-id 61e743ed-35e4-48da-b2de-5c8333821c84 --reason
"Terminating job."
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TerminateJob](#)中的。

### update-compute-environment

以下代码示例演示了如何使用 `update-compute-environment`。

## AWS CLI

### 更新计算环境

此示例禁用 P2 OnDemand 计算环境，因此可以将其删除。

命令:

```
aws batch update-compute-environment --compute-environment P2OnDemand --state
DISABLED
```

输出:

```
{
  "computeEnvironmentName": "P2OnDemand",
  "computeEnvironmentArn": "arn:aws:batch:us-east-1:012345678910:compute-
environment/P2OnDemand"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateComputeEnvironment](#)中的。

## update-job-queue

以下代码示例演示了如何使用 update-job-queue。

### AWS CLI

#### 更新任务队列

此示例禁用了作业队列，以便可以将其删除。

命令:

```
aws batch update-job-queue --job-queue GPGPU --state DISABLED
```

输出:

```
{
  "jobQueueArn": "arn:aws:batch:us-east-1:012345678910:job-queue/GPGPU",
```

```
"jobQueueName": "GPGPU"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateJobQueue](#)中的。

## AWS Budgets 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Budgets。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### create-budget

以下代码示例演示了如何使用 create-budget。

AWS CLI

创建成本和使用量预算

以下create-budget命令创建成本和使用量预算。

```
aws budgets create-budget \  
  --account-id 111122223333 \  
  --budget file://budget.json \  
  --notifications-with-subscribers file://notifications-with-subscribers.json
```

budget.json 的内容：



```
{
  "BudgetLimit": {
    "Amount": "100",
    "Unit": "USD"
  },
  "BudgetName": "Example Tag Budget",
  "BudgetType": "COST",
  "CostFilters": {
    "TagKeyValue": [
      "user:Key$value1",
      "user:Key$value2"
    ]
  },
  "CostTypes": {
    "IncludeCredit": true,
    "IncludeDiscount": true,
    "IncludeOtherSubscription": true,
    "IncludeRecurring": true,
    "IncludeRefund": true,
    "IncludeSubscription": true,
    "IncludeSupport": true,
    "IncludeTax": true,
    "IncludeUpfront": true,
    "UseBlended": false
  },
  "TimePeriod": {
    "Start": 1477958399,
    "End": 3706473600
  },
  "TimeUnit": "MONTHLY"
}
```

notifications-with-subscribers.json 的内容：

```
[
  {
    "Notification": {
      "ComparisonOperator": "GREATER_THAN",
      "NotificationType": "ACTUAL",
      "Threshold": 80,
      "ThresholdType": "PERCENTAGE"
    },
    "Subscribers": [
```

```
    {
      "Address": "example@example.com",
      "SubscriptionType": "EMAIL"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateBudget](#)中的。

## create-notification

以下代码示例演示了如何使用 create-notification。

### AWS CLI

为指定的“成本和使用情况”预算创建通知

此示例为指定的“成本和使用情况”预算创建通知。

命令:

```
aws budgets create-notification --account-id 111122223333 --budget-name "Example
Budget" --notification
NotificationType=ACTUAL,ComparisonOperator=GREATER_THAN,Threshold=80,ThresholdType=PERCENTA
--subscriber SubscriptionType=EMAIL,Address=example@example.com
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateNotification](#)中的。

## create-subscriber

以下代码示例演示了如何使用 create-subscriber。

### AWS CLI

为与成本和使用量预算关联的通知创建订阅者

此示例为指定通知创建订阅者。

命令:

```
aws budgets create-subscriber --account-id 111122223333 --budget-name "Example
Budget" --notification
```

```
NotificationType=ACTUAL,ComparisonOperator=GREATER_THAN,Threshold=80,ThresholdType=PERCENTAGE
--subscriber SubscriptionType=EMAIL,Address=example@example.com
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSubscriber](#)中的。

## delete-budget

以下代码示例演示了如何使用 delete-budget。

### AWS CLI

删除成本和使用量预算

此示例删除了指定的成本和使用量预算。

命令:

```
aws budgets delete-budget --account-id 111122223333 --budget-name "Example Budget"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBudget](#)中的。

## delete-notification

以下代码示例演示了如何使用 delete-notification。

### AWS CLI

从预算中删除通知

此示例从指定预算中删除指定的通知。

命令:

```
aws budgets delete-notification --account-id 111122223333 --budget-name "Example
Budget" --notification
NotificationType=ACTUAL,ComparisonOperator=GREATER_THAN,Threshold=80,ThresholdType=PERCENTAGE
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNotification](#)中的。

## delete-subscriber

以下代码示例演示了如何使用 delete-subscriber。

## AWS CLI

### 从通知中删除订阅者

此示例将从指定通知中删除指定的订阅者。

命令:

```
aws budgets delete-subscriber --account-id 111122223333 --budget-name "Example Budget" --notification NotificationType=ACTUAL,ComparisonOperator=GREATER_THAN,Threshold=80,ThresholdType=PERCENTAGE --subscriber SubscriptionType=EMAIL,Address=example@example.com
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSubscriber](#)中的。

## describe-budget

以下代码示例演示了如何使用 describe-budget。

## AWS CLI

### 检索与账户关联的预算

此示例检索指定的成本和使用量预算。

命令:

```
aws budgets describe-budget --account-id 111122223333 --budget-name "Example Budget"
```

输出:

```
{
  "Budget": {
    "CalculatedSpend": {
      "ForecastedSpend": {
        "Amount": "2641.548000000000022919266484677791595458984375",
        "Unit": "USD"
      },
      "ActualSpend": {
        "Amount": "604.45600000000000172803993336856365203857421875",
        "Unit": "USD"
      }
    }
  }
}
```

```
    }
  },
  "BudgetType": "COST",
  "BudgetLimit": {
    "Amount": "100",
    "Unit": "USD"
  },
  "BudgetName": "Example Budget",
  "CostTypes": {
    "IncludeOtherSubscription": true,
    "IncludeUpfront": true,
    "IncludeRefund": true,
    "UseBlended": false,
    "IncludeDiscount": true,
    "UseAmortized": false,
    "IncludeTax": true,
    "IncludeCredit": true,
    "IncludeSupport": true,
    "IncludeRecurring": true,
    "IncludeSubscription": true
  },
  "TimeUnit": "MONTHLY",
  "TimePeriod": {
    "Start": 1477958399.0,
    "End": 3706473600.0
  },
  "CostFilters": {
    "AZ": [
      "us-east-1"
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeBudget](#)中的。

## describe-budgets

以下代码示例演示了如何使用 describe-budgets。

### AWS CLI

#### 检索与账户关联的预算

此示例检索账户的成本和使用量预算。

命令:

```
aws budgets describe-budgets --account-id 111122223333 --max-results 20
```

输出:

```
{
  "Budgets": [
    {
      "CalculatedSpend": {
        "ForecastedSpend": {
          "Amount": "2641.548000000000022919266484677791595458984375",
          "Unit": "USD"
        },
        "ActualSpend": {
          "Amount": "604.45600000000000172803993336856365203857421875",
          "Unit": "USD"
        }
      },
      "BudgetType": "COST",
      "BudgetLimit": {
        "Amount": "100",
        "Unit": "USD"
      },
      "BudgetName": "Example Budget",
      "CostTypes": {
        "IncludeOtherSubscription": true,
        "IncludeUpfront": true,
        "IncludeRefund": true,
        "UseBlended": false,
        "IncludeDiscount": true,
        "UseAmortized": false,
        "IncludeTax": true,
        "IncludeCredit": true,
        "IncludeSupport": true,
        "IncludeRecurring": true,
        "IncludeSubscription": true
      },
      "TimeUnit": "MONTHLY",
      "TimePeriod": {
        "Start": 1477958399.0,
```

```
        "End": 3706473600.0
      },
      "CostFilters": {
        "AZ": [
          "us-east-1"
        ]
      }
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeBudgets](#)中的。

## describe-notifications-for-budget

以下代码示例演示了如何使用 `describe-notifications-for-budget`。

### AWS CLI

#### 检索预算通知

此示例检索成本和使用量预算的通知。

命令:

```
aws budgets describe-notifications-for-budget --account-id 111122223333 --budget-name "Example Budget" --max-results 5
```

输出:

```
{
  "Notifications": [
    {
      "Threshold": 80.0,
      "ComparisonOperator": "GREATER_THAN",
      "NotificationType": "ACTUAL"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeNotificationsForBudget](#)中的。

## describe-subscribers-for-notification

以下代码示例演示了如何使用 `describe-subscribers-for-notification`。

### AWS CLI

检索预算通知的订阅者

此示例检索成本和使用量预算通知的订阅者。

命令:

```
aws budgets describe-subscribers-for-notification --account-id 111122223333 --budget-name "Example Budget" --notification NotificationType=ACTUAL,ComparisonOperator=GREATER_THAN,Threshold=80,ThresholdType=PERCENTAGE --max-results 5
```

输出:

```
{
  "Subscribers": [
    {
      "SubscriptionType": "EMAIL",
      "Address": "example2@example.com"
    },
    {
      "SubscriptionType": "EMAIL",
      "Address": "example@example.com"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSubscribersForNotification](#)中的。

## update-budget

以下代码示例演示了如何使用 `update-budget`。

### AWS CLI

替换成本和使用情况预算的预算

此示例将成本和使用量预算替换为新预算。



**命令:**

```
aws budgets update-budget --account-id 111122223333 --new-budget file://new-budget.json
```

**新预算.json :**

```
{
  "BudgetLimit": {
    "Amount": "100",
    "Unit": "USD"
  },
  "BudgetName": "Example Budget",
  "BudgetType": "COST",
  "CostFilters": {
    "AZ" : [ "us-east-1" ]
  },
  "CostTypes": {
    "IncludeCredit": false,
    "IncludeDiscount": true,
    "IncludeOtherSubscription": true,
    "IncludeRecurring": true,
    "IncludeRefund": true,
    "IncludeSubscription": true,
    "IncludeSupport": true,
    "IncludeTax": true,
    "IncludeUpfront": true,
    "UseBlended": false,
    "UseAmortized": true
  },
  "TimePeriod": {
    "Start": 1477958399,
    "End": 3706473600
  },
  "TimeUnit": "MONTHLY"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateBudget](#)中的。

**update-notification**

以下代码示例演示了如何使用 update-notification。

## AWS CLI

### 替换成本和使用量预算的通知

此示例将成本和使用率预算的 80% 通知替换为 90% 的通知。

命令:

```
aws budgets update-notification --account-id 111122223333 --budget-name "Example Budget" --old-notification NotificationType=ACTUAL,ComparisonOperator=GREATER_THAN,Threshold=80,ThresholdType=PERCENTAGE --new-notification NotificationType=ACTUAL,ComparisonOperator=GREATER_THAN,Threshold=90,ThresholdType=PERCENTAGE
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateNotification](#)中的。

## update-subscriber

以下代码示例演示了如何使用 update-subscriber。

## AWS CLI

### 根据成本和使用量预算替换订阅者

此示例取代了订阅者的“成本和使用情况”预算。

命令:

```
aws budgets update-subscriber --account-id 111122223333 --budget-name "Example Budget" --notification NotificationType=ACTUAL,ComparisonOperator=GREATER_THAN,Threshold=80,ThresholdType=PERCENTAGE --old-subscriber SubscriptionType=EMAIL,Address=example@example.com --new-subscriber SubscriptionType=EMAIL,Address=example2@example.com
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateSubscriber](#)中的。

## 使用 Amazon Chime 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon Chime 搭配使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-phone-number-with-user**

以下代码示例演示了如何使用 `associate-phone-number-with-user`。

AWS CLI

将电话号码与用户关联

以下 `associate-phone-number-with-user` 示例将指定的电话号码与用户关联。

```
aws chime associate-phone-number-with-user \  
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
  --user-id 1ab2345c-67de-8901-f23g-45h678901j2k \  
  --e164-phone-number "+12065550100"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Chime 管理指南中的管理 [用户电话号码](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociatePhoneNumberWithUser](#) 中的。

### **associate-phone-numbers-with-voice-connector-group**

以下代码示例演示了如何使用 `associate-phone-numbers-with-voice-connector-group`。

AWS CLI

将电话号码与 Amazon Chime 语音连接器群组关联

以下 `associate-phone-numbers-with-voice-connector-group` 示例将指定的电话号码与 Amazon Chime 语音连接器组相关联。

```
aws chime associate-phone-numbers-with-voice-connector-group \  
  --voice-connector-group-id 123a456b-c7d8-90e1-fg23-4h567jkl8901 \  
  --e164-phone-numbers "+12065550100" "+12065550101" \  
  --force-associate
```

输出：

```
{  
  "PhoneNumberErrors": []  
}
```

有关更多信息，请参阅 [《亚马逊 Chime 管理指南》](#) 中的“使用 Amazon Chime 语音连接器群组”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociatePhoneNumbersWithVoiceConnectorGroup](#) 中的。

## associate-phone-numbers-with-voice-connector

以下代码示例演示了如何使用 `associate-phone-numbers-with-voice-connector`。

### AWS CLI

将电话号码与 Amazon Chime 语音连接器关联

以下 `associate-phone-numbers-with-voice-connector` 示例将指定的电话号码与 Amazon Chime 语音连接器相关联。

```
aws chime associate-phone-numbers-with-voice-connector \  
  --voice-connector-id abcdef1ghij2klmno3pqr4 \  
  --e164-phone-numbers "+12065550100" "+12065550101" \  
  --force-associate
```

输出：

```
{  
  "PhoneNumberErrors": []  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[使用 Amazon Chime 语音连接器](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociatePhoneNumbersWithVoiceConnector](#)中的。

## associate-signin-delegate-groups-with-account

以下代码示例演示了如何使用 `associate-signin-delegate-groups-with-account`。

### AWS CLI

#### 关联登录代表群组

以下`associate-signin-delegate-groups-with-account`示例将指定的登录委托群组与指定的 Amazon Chime 账户关联起来。

```
aws chime associate-signin-delegate-groups-with-account \  
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
  --signin-delegate-groups GroupName=my_users
```

此命令不生成任何输出。

有关更多信息，请参阅《[Amazon Chime 管理指南](#)》中的管理[用户访问和权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateSigninDelegateGroupsWithAccount](#)中的。

## batch-create-room-membership

以下代码示例演示了如何使用 `batch-create-room-membership`。

### AWS CLI

#### 创建多个房间成员资格

以下`batch-create-room-membership`示例将多个用户作为聊天室成员添加到聊天室。它还为用户分配管理员和成员角色。

```
aws chime batch-create-room-membership \  
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
  --room-id abcd1e2d-3e45-6789-01f2-3g45h67i890j \  
  --members
```

```
--membership-item-list "MemberId=1ab2345c-67de-8901-  
f23g-45h678901j2k,Role=Administrator" "MemberId=2ab2345c-67de-8901-  
f23g-45h678901j2k,Role=Member"
```

输出：

```
{  
  "ResponseMetadata": {  
    "RequestId": "169ba401-d886-475f-8b3f-e01eac6fadfb",  
    "HTTPStatusCode": 201,  
    "HTTPHeaders": {  
      "x-amzn-requestid": "169ba401-d886-475f-8b3f-e01eac6fadfb",  
      "content-type": "application/json",  
      "content-length": "13",  
      "date": "Mon, 02 Dec 2019 22:46:58 GMT",  
      "connection": "keep-alive"  
    },  
    "RetryAttempts": 0  
  },  
  "Errors": []  
}
```

有关更多信息，请参阅 Amazon Chime 用户指南中的[创建聊天室](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[BatchCreateRoomMembership](#)中的。

## batch-delete-phone-number

以下代码示例演示了如何使用 batch-delete-phone-number。

### AWS CLI

删除多个电话号码

以下 batch-delete-phone-number 示例删除了所有指定的电话号码。

```
aws chime batch-delete-phone-number \  
  --phone-number-ids "%2B12065550100" "%2B12065550101"
```

此命令不生成任何输出。输出：

```
{
```

```
"PhoneNumberErrors": []
}
```

有关更多信息，请参阅 Amazon Chime 管理指南中的[使用电话号码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchDeletePhoneNumber](#)中的。

## batch-suspend-user

以下代码示例演示了如何使用 batch-suspend-user。

### AWS CLI

暂停多个用户

以下batch-suspend-user示例暂停列出的用户使用指定 Amazon Chime 账户。

```
aws chime batch-suspend-user \
  --account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \
  --user-id-list "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE" "a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE" "a1b2c3d4-5678-90ab-cdef-44444EXAMPLE"
```

输出：

```
{
  "UserErrors": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchSuspendUser](#)中的。

## batch-unsuspend-user

以下代码示例演示了如何使用 batch-unsuspend-user。

### AWS CLI

取消暂停多个用户

以下batch-unsuspend-user示例取消了之前对指定 Amazon Chime 账户中列出的用户的任何暂停。

```
aws chime batch-unsuspend-user \
```

```
--account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \  
--user-id-list "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE" "a1b2c3d4-5678-90ab-  
cdef-33333EXAMPLE" "a1b2c3d4-5678-90ab-cdef-44444EXAMPLE"
```

输出：

```
{  
  "UserErrors": []  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchUnsuspendUser](#)中的。

## batch-update-phone-number

以下代码示例演示了如何使用 batch-update-phone-number。

### AWS CLI

同时更新多个电话号码产品类型

以下batch-update-phone-number示例更新了所有指定电话号码的产品类型。

```
aws chime batch-update-phone-number \  
  --update-phone-number-request-items PhoneNumberId=  
%2B12065550100,ProductType=BusinessCalling PhoneNumberId=  
%2B12065550101,ProductType=BusinessCalling
```

输出：

```
{  
  "PhoneNumberErrors": []  
}
```

要同时更新多个电话号码呼叫姓名

以下batch-update-phone-number示例更新了所有指定电话号码的呼叫者姓名。

```
aws chime batch-update-phone-number \  
  --update-phone-number-request-items PhoneNumberId=  
%2B14013143874,CallingName=phonenumber1 PhoneNumberId=  
%2B14013144061,CallingName=phonenumber2
```



输出：

```
{
  "PhoneNumberErrors": []
}
```

有关更多信息，请参阅 Amazon Chime 管理指南中的[使用电话号码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchUpdatePhoneNumber](#)中的。

## batch-update-user

以下代码示例演示了如何使用 batch-update-user。

### AWS CLI

使用单个命令更新多个用户

以下batch-update-user示例更新了LicenseType指定 Amazon Chime 账户中列出的每位用户的。

```
aws chime batch-update-user \
  --account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
  --update-user-request-items "UserId=a1b2c3d4-5678-90ab-
cdef-22222EXAMPLE,LicenseType=Basic" "UserId=a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE,LicenseType=Basic"
```

输出：

```
{
  "UserErrors": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchUpdateUser](#)中的。

## create-account

以下代码示例演示了如何使用 create-account。

### AWS CLI

创建账户

以下create-account示例在管理员账户下创建了一个 Amazon Chime 账户。AWS

```
aws chime create-account \  
  --name MyChimeAccount
```

输出：

```
{  
  "Account": {  
    "AwsAccountId": "111122223333",  
    "AccountId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
    "Name": "MyChimeAccount",  
    "AccountType": "Team",  
    "CreatedTimestamp": "2019-01-04T17:11:22.003Z",  
    "DefaultLicense": "Pro",  
    "SupportedLicenses": [  
      "Basic",  
      "Pro"  
    ],  
    "SignInDelegateGroups": [  
      {  
        "GroupName": "myGroup"  
      },  
    ]  
  }  
}
```

有关更多信息，请参阅《Amazon Chime 管理指南》中的[入门指南](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAccount](#)中的。

## create-bot

以下代码示例演示了如何使用 create-bot。

### AWS CLI

创建 Amazon Chime 机器人

以下create-bot示例为指定的 Amazon Chime Enterprise 账户创建了一个机器人。

```
aws chime create-bot \  
  --name MyChimeBot
```

```
--account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
--display-name "myBot" \  
--domain "example.com"
```

输出：

```
{  
  "Bot": {  
    "BotId": "123abcd4-5ef6-789g-0h12-34j56789012k",  
    "UserId": "123abcd4-5ef6-789g-0h12-34j56789012k",  
    "DisplayName": "myBot (Bot)",  
    "BotType": "ChatBot",  
    "Disabled": false,  
    "CreatedTimestamp": "2019-09-09T18:05:56.749Z",  
    "UpdatedTimestamp": "2019-09-09T18:05:56.749Z",  
    "BotEmail": "myBot-chimebot@example.com",  
    "SecurityToken": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"  
  }  
}
```

有关更多信息，请参阅《亚马逊 Chime 开发者指南》中的将[聊天机器人与 Amazon Chime 集成](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateBot](#)中的。

## create-phone-number-order

以下代码示例演示了如何使用 create-phone-number-order。

### AWS CLI

#### 创建电话号码订单

以下create-phone-number-order示例为指定的电话号码创建电话号码顺序。

```
aws chime create-phone-number-order \  
  --product-type VoiceConnector \  
  --e164-phone-numbers "+12065550100" "+12065550101" "+12065550102"
```

输出：

```
{
```

```
"PhoneNumberOrder": {
  "PhoneNumberOrderId": "abc12345-de67-89f0-123g-h45i678j9012",
  "ProductType": "VoiceConnector",
  "Status": "Processing",
  "OrderedPhoneNumbers": [
    {
      "E164PhoneNumber": "+12065550100",
      "Status": "Processing"
    },
    {
      "E164PhoneNumber": "+12065550101",
      "Status": "Processing"
    },
    {
      "E164PhoneNumber": "+12065550102",
      "Status": "Processing"
    }
  ],
  "CreatedTimestamp": "2019-08-09T21:35:21.427Z",
  "UpdatedTimestamp": "2019-08-09T21:35:22.408Z"
}
}
```

有关更多信息，请参阅 Amazon Chime 管理指南中的[使用电话号码](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreatePhoneNumberOrder](#)中的。

## create-proxy-session

以下代码示例演示了如何使用 create-proxy-session。

### AWS CLI

#### 创建代理会话

以下 create-proxy-session 示例创建具有语音和短信功能的代理会话。

```
aws chime create-proxy-session \
  --voice-connector-id abcdef1ghij2klmno3pqr4 \
  --participant-phone-numbers "+14015550101" "+12065550100" \
  --capabilities "Voice" "SMS"
```

输出：

```
{
  "ProxySession": {
    "VoiceConnectorId": "abcdef1ghij2klmno3pqr4",
    "ProxySessionId": "123a4bc5-67d8-901e-2f3g-h4ghjk567891",
    "Status": "Open",
    "ExpiryMinutes": 60,
    "Capabilities": [
      "SMS",
      "Voice"
    ],
    "CreatedTimestamp": "2020-04-15T16:10:10.288Z",
    "UpdatedTimestamp": "2020-04-15T16:10:10.288Z",
    "Participants": [
      {
        "PhoneNumber": "+12065550100",
        "ProxyPhoneNumber": "+19135550199"
      },
      {
        "PhoneNumber": "+14015550101",
        "ProxyPhoneNumber": "+19135550199"
      }
    ]
  }
}
```

有关更多信息，请参阅 Amazon Chime 开发者指南中的[代理电话会话](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateProxySession](#) 中的。

## create-room-membership

以下代码示例演示了如何使用 create-room-membership。

### AWS CLI

#### 创建聊天室成员资格

以下 create-room-membership 示例将指定用户作为聊天室成员添加到聊天室。

```
aws chime create-room-membership \
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \
  --room-id abcd1e2d-3e45-6789-01f2-3g45h67i890j \
```

```
--member-id 1ab2345c-67de-8901-f23g-45h678901j2k
```

输出：

```
{
  "RoomMembership": {
    "RoomId": "abcd1e2d-3e45-6789-01f2-3g45h67i890j",
    "Member": {
      "MemberId": "1ab2345c-67de-8901-f23g-45h678901j2k",
      "MemberType": "User",
      "Email": "janed@example.com",
      "FullName": "Jane Doe",
      "AccountId": "12a3456b-7c89-012d-3456-78901e23fg45"
    },
    "Role": "Member",
    "InvitedBy": "arn:aws:iam::111122223333:user/alejandro",
    "UpdatedTimestamp": "2019-12-02T22:36:41.969Z"
  }
}
```

有关更多信息，请参阅 Amazon Chime 用户指南中的[创建聊天室](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRoomMembership](#)中的。

## create-room

以下代码示例演示了如何使用 create-room。

### AWS CLI

#### 创建聊天室

以下create-room示例为指定的 Amazon Chime 账户创建聊天室。

```
aws chime create-room \
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \
  --name chatRoom
```

输出：

```
{
  "Room": {
```

```

    "RoomId": "abcd1e2d-3e45-6789-01f2-3g45h67i890j",
    "Name": "chatRoom",
    "AccountId": "12a3456b-7c89-012d-3456-78901e23fg45",
    "CreatedBy": "arn:aws:iam::111122223333:user/alejandro",
    "CreatedTimestamp": "2019-12-02T22:29:31.549Z",
    "UpdatedTimestamp": "2019-12-02T22:29:31.549Z"
  }
}

```

有关更多信息，请参阅 Amazon Chime 用户指南中的[创建聊天室](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateRoom](#)中的。

## create-user

以下代码示例演示了如何使用 create-user。

### AWS CLI

为共享设备创建用户配置文件

以下 create-user 示例为指定的电子邮件地址创建共享设备配置文件。

```

aws chime create-user \
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \
  --email roomdevice@example.com \
  --user-type SharedDevice

```

输出：

```

{
  "User": {
    "UserId": "1ab2345c-67de-8901-f23g-45h678901j2k",
    "AccountId": "12a3456b-7c89-012d-3456-78901e23fg45",
    "PrimaryEmail": "roomdevice@example.com",
    "DisplayName": "Room Device",
    "LicenseType": "Pro",
    "UserType": "SharedDevice",
    "UserRegistrationStatus": "Registered",
    "RegisteredOn": "2020-01-15T22:38:09.806Z",
    "AlexaForBusinessMetadata": {
      "IsAlexaForBusinessEnabled": false
    }
  }
}

```

```
}  
}
```

有关更多信息，请参阅《Amazon Chime 管理指南》中的“[准备安装](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUser](#)中的。

## create-voice-connector-group

以下代码示例演示了如何使用 create-voice-connector-group。

### AWS CLI

创建 Amazon Chime 语音连接器群组

以下create-voice-connector-group示例创建了一个 Amazon Chime 语音连接器组，其中包括指定的 Amazon Chime 语音连接器。

```
aws chime create-voice-connector-group \  
  --name myGroup \  
  --voice-connector-items VoiceConnectorId=abcdefghijklmno3pqr4,Priority=2
```

输出：

```
{  
  "VoiceConnectorGroup": {  
    "VoiceConnectorGroupId": "123a456b-c7d8-90e1-fg23-4h567jk18901",  
    "Name": "myGroup",  
    "VoiceConnectorItems": [],  
    "CreatedTimestamp": "2019-09-18T16:38:34.734Z",  
    "UpdatedTimestamp": "2019-09-18T16:38:34.734Z"  
  }  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[使用亚马逊 Chime 语音连接器组](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVoiceConnectorGroup](#)中的。

## create-voice-connector

以下代码示例演示了如何使用 create-voice-connector。



## AWS CLI

### 创建 Amazon Chime 语音连接器

以下`create-voice-connector`示例在指定 AWS 区域创建启用加密的 Amazon Chime 语音连接器。

```
aws chime create-voice-connector \  
  --name newVoiceConnector \  
  --aws-region us-west-2 \  
  --require-encryption
```

输出：

```
{  
  "VoiceConnector": {  
    "VoiceConnectorId": "abcdef1ghij2klmno3pqr4",  
    "AwsRegion": "us-west-2",  
    "Name": "newVoiceConnector",  
    "OutboundHostName": "abcdef1ghij2klmno3pqr4.voiceconnector.chime.aws",  
    "RequireEncryption": true,  
    "CreatedTimestamp": "2019-09-18T20:34:01.352Z",  
    "UpdatedTimestamp": "2019-09-18T20:34:01.352Z"  
  }  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[使用 Amazon Chime 语音连接器](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateVoiceConnector](#) 中的。

## delete-account

以下代码示例演示了如何使用 `delete-account`。

### AWS CLI

#### 删除账户

以下`delete-account`示例删除了指定的账户。

```
aws chime delete-account --account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Chime 管理指南》中的[“删除您的账户”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAccount](#)中的。

## delete-phone-number

以下代码示例演示了如何使用 delete-phone-number。

### AWS CLI

删除电话号码

以下delete-phone-number示例将指定的电话号码移入删除队列。

```
aws chime delete-phone-number \  
  --phone-number-id "+12065550100"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Chime 管理指南中的[使用电话号码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePhoneNumber](#)中的。

## delete-proxy-session

以下代码示例演示了如何使用 delete-proxy-session。

### AWS CLI

删除代理会话

以下delete-proxy-session示例删除了指定的代理会话。

```
aws chime delete-proxy-session \  
  --voice-connector-id abcdef1ghij2klmno3pqr4 \  
  --proxy-session-id 123a4bc5-67d8-901e-2f3g-h4ghjk567891
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Chime 开发者指南中的[代理电话会话](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteProxySession](#)中的。

## delete-room-membership

以下代码示例演示了如何使用 delete-room-membership。

### AWS CLI

删除聊天室成员的用户

以下delete-room-membership示例将指定成员从指定的聊天室中移除。

```
aws chime delete-room-membership \  
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
  --room-id abcd1e2d-3e45-6789-01f2-3g45h67i890j \  
  --member-id 1ab2345c-67de-8901-f23g-45h678901j2k
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Chime 用户指南中的[创建聊天室](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRoomMembership](#)中的。

## delete-room

以下代码示例演示了如何使用 delete-room。

### AWS CLI

删除聊天室

以下delete-room示例删除了指定的聊天室并删除了聊天室成员资格。

```
aws chime delete-room \  
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
  --room-id abcd1e2d-3e45-6789-01f2-3g45h67i890j
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Chime 用户指南中的[创建聊天室](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRoom](#)中的。

## delete-voice-connector-group

以下代码示例演示了如何使用 delete-voice-connector-group。

## AWS CLI

### title

以下delete-voice-connector-group示例删除指定的 Amazon Chime 语音连接器组。

```
aws chime delete-voice-connector-group \  
  --voice-connector-group-id 123a456b-c7d8-90e1-fg23-4h567jkl8901
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[使用亚马逊 Chime 语音连接器组](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVoiceConnectorGroup](#)中的。

## delete-voice-connector-origination

以下代码示例演示了如何使用 delete-voice-connector-origination。

## AWS CLI

### 删除起源设置

以下delete-voice-connector-origination示例从指定的 Amazon Chime 语音连接器中删除源主机、端口、协议、优先级和权重。

```
aws chime delete-voice-connector-origination \  
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[使用 Amazon Chime 语音连接器](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVoiceConnectorOrigination](#)中的。

## delete-voice-connector-proxy

以下代码示例演示了如何使用 delete-voice-connector-proxy。

## AWS CLI

### 删除代理配置

以下delete-voice-connector-proxy示例从您的 Amazon Chime 语音连接器中删除代理配置。

```
aws chime delete-voice-connector-proxy \  
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Chime 开发者指南中的[代理电话会话](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVoiceConnectorProxy](#)中的。

## delete-voice-connector-streaming-configuration

以下代码示例演示了如何使用 delete-voice-connector-streaming-configuration。

### AWS CLI

#### 删除直播配置

以下delete-voice-connector-streaming-configuration示例删除了指定 Amazon Chime Voice Connector 的直播配置。

```
aws chime delete-voice-connector-streaming-configuration \  
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[将 Amazon Chime 语音连接器数据流式传输到 Kinesis](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVoiceConnectorStreamingConfiguration](#)中的。

## delete-voice-connector-termination-credentials

以下代码示例演示了如何使用 delete-voice-connector-termination-credentials。

### AWS CLI

#### 删除终止凭证

以下`delete-voice-connector-termination-credentials`示例删除了指定用户名和 Amazon Chime Voice Connector 的终止凭证。

```
aws chime delete-voice-connector-termination-credentials \  
  --voice-connector-id abcdef1ghij2klmno3pqr4 \  
  --usernames "jdoe"
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“使用 Amazon Chime 语音连接器”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVoiceConnectorTerminationCredentials](#)中的。

## **delete-voice-connector-termination**

以下代码示例演示了如何使用 `delete-voice-connector-termination`。

### AWS CLI

#### 删除终止设置

以下`delete-voice-connector-termination`示例删除了指定 Amazon Chime 语音连接器的终止设置。

```
aws chime delete-voice-connector-termination \  
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“使用 Amazon Chime 语音连接器”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVoiceConnectorTermination](#)中的。

## **delete-voice-connector**

以下代码示例演示了如何使用 `delete-voice-connector`。

### AWS CLI

#### 删除 Amazon Chime 语音连接器

以下delete-voice-connector示例就是这样做的

```
aws chime delete-voice-connector \  
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“使用 Amazon Chime 语音连接器”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVoiceConnector](#)中的。

## disassociate-phone-number-from-user

以下代码示例演示了如何使用 disassociate-phone-number-from-user。

### AWS CLI

取消电话号码与用户的关联

以下disassociate-phone-number-from-user示例取消电话号码与指定用户的关联。

```
aws chime disassociate-phone-number-from-user \  
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
  --user-id 1ab2345c-67de-8901-f23g-45h678901j2k
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Chime 管理指南中的管理[用户电话号码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociatePhoneNumberFromUser](#)中的。

## disassociate-phone-numbers-from-voice-connector-group

以下代码示例演示了如何使用 disassociate-phone-numbers-from-voice-connector-group。

### AWS CLI

取消电话号码与 Amazon Chime 语音连接器群组的关联

以下disassociate-phone-numbers-from-voice-connector-group示例取消指定电话号码与 Amazon Chime 语音连接器组的关联。

```
aws chime disassociate-phone-numbers-from-voice-connector-group \  
  --voice-connector-group-id 123a456b-c7d8-90e1-fg23-4h567jkl8901 \  
  --e164-phone-numbers "+12065550100" "+12065550101"
```

输出：

```
{  
  "PhoneNumberErrors": []  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[使用亚马逊 Chime 语音连接器组](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociatePhoneNumbersFromVoiceConnectorGroup](#)中的。

## disassociate-phone-numbers-from-voice-connector

以下代码示例演示了如何使用 disassociate-phone-numbers-from-voice-connector。

### AWS CLI

取消电话号码与 Amazon Chime 语音连接器的关联

以下disassociate-phone-numbers-from-voice-connector示例取消指定电话号码与 Amazon Chime 语音连接器的关联。

```
aws chime disassociate-phone-numbers-from-voice-connector \  
  --voice-connector-id abcdef1ghij2klmno3pqr4 \  
  --e164-phone-numbers "+12065550100" "+12065550101"
```

输出：

```
{  
  "PhoneNumberErrors": []  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[使用 Amazon Chime 语音连接器](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociatePhoneNumbersFromVoiceConnector](#)中的。



## disassociate-signin-delegate-groups-from-account

以下代码示例演示了如何使用 `disassociate-signin-delegate-groups-from-account`。

### AWS CLI

取消关联登录代表群组

以下 `disassociate-signin-delegate-groups-from-account` 示例取消指定登录委托群组与指定的 Amazon Chime 账户的关联。

```
aws chime disassociate-signin-delegate-groups-from-account \  
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
  --group-names "my_users"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Chime 管理指南》中的管理[用户访问和权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateSigninDelegateGroupsFromAccount](#) 中的。

## get-account-settings

以下代码示例演示了如何使用 `get-account-settings`。

### AWS CLI

检索账户的设置

以下 `get-account-settings` 示例检索指定账户的账户设置。

```
aws chime get-account-settings --account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
```

输出：

```
{  
  "AccountSettings": {  
    "DisableRemoteControl": false,  
    "EnableDialOut": false  
  }  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[管理您的亚马逊 Chime 账户](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAccountSettings](#)中的。

## get-account

以下代码示例演示了如何使用 get-account。

### AWS CLI

检索账户的详细信息

以下get-account示例检索指定的 Amazon Chime 账户的详细信息。

```
aws chime get-account \  
  --account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
```

输出：

```
{  
  "Account": {  
    "AwsAccountId": "111122223333",  
    "AccountId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
    "Name": "EnterpriseDirectory",  
    "AccountType": "EnterpriseDirectory",  
    "CreatedTimestamp": "2018-12-20T18:38:02.181Z",  
    "DefaultLicense": "Pro",  
    "SupportedLicenses": [  
      "Basic",  
      "Pro"  
    ],  
    "SigninDelegateGroups": [  
      {  
        "GroupName": "myGroup"  
      },  
    ]  
  }  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[管理您的亚马逊 Chime 账户](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAccount](#)中的。

## get-bot

以下代码示例演示了如何使用 get-bot。

### AWS CLI

检索有关机器人的详细信息

以下get-bot示例显示了指定机器人的详细信息。

```
aws chime get-bot \  
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
  --bot-id 123abcd4-5ef6-789g-0h12-34j56789012k
```

输出：

```
{  
  "Bot": {  
    "BotId": "123abcd4-5ef6-789g-0h12-34j56789012k",  
    "UserId": "123abcd4-5ef6-789g-0h12-34j56789012k",  
    "DisplayName": "myBot (Bot)",  
    "BotType": "ChatBot",  
    "Disabled": false,  
    "CreatedTimestamp": "2019-09-09T18:05:56.749Z",  
    "UpdatedTimestamp": "2019-09-09T18:05:56.749Z",  
    "BotEmail": "myBot-chimebot@example.com",  
    "SecurityToken": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"  
  }  
}
```

有关更多信息，请参阅 Amazon Chime 开发者指南中的[更新聊天机器人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBot](#)中的。

## get-global-settings

以下代码示例演示了如何使用 get-global-settings。

### AWS CLI

获取全局设置

以下`get-global-settings`示例检索用于存储与管理员账户关联的 Amazon Chime Business Calling 和 Amazon Chime 语音连接器的通话详细记录的 S3 存储桶名称。AWS

```
aws chime get-global-settings
```

输出：

```
{
  "BusinessCalling": {
    "CdrBucket": "s3bucket"
  },
  "VoiceConnector": {
    "CdrBucket": "s3bucket"
  }
}
```

有关更多信息，请参阅 Amazon Chime 管理指南中的管理全局[设置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetGlobalSettings](#)中的。

## get-phone-number-order

以下代码示例演示了如何使用 `get-phone-number-order`。

### AWS CLI

要获取电话号码订单的详细信息

以下`get-phone-number-order`示例显示了指定电话号码顺序的详细信息。

```
aws chime get-phone-number-order \
  --phone-number-order-id abc12345-de67-89f0-123g-h45i678j9012
```

输出：

```
{
  "PhoneNumberOrder": {
    "PhoneNumberOrderId": "abc12345-de67-89f0-123g-h45i678j9012",
    "ProductType": "VoiceConnector",
    "Status": "Partial",
    "OrderedPhoneNumbers": [
```

```
{
  "E164PhoneNumber": "+12065550100",
  "Status": "Acquired"
},
{
  "E164PhoneNumber": "+12065550101",
  "Status": "Acquired"
},
{
  "E164PhoneNumber": "+12065550102",
  "Status": "Failed"
}
],
"CreatedTimestamp": "2019-08-09T21:35:21.427Z",
"UpdatedTimestamp": "2019-08-09T21:35:31.926Z"
}
```

有关更多信息，请参阅 Amazon Chime 管理指南中的[使用电话号码](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetPhoneNumberOrder](#)中的。

## get-phone-number-settings

以下代码示例演示了如何使用 get-phone-number-settings。

### AWS CLI

检索出站呼叫者姓名

以下 get-phone-number-settings 示例检索主叫用户 AWS 账户的默认出站呼叫名称。

```
aws chime get-phone-number-settings
```

此命令不生成任何输出。输出：

```
{
  "CallingName": "myName",
  "CallingNameUpdatedTimestamp": "2019-10-28T18:56:42.911Z"
}
```

有关更多信息，请参阅 Amazon Chime 管理指南中的[使用电话号码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPhoneNumberSettings](#)中的。

## get-phone-number

以下代码示例演示了如何使用 get-phone-number。

### AWS CLI

获取电话号码详情

以下get-phone-number示例显示了指定电话号码的详细信息。

```
aws chime get-phone-number \  
  --phone-number-id +12065550100
```

输出：

```
{  
  "PhoneNumber": {  
    "PhoneNumberId": "%2B12065550100",  
    "E164PhoneNumber": "+12065550100",  
    "Type": "Local",  
    "ProductType": "VoiceConnector",  
    "Status": "Unassigned",  
    "Capabilities": {  
      "InboundCall": true,  
      "OutboundCall": true,  
      "InboundSMS": true,  
      "OutboundSMS": true,  
      "InboundMMS": true,  
      "OutboundMMS": true  
    },  
    "Associations": [  
      {  
        "Value": "abcdef1ghij2klmno3pqr4",  
        "Name": "VoiceConnectorId",  
        "AssociatedTimestamp": "2019-10-28T18:40:37.453Z"  
      }  
    ],  
    "CallingNameStatus": "UpdateInProgress",  
    "CreatedTimestamp": "2019-08-09T21:35:21.445Z",  
    "UpdatedTimestamp": "2019-08-09T21:35:31.745Z"  
  }  
}
```

```
}
```

有关更多信息，请参阅 Amazon Chime 管理指南中的[使用电话号码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPhoneNumber](#)中的。

## get-proxy-session

以下代码示例演示了如何使用 get-proxy-session。

### AWS CLI

获取代理会话详细信息

以下get-proxy-session示例列出了指定代理会话的详细信息。

```
aws chime get-proxy-session \  
  --voice-connector-id abcdef1ghij2klmno3pqr4 \  
  --proxy-session-id 123a4bc5-67d8-901e-2f3g-h4ghjk567891
```

输出：

```
{  
  "ProxySession": {  
    "VoiceConnectorId": "abcdef1ghij2klmno3pqr4",  
    "ProxySessionId": "123a4bc5-67d8-901e-2f3g-h4ghjk567891",  
    "Status": "Open",  
    "ExpiryMinutes": 60,  
    "Capabilities": [  
      "SMS",  
      "Voice"  
    ],  
    "CreatedTimestamp": "2020-04-15T16:10:10.288Z",  
    "UpdatedTimestamp": "2020-04-15T16:10:10.288Z",  
    "Participants": [  
      {  
        "PhoneNumber": "+12065550100",  
        "ProxyPhoneNumber": "+19135550199"  
      },  
      {  
        "PhoneNumber": "+14015550101",  
        "ProxyPhoneNumber": "+19135550199"  
      }  
    ]  
  }  
}
```

```
    ]  
  }  
}
```

有关更多信息，请参阅 Amazon Chime 开发者指南中的[代理电话会话](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetProxySession](#)中的。

## get-room

以下代码示例演示了如何使用 get-room。

### AWS CLI

要获取有关聊天室的详细信息

以下get-room示例显示有关指定聊天室的详细信息。

```
aws chime get-room \  
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
  --room-id abcd1e2d-3e45-6789-01f2-3g45h67i890j
```

输出：

```
{  
  "Room": {  
    "RoomId": "abcd1e2d-3e45-6789-01f2-3g45h67i890j",  
    "Name": "chatRoom",  
    "AccountId": "12a3456b-7c89-012d-3456-78901e23fg45",  
    "CreatedBy": "arn:aws:iam::111122223333:user/alejandro",  
    "CreatedTimestamp": "2019-12-02T22:29:31.549Z",  
    "UpdatedTimestamp": "2019-12-02T22:29:31.549Z"  
  }  
}
```

有关更多信息，请参阅 Amazon Chime 用户指南中的[创建聊天室](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRoom](#)中的。

## get-user-settings

以下代码示例演示了如何使用 get-user-settings。



## AWS CLI

### 检索用户设置

以下`get-user-settings`示例显示了指定的用户设置。

```
aws chime get-user-settings \  
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
  --user-id 1ab2345c-67de-8901-f23g-45h678901j2k
```

输出：

```
{  
  "UserSettings": {  
    "Telephony": {  
      "InboundCalling": true,  
      "OutboundCalling": true,  
      "SMS": true  
    }  
  }  
}
```

有关更多信息，请参阅 Amazon Chime 管理指南中的管理[用户电话号码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetUserSettings](#)中的。

### get-user

以下代码示例演示了如何使用 `get-user`。

## AWS CLI

获取有关用户的详细信息

以下`get-user`示例检索指定用户的详细信息。

```
aws chime get-user \  
  --account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \  
  --user-id a1b2c3d4-5678-90ab-cdef-22222EXAMPLE
```

输出：

```
{
```

```

    "User": {
      "UserId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "AccountId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "PrimaryEmail": "marthar@example.com",
      "DisplayName": "Martha Rivera",
      "LicenseType": "Pro",
      "UserRegistrationStatus": "Registered",
      "RegisteredOn": "2018-12-20T18:45:25.231Z",
      "InvitedOn": "2018-12-20T18:45:25.231Z",
      "AlexaForBusinessMetadata": {
        "IsAlexaForBusinessEnabled": False,
        "AlexaForBusinessRoomArn": "null"
      },
      "PersonalPIN": "XXXXXXXXXX"
    }
  }
}

```

有关更多信息，请参阅 Amazon Chime 管理指南中的管理[用户](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetUser](#) 中的。

## get-voice-connector-group

以下代码示例演示了如何使用 `get-voice-connector-group`。

### AWS CLI

要获取 Amazon Chime Voice Connector 群组的详细信息

以下 `get-voice-connector-group` 示例显示了指定 Amazon Chime 语音连接器组的详细信息。

```

aws chime get-voice-connector-group \
  --voice-connector-group-id 123a456b-c7d8-90e1-fg23-4h567jkl8901

```

输出：

```

{
  "VoiceConnectorGroup": {
    "VoiceConnectorGroupId": "123a456b-c7d8-90e1-fg23-4h567jkl8901",
    "Name": "myGroup",
    "VoiceConnectorItems": [],
    "CreatedTimestamp": "2019-09-18T16:38:34.734Z",
    "UpdatedTimestamp": "2019-09-18T16:38:34.734Z"
  }
}

```

```
}  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“使用亚马逊 Chime 语音连接器组”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVoiceConnectorGroup](#)中的。

## get-voice-connector-logging-configuration

以下代码示例演示了如何使用 `get-voice-connector-logging-configuration`。

### AWS CLI

获取日志配置详细信息

以下`get-voice-connector-logging-configuration`示例检索了指定 Amazon Chime Voice Connector 的日志配置详细信息。

```
aws chime get-voice-connector-logging-configuration \  
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

输出：

```
{  
  "LoggingConfiguration": {  
    "EnableSIPLogs": true  
  }  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“将 Amazon Chime 语音连接器媒体流式传输到 Kinesis”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVoiceConnectorLoggingConfiguration](#)中的。

## get-voice-connector-origination

以下代码示例演示了如何使用 `get-voice-connector-origination`。

### AWS CLI

检索起源设置

以下`get-voice-connector-origination`示例检索指定 Amazon Chime Voice Connector 的源主机、端口、协议、优先级和权重。

```
aws chime get-voice-connector-origination \  
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

输出：

```
{  
  "Origination": {  
    "Routes": [  
      {  
        "Host": "10.24.34.0",  
        "Port": 1234,  
        "Protocol": "TCP",  
        "Priority": 1,  
        "Weight": 5  
      }  
    ],  
    "Disabled": false  
  }  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[使用 Amazon Chime 语音连接器](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVoiceConnectorOrigination](#)中的。

## get-voice-connector-proxy

以下代码示例演示了如何使用 `get-voice-connector-proxy`。

### AWS CLI

获取代理配置的详细信息

以下`get-voice-connector-proxy`示例获取您的 Amazon Chime Voice Connector 的代理配置详细信息。

```
aws chime get-voice-connector-proxy \  
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

输出：

```
{
  "Proxy": {
    "DefaultSessionExpiryMinutes": 60,
    "Disabled": false,
    "PhoneNumberCountries": [
      "US"
    ]
  }
}
```

有关更多信息，请参阅 Amazon Chime 开发者指南中的[代理电话会话](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetVoiceConnectorProxy](#)中的。

## get-voice-connector-streaming-configuration

以下代码示例演示了如何使用 get-voice-connector-streaming-configuration。

### AWS CLI

获取直播配置详情

以下 get-voice-connector-streaming-configuration 示例获取了指定 Amazon Chime Voice Connector 的直播配置详细信息。

```
aws chime get-voice-connector-streaming-configuration \
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

输出：

```
{
  "StreamingConfiguration": {
    "DataRetentionInHours": 24,
    "Disabled": false
  }
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[将 Amazon Chime 语音连接器数据流式传输到 Kinesis](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetVoiceConnectorStreamingConfiguration](#)中的。

## get-voice-connector-termination-health

以下代码示例演示了如何使用 `get-voice-connector-termination-health`。

### AWS CLI

检索终止健康状况详细信息

以下 `get-voice-connector-termination-health` 示例检索指定 Amazon Chime 语音连接器的终止运行状况详情。

```
aws chime get-voice-connector-termination-health \  
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

输出：

```
{  
  "TerminationHealth": {  
    "Timestamp": "Fri Aug 23 16:45:55 UTC 2019",  
    "Source": "10.24.34.0"  
  }  
}
```

有关更多信息，请参阅 [《亚马逊 Chime 管理指南》](#) 中的“使用 Amazon Chime 语音连接器”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetVoiceConnectorTerminationHealth](#) 中的。

## get-voice-connector-termination

以下代码示例演示了如何使用 `get-voice-connector-termination`。

### AWS CLI

检索终止设置

以下 `get-voice-connector-termination` 示例检索指定 Amazon Chime 语音连接器的终止设置。

```
aws chime get-voice-connector-termination \  
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

此命令不生成任何输出。输出：

```
{
  "Termination": {
    "CpsLimit": 1,
    "DefaultPhoneNumber": "+12065550100",
    "CallingRegions": [
      "US"
    ],
    "CidrAllowedList": [
      "10.24.34.0/23"
    ],
    "Disabled": false
  }
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“使用 Amazon Chime 语音连接器”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetVoiceConnectorTermination](#) 中的。

## get-voice-connector

以下代码示例演示了如何使用 get-voice-connector。

### AWS CLI

要获取 Amazon Chime 语音连接器的详细信息

以下 get-voice-connector 示例显示了指定 Amazon Chime 语音连接器的详细信息。

```
aws chime get-voice-connector \
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

输出：

```
{
  "VoiceConnector": {
    "VoiceConnectorId": "abcdef1ghij2klmno3pqr4",
    "AwsRegion": "us-west-2",
    "Name": "newVoiceConnector",
    "OutboundHostName": "abcdef1ghij2klmno3pqr4.voiceconnector.chime.aws",
    "RequireEncryption": true,
    "CreatedTimestamp": "2019-09-18T20:34:01.352Z",
    "UpdatedTimestamp": "2019-09-18T20:34:01.352Z"
  }
}
```

```
}  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[使用 Amazon Chime 语音连接器](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetVoiceConnector](#) 中的。

## invite-users

以下代码示例演示了如何使用 `invite-users`。

### AWS CLI

邀请用户加入 Amazon Chime

以下 `invite-users` 示例发送一封电子邮件邀请用户使用指定的 Amazon Chime 账户。

```
aws chime invite-users \  
  --account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \  
  --user-email-list "alejandr@example.com" "janed@example.com"
```

输出：

```
{  
  "Invites": [  
    {  
      "InviteId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",  
      "Status": "Pending",  
      "EmailAddress": "alejandr@example.com",  
      "EmailStatus": "Sent"  
    },  
    {  
      "InviteId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",  
      "Status": "Pending",  
      "EmailAddress": "janed@example.com",  
      "EmailStatus": "Sent"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon Chime [管理指南中的邀请和暂停用户](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[InviteUsers](#)中的。

## list-accounts

以下代码示例演示了如何使用 list-accounts。

### AWS CLI

#### 获取账户列表

以下list-accounts示例检索管理员 AWS 账户中的 Amazon Chime 账户列表。

```
aws chime list-accounts
```

输出：

```
{
  "Accounts": [
    {
      "AwsAccountId": "111122223333",
      "AccountId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "Name": "First Chime Account",
      "AccountType": "EnterpriseDirectory",
      "CreatedTimestamp": "2018-12-20T18:38:02.181Z",
      "DefaultLicense": "Pro",
      "SupportedLicenses": [
        "Basic",
        "Pro"
      ],
      "SigninDelegateGroups": [
        {
          "GroupName": "myGroup"
        }
      ]
    },
    {
      "AwsAccountId": "111122223333",
      "AccountId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "Name": "Second Chime Account",
      "AccountType": "Team",
      "CreatedTimestamp": "2018-09-04T21:44:22.292Z",
      "DefaultLicense": "Pro",

```

```

        "SupportedLicenses": [
            "Basic",
            "Pro"
        ],
        "SigninDelegateGroups": [
            {
                "GroupName": "myGroup"
            },
        ]
    }
]
}

```

有关更多信息，请参阅 [《亚马逊 Chime 管理指南》](#) 中的“[管理您的亚马逊 Chime 账户](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAccounts](#) 中的。

## list-bots

以下代码示例演示了如何使用 list-bots。

### AWS CLI

#### 检索机器人列表

以下 list-bots 示例列出了与指定的 Amazon Chime Enterprise 账户关联的机器人。

```
aws chime list-bots \
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45
```

输出：

```

{
  "Bot": {
    "BotId": "123abcd4-5ef6-789g-0h12-34j56789012k",
    "UserId": "123abcd4-5ef6-789g-0h12-34j56789012k",
    "DisplayName": "myBot (Bot)",
    "BotType": "ChatBot",
    "Disabled": false,
    "CreatedTimestamp": "2019-09-09T18:05:56.749Z",
    "UpdatedTimestamp": "2019-09-09T18:05:56.749Z",
    "BotEmail": "myBot-chimebot@example.com",
    "SecurityToken": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
  }
}

```

```
}  
}
```

有关更多信息，请参阅《亚马逊 Chime 开发者指南》中的[将聊天机器人与 Amazon Chime 配合使用](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBots](#)中的。

## list-phone-number-orders

以下代码示例演示了如何使用 list-phone-number-orders。

### AWS CLI

列出电话号码订单

以下list-phone-number-orders示例列出了与 Amazon Chime 管理员账户关联的电话号码订单。

```
aws chime list-phone-number-orders
```

输出：

```
{  
  "PhoneNumberOrders": [  
    {  
      "PhoneNumberOrderId": "abc12345-de67-89f0-123g-h45i678j9012",  
      "ProductType": "VoiceConnector",  
      "Status": "Partial",  
      "OrderedPhoneNumbers": [  
        {  
          "E164PhoneNumber": "+12065550100",  
          "Status": "Acquired"  
        },  
        {  
          "E164PhoneNumber": "+12065550101",  
          "Status": "Acquired"  
        },  
        {  
          "E164PhoneNumber": "+12065550102",  
          "Status": "Failed"  
        }  
      ]  
    }  
  ]  
}
```

```
    ],
    "CreatedTimestamp": "2019-08-09T21:35:21.427Z",
    "UpdatedTimestamp": "2019-08-09T21:35:31.926Z"
  }
  {
    "PhoneNumberOrderId": "cba54321-ed76-09f5-321g-h54i876j2109",
    "ProductType": "BusinessCalling",
    "Status": "Partial",
    "OrderedPhoneNumbers": [
      {
        "E164PhoneNumber": "+12065550103",
        "Status": "Acquired"
      },
      {
        "E164PhoneNumber": "+12065550104",
        "Status": "Acquired"
      },
      {
        "E164PhoneNumber": "+12065550105",
        "Status": "Failed"
      }
    ],
    "CreatedTimestamp": "2019-08-09T21:35:21.427Z",
    "UpdatedTimestamp": "2019-08-09T21:35:31.926Z"
  }
]
}
```

有关更多信息，请参阅 Amazon Chime 管理指南中的[使用电话号码](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListPhoneNumberOrders](#)中的。

## list-phone-numbers

以下代码示例演示了如何使用 list-phone-numbers。

### AWS CLI

列出 Amazon Chime 账户的电话号码

以下 list-phone-numbers 示例列出了与管理员的 Amazon Chime 账户关联的电话号码。

```
aws chime list-phone-numbers
```

此命令不生成任何输出。输出：

```
{
  "PhoneNumbers": [
    {
      "PhoneNumberId": "%2B12065550100",
      "E164PhoneNumber": "+12065550100",
      "Type": "Local",
      "ProductType": "VoiceConnector",
      "Status": "Assigned",
      "Capabilities": {
        "InboundCall": true,
        "OutboundCall": true,
        "InboundSMS": true,
        "OutboundSMS": true,
        "InboundMMS": true,
        "OutboundMMS": true
      },
      "Associations": [
        {
          "Value": "abcdef1ghij2klmno3pqr4",
          "Name": "VoiceConnectorId",
          "AssociatedTimestamp": "2019-10-28T18:40:37.453Z"
        }
      ],
      "CallingNameStatus": "UpdateInProgress",
      "CreatedTimestamp": "2019-08-12T22:10:20.521Z",
      "UpdatedTimestamp": "2019-10-28T18:42:07.964Z"
    },
    {
      "PhoneNumberId": "%2B12065550101",
      "E164PhoneNumber": "+12065550101",
      "Type": "Local",
      "ProductType": "VoiceConnector",
      "Status": "Assigned",
      "Capabilities": {
        "InboundCall": true,
        "OutboundCall": true,
        "InboundSMS": true,
        "OutboundSMS": true,
        "InboundMMS": true,
        "OutboundMMS": true
      },
      "Associations": [
```

```

        {
            "Value": "abcdef1ghij2klmno3pqr4",
            "Name": "VoiceConnectorId",
            "AssociatedTimestamp": "2019-10-28T18:40:37.511Z"
        }
    ],
    "CallingNameStatus": "UpdateInProgress",
    "CreatedTimestamp": "2019-08-12T22:10:20.521Z",
    "UpdatedTimestamp": "2019-10-28T18:42:07.960Z"
}
]
}

```

有关更多信息，请参阅 Amazon Chime 管理指南中的[使用电话号码](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListPhoneNumbers](#)中的。

## list-proxy-sessions

以下代码示例演示了如何使用 list-proxy-sessions。

### AWS CLI

列出代理会话

以下 list-proxy-sessions 示例列出了您的 Amazon Chime 语音连接器的代理会话。

```
aws chime list-proxy-sessions \
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

输出：

```

{
  "ProxySession": {
    "VoiceConnectorId": "abcdef1ghij2klmno3pqr4",
    "ProxySessionId": "123a4bc5-67d8-901e-2f3g-h4ghjk567891",
    "Status": "Open",
    "ExpiryMinutes": 60,
    "Capabilities": [
      "SMS",
      "Voice"
    ],
    "CreatedTimestamp": "2020-04-15T16:10:10.288Z",
  }
}

```

```
    "UpdatedTimestamp": "2020-04-15T16:10:10.288Z",
    "Participants": [
      {
        "PhoneNumber": "+12065550100",
        "ProxyPhoneNumber": "+19135550199"
      },
      {
        "PhoneNumber": "+14015550101",
        "ProxyPhoneNumber": "+19135550199"
      }
    ]
  }
}
```

有关更多信息，请参阅 Amazon Chime 开发者指南中的[代理电话会话](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListProxySessions](#)中的。

## list-room-memberships

以下代码示例演示了如何使用 `list-room-memberships`。

### AWS CLI

列出聊天室成员资格

以下 `list-room-memberships` 示例显示了指定聊天室的成员资格详细信息列表。

```
aws chime list-room-memberships \
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \
  --room-id abcd1e2d-3e45-6789-01f2-3g45h67i890j
```

输出：

```
{
  "RoomMemberships": [
    {
      "RoomId": "abcd1e2d-3e45-6789-01f2-3g45h67i890j",
      "Member": {
        "MemberId": "2ab2345c-67de-8901-f23g-45h678901j2k",
        "MemberType": "User",
        "Email": "zhangw@example.com",
        "FullName": "Zhang Wei",
```

```

        "AccountId": "12a3456b-7c89-012d-3456-78901e23fg45"
    },
    "Role": "Member",
    "InvitedBy": "arn:aws:iam::111122223333:user/alejandro",
    "UpdatedTimestamp": "2019-12-02T22:46:58.532Z"
},
{
    "RoomId": "abcd1e2d-3e45-6789-01f2-3g45h67i890j",
    "Member": {
        "MemberId": "1ab2345c-67de-8901-f23g-45h678901j2k",
        "MemberType": "User",
        "Email": "janed@example.com",
        "FullName": "Jane Doe",
        "AccountId": "12a3456b-7c89-012d-3456-78901e23fg45"
    },
    "Role": "Administrator",
    "InvitedBy": "arn:aws:iam::111122223333:user/alejandro",
    "UpdatedTimestamp": "2019-12-02T22:46:58.532Z"
}
]
}

```

有关更多信息，请参阅 Amazon Chime 用户指南中的[创建聊天室](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRoomMemberships](#)中的。

## list-rooms

以下代码示例演示了如何使用 list-rooms。

### AWS CLI

#### 列出聊天室

以下list-rooms示例显示了指定账户中的聊天室列表。列表仅筛选到指定成员所属的聊天室。

```

aws chime list-rooms \
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \
  --member-id 1ab2345c-67de-8901-f23g-45h678901j2k

```

输出：

```
{
```



```
"Room": {
  "RoomId": "abcd1e2d-3e45-6789-01f2-3g45h67i890j",
  "Name": "teamRoom",
  "AccountId": "12a3456b-7c89-012d-3456-78901e23fg45",
  "CreatedBy": "arn:aws:iam::111122223333:user/alejandro",
  "CreatedTimestamp": "2019-12-02T22:29:31.549Z",
  "UpdatedTimestamp": "2019-12-02T22:33:19.310Z"
}
```

有关更多信息，请参阅 Amazon Chime 用户指南中的[创建聊天室](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListRooms](#)中的。

## list-users

以下代码示例演示了如何使用 `list-users`。

### AWS CLI

列出账户中的用户

以下 `list-users` 示例列出了指定 Amazon Chime 账户的用户。

```
aws chime list-users --account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
```

输出：

```
{
  "Users": [
    {
      "UserId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "AccountId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "PrimaryEmail": "mariag@example.com",
      "DisplayName": "Maria Garcia",
      "LicenseType": "Pro",
      "UserType": "PrivateUser",
      "UserRegistrationStatus": "Registered",
      "RegisteredOn": "2018-12-20T18:45:25.231Z"
      "AlexaForBusinessMetadata": {
        "IsAlexaForBusinessEnabled": false
      }
    }
  ],
}
```

```
{
  "UserId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
  "AccountId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "PrimaryEmail": "richardr@example.com",
  "DisplayName": "Richard Roe",
  "LicenseType": "Pro",
  "UserType": "PrivateUser",
  "UserRegistrationStatus": "Registered",
  "RegisteredOn": "2018-12-20T18:45:45.415Z"
  "AlexaForBusinessMetadata": {
    "IsAlexaForBusinessEnabled": false
  }
},
{
  "UserId": "a1b2c3d4-5678-90ab-cdef-44444EXAMPLE",
  "AccountId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "PrimaryEmail": "saanvis@example.com",
  "DisplayName": "Saanvi Sarkar",
  "LicenseType": "Basic",
  "UserType": "PrivateUser",
  "UserRegistrationStatus": "Registered",
  "RegisteredOn": "2018-12-20T18:46:57.747Z"
  "AlexaForBusinessMetadata": {
    "IsAlexaForBusinessEnabled": false
  }
},
{
  "UserId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE",
  "AccountId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "PrimaryEmail": "wxiulan@example.com",
  "DisplayName": "Wang Xiulan",
  "LicenseType": "Basic",
  "UserType": "PrivateUser",
  "UserRegistrationStatus": "Registered",
  "RegisteredOn": "2018-12-20T18:47:15.390Z"
  "AlexaForBusinessMetadata": {
    "IsAlexaForBusinessEnabled": false
  }
}
]
```

有关更多信息，请参阅 Amazon Chime 管理指南中的管理[用户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUsers](#)中的。

## list-voice-connector-groups

以下代码示例演示了如何使用 `list-voice-connector-groups`。

### AWS CLI

列出亚马逊 Chime 账户的亚马逊 Chime 语音连接器群组

以下`list-voice-connector-groups`示例列出了与管理员的 Amazon Chime 账户关联的 Amazon Chime Voice Connector 群组。

```
aws chime list-voice-connector-groups
```

输出：

```
{
  "VoiceConnectorGroups": [
    {
      "VoiceConnectorGroupId": "123a456b-c7d8-90e1-fg23-4h567jkl8901",
      "Name": "myGroup",
      "VoiceConnectorItems": [],
      "CreatedTimestamp": "2019-09-18T16:38:34.734Z",
      "UpdatedTimestamp": "2019-09-18T16:38:34.734Z"
    }
  ]
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“使用 Amazon Chime 语音连接器群组”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListVoiceConnectorGroups](#)中的。

## list-voice-connector-termination-credentials

以下代码示例演示了如何使用 `list-voice-connector-termination-credentials`。

### AWS CLI

检索终止凭证列表

以下`list-voice-connector-termination-credentials`示例检索指定 Amazon Chime Voice Connector 的终止凭证列表。

```
aws chime list-voice-connector-termination-credentials \  
  --voice-connector-id abcdef1ghij2klmno3pqr4
```

此命令不生成任何输出。输出：

```
{  
  "Usernames": [  
    "jdoe"  
  ]  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[使用 Amazon Chime 语音连接器](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListVoiceConnectorTerminationCredentials](#)中的。

## list-voice-connectors

以下代码示例演示了如何使用 `list-voice-connectors`。

### AWS CLI

列出账户的 Amazon Chime 语音连接器

以下`list-voice-connectors`示例列出了与来电者账户关联的 Amazon Chime 语音连接器。

```
aws chime list-voice-connectors
```

输出：

```
{  
  "VoiceConnectors": [  
    {  
      "VoiceConnectorId": "abcdef1ghij2klmno3pqr4",  
      "AwsRegion": "us-east-1",  
      "Name": "MyVoiceConnector",  
      "OutboundHostName": "abcdef1ghij2klmno3pqr4.voiceconnector.chime.aws",  
      "RequireEncryption": true,  
      "CreatedTimestamp": "2019-06-04T18:46:56.508Z",  
    }  
  ]  
}
```

```
    "UpdatedTimestamp": "2019-09-18T16:33:00.806Z"
  },
  {
    "VoiceConnectorId": "cbadef1ghij2klmno3pqr5",
    "AwsRegion": "us-west-2",
    "Name": "newVoiceConnector",
    "OutboundHostName": "cbadef1ghij2klmno3pqr5.voiceconnector.chime.aws",
    "RequireEncryption": true,
    "CreatedTimestamp": "2019-09-18T20:34:01.352Z",
    "UpdatedTimestamp": "2019-09-18T20:34:01.352Z"
  }
]
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“使用 Amazon Chime 语音连接器”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListVoiceConnectors](#) 中的。

## logout-user

以下代码示例演示了如何使用 `logout-user`。

### AWS CLI

注销用户

以下 `logout-user` 示例注销了指定的用户。

```
aws chime logout-user \
  --account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \
  --user-id a1b2c3d4-5678-90ab-cdef-22222EXAMPLE
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [LogoutUser](#) 中的。

## put-voice-connector-logging-configuration

以下代码示例演示了如何使用 `put-voice-connector-logging-configuration`。

### AWS CLI

为 Amazon Chime 语音连接器添加日志配置

以下 `put-voice-connector-logging-configuration` 示例为指定的 Amazon Chime 语音连接器开启了 SIP 日志配置。

```
aws chime put-voice-connector-logging-configuration \  
  --voice-connector-id abcdef1ghij2klmno3pqr4 \  
  --logging-configuration EnableSIPLogs=true
```

输出：

```
{  
  "LoggingConfiguration": {  
    "EnableSIPLogs": true  
  }  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[将 Amazon Chime 语音连接器媒体流式传输到 Kinesis](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutVoiceConnectorLoggingConfiguration](#) 中的。

## put-voice-connector-origination

以下代码示例演示了如何使用 `put-voice-connector-origination`。

AWS CLI

要设置起源设置

以下 `put-voice-connector-origination` 示例为指定的 Amazon Chime Voice Connector 设置源主机、端口、协议、优先级和权重。

```
aws chime put-voice-connector-origination \  
  --voice-connector-id abcdef1ghij2klmno3pqr4 \  
  --origination  
  Routes=[{Host="10.24.34.0",Port=1234,Protocol="TCP",Priority=1,Weight=5}],Disabled=false
```

输出：

```
{
```

```
"Origination": {
  "Routes": [
    {
      "Host": "10.24.34.0",
      "Port": 1234,
      "Protocol": "TCP",
      "Priority": 1,
      "Weight": 5
    }
  ],
  "Disabled": false
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“使用 Amazon Chime 语音连接器”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutVoiceConnectorOrigination](#) 中的。

## put-voice-connector-proxy

以下代码示例演示了如何使用 put-voice-connector-proxy。

### AWS CLI

#### 放置代理配置

以下 put-voice-connector-proxy 示例为您的 Amazon Chime 语音连接器设置代理配置。

```
aws chime put-voice-connector-proxy \
  --voice-connector-id abcdef1ghij2klmno3pqr4 \
  --default-session-expiry-minutes 60 \
  --phone-number-pool-countries "US"
```

输出：

```
{
  "Proxy": {
    "DefaultSessionExpiryMinutes": 60,
    "Disabled": false,
    "PhoneNumberCountries": [
      "US"
    ]
  }
}
```

```
}  
}
```

有关更多信息，请参阅 Amazon Chime 开发者指南中的[代理电话会话](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutVoiceConnectorProxy](#) 中的。

## put-voice-connector-streaming-configuration

以下代码示例演示了如何使用 put-voice-connector-streaming-configuration。

### AWS CLI

#### 创建直播配置

以下 put-voice-connector-streaming-configuration 示例为指定的 Amazon Chime 语音连接器创建直播配置。它支持将媒体从 Amazon Chime Voice Connector 流式传输到 Amazon Kinesis，并将数据保留期设置为 24 小时。

```
aws chime put-voice-connector-streaming-configuration \  
  --voice-connector-id abcdef1ghij2klmno3pqr4 \  
  --streaming-configuration DataRetentionInHours=24,Disabled=false
```

输出：

```
{  
  "StreamingConfiguration": {  
    "DataRetentionInHours": 24,  
    "Disabled": false  
  }  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[将 Amazon Chime 语音连接器数据流式传输到 Kinesis](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutVoiceConnectorStreamingConfiguration](#) 中的。

## put-voice-connector-termination-credentials

以下代码示例演示了如何使用 put-voice-connector-termination-credentials。



## AWS CLI

### 设置终止凭证

以下 `put-voice-connector-termination-credentials` 示例为指定的 Amazon Chime 语音连接器设置终止凭证。

```
aws chime put-voice-connector-termination-credentials \  
  --voice-connector-id abcdef1ghij2klmno3pqr4 \  
  --credentials Username="jdoe",Password="XXXXXXXX"
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“[使用 Amazon Chime 语音连接器](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutVoiceConnectorTerminationCredentials](#) 中的。

## `put-voice-connector-termination`

以下代码示例演示了如何使用 `put-voice-connector-termination`。

## AWS CLI

### 要设置终止设置

以下 `put-voice-connector-termination` 示例为指定的 Amazon Chime 语音连接器设置呼叫区域和允许的 IP 主机终止设置。

```
aws chime put-voice-connector-termination \  
  --voice-connector-id abcdef1ghij2klmno3pqr4 \  
  --termination CallingRegions="US",CidrAllowedList="10.24.34.0/23",Disabled=false
```

输出：

```
{  
  "Termination": {  
    "CpsLimit": 0,  
    "CallingRegions": [  
      "US"  
    ],  
    "CidrAllowedList": [  

```

```
        "10.24.34.0/23"  
    ],  
    "Disabled": false  
  }  
}
```

有关更多信息，请参阅《[亚马逊 Chime 管理指南](#)》中的“使用 Amazon Chime 语音连接器”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutVoiceConnectorTermination](#)中的。

## regenerate-security-token

以下代码示例演示了如何使用 regenerate-security-token。

### AWS CLI

#### 重新生成安全令牌

以下regenerate-security-token示例为指定的机器人重新生成安全令牌。

```
aws chime regenerate-security-token \  
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
  --bot-id 123abcd4-5ef6-789g-0h12-34j56789012k
```

输出：

```
{  
  "Bot": {  
    "BotId": "123abcd4-5ef6-789g-0h12-34j56789012k",  
    "UserId": "123abcd4-5ef6-789g-0h12-34j56789012k",  
    "DisplayName": "myBot (Bot)",  
    "BotType": "ChatBot",  
    "Disabled": false,  
    "CreatedTimestamp": "2019-09-09T18:05:56.749Z",  
    "UpdatedTimestamp": "2019-09-09T18:05:56.749Z",  
    "BotEmail": "myBot-chimebot@example.com",  
    "SecurityToken": "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY"  
  }  
}
```

有关更多信息，请参阅 Amazon Chime 开发者指南中的[验证聊天机器人请求](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegenerateSecurityToken](#)中的。

## reset-personal-pin

以下代码示例演示了如何使用 `reset-personal-pin`。

### AWS CLI

#### 重置用户的个人会议 PIN

以下 `reset-personal-pin` 示例重置了指定用户的个人会议 PIN。

```
aws chime reset-personal-pin \  
  --account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \  
  --user-id a1b2c3d4-5678-90ab-cdef-22222EXAMPLE
```

输出：

```
{  
  "User": {  
    "UserId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",  
    "AccountId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
    "PrimaryEmail": "mateo@example.com",  
    "DisplayName": "Mateo Jackson",  
    "LicenseType": "Pro",  
    "UserType": "PrivateUser",  
    "UserRegistrationStatus": "Registered",  
    "RegisteredOn": "2018-12-20T18:45:25.231Z",  
    "AlexaForBusinessMetadata": {  
      "IsAlexaForBusinessEnabled": False,  
      "AlexaForBusinessRoomArn": "null"  
    },  
    "PersonalPIN": "XXXXXXXXXX"  
  }  
}
```

有关更多信息，请参阅 Amazon Chime 管理指南中的[更改个人会议 PIN](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ResetPersonalPin](#)中的。

## restore-phone-number

以下代码示例演示了如何使用 `restore-phone-number`。

## AWS CLI

### 恢复电话号码

以下 `restore-phone-number` 示例从删除队列中恢复指定的电话号码。

```
aws chime restore-phone-number \  
  --phone-number-id "+12065550100"
```

输出：

```
{  
  "PhoneNumber": {  
    "PhoneNumberId": "%2B12065550100",  
    "E164PhoneNumber": "+12065550100",  
    "Type": "Local",  
    "ProductType": "BusinessCalling",  
    "Status": "Unassigned",  
    "Capabilities": {  
      "InboundCall": true,  
      "OutboundCall": true,  
      "InboundSMS": true,  
      "OutboundSMS": true,  
      "InboundMMS": true,  
      "OutboundMMS": true  
    },  
    "Associations": [],  
    "CreatedTimestamp": "2019-08-09T21:35:21.445Z",  
    "UpdatedTimestamp": "2019-08-12T22:06:36.355Z"  
  }  
}
```

有关更多信息，请参阅 Amazon Chime 管理指南中的[使用电话号码](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RestorePhoneNumber](#)中的。

## search-available-phone-numbers

以下代码示例演示了如何使用 `search-available-phone-numbers`。

## AWS CLI

### 搜索可用的电话号码

以下search-available-phone-numbers示例按区号搜索可用的电话号码。

```
aws chime search-available-phone-numbers \  
  --area-code "206"
```

输出：

```
{  
  "E164PhoneNumbers": [  
    "+12065550100",  
    "+12065550101",  
    "+12065550102",  
    "+12065550103",  
    "+12065550104",  
    "+12065550105",  
    "+12065550106",  
    "+12065550107",  
    "+12065550108",  
    "+12065550109",  
  ]  
}
```

有关更多信息，请参阅 Amazon Chime 管理指南中的[使用电话号码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SearchAvailablePhoneNumbers](#)中的。

## update-account-settings

以下代码示例演示了如何使用 update-account-settings。

### AWS CLI

#### 更新您的账户设置

以下update-account-settings示例禁用了指定 Amazon Chime 账户对共享屏幕的远程控制。

```
aws chime update-account-settings \  
  --account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \  
  --account-settings DisableRemoteControl=true
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAccountSettings](#)中的。

## update-account

以下代码示例演示了如何使用 update-account。

### AWS CLI

#### 更新账户

以下update-account示例更新了指定的账户名。

```
aws chime update-account \
  --account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \
  --name MyAccountName
```

输出：

```
{
  "Account": {
    "AwsAccountId": "111122223333",
    "AccountId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "Name": "MyAccountName",
    "AccountType": "Team",
    "CreatedTimestamp": "2018-09-04T21:44:22.292Z",
    "DefaultLicense": "Pro",
    "SupportedLicenses": [
      "Basic",
      "Pro"
    ],
    "SigninDelegateGroups": [
      {
        "GroupName": "myGroup"
      }
    ]
  }
}
```

有关更多信息，请参阅 Amazon Chime [管理指南中的重命名您的账户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAccount](#)中的。

## update-bot

以下代码示例演示了如何使用 update-bot。

## AWS CLI

### 更新机器人

以下update-bot示例更新了指定机器人的状态以阻止其运行。

```
aws chime update-bot \  
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
  --bot-id 123abcd4-5ef6-789g-0h12-34j56789012k \  
  --disabled
```

输出：

```
{  
  "Bot": {  
    "BotId": "123abcd4-5ef6-789g-0h12-34j56789012k",  
    "UserId": "123abcd4-5ef6-789g-0h12-34j56789012k",  
    "DisplayName": "myBot (Bot)",  
    "BotType": "ChatBot",  
    "Disabled": true,  
    "CreatedTimestamp": "2019-09-09T18:05:56.749Z",  
    "UpdatedTimestamp": "2019-09-09T18:05:56.749Z",  
    "BotEmail": "myBot-chimebot@example.com",  
    "SecurityToken": "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY"  
  }  
}
```

有关更多信息，请参阅 Amazon Chime 开发者指南中的[更新聊天机器人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateBot](#)中的。

## update-global-settings

以下代码示例演示了如何使用 update-global-settings。

## AWS CLI

### 更新全局设置

以下update-global-settings示例更新了用于存储与管理员账户关联的 Amazon Chime Business Calling 和 Amazon Chime 语音连接器的通话详细记录的 S3 存储桶。AWS

```
aws chime update-global-settings \  
  --business-calling CdrBucket="s3bucket" \  
  --voice-connector CdrBucket="s3bucket"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Chime 管理指南中的管理全局[设置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateGlobalSettings](#)中的。

## update-phone-number-settings

以下代码示例演示了如何使用 update-phone-number-settings。

### AWS CLI

更新出站呼叫姓名

以下update-phone-number-settings示例更新了管理员 AWS 账户的默认出站呼叫名称。

```
aws chime update-phone-number-settings \  
  --calling-name "myName"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Chime 管理指南中的[使用电话号码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePhoneNumberSettings](#)中的。

## update-phone-number

以下代码示例演示了如何使用 update-phone-number。

### AWS CLI

示例 1：更新电话号码的产品类型

以下update-phone-number示例更新了指定电话号码的产品类型。

```
aws chime update-phone-number \  
  --phone-number-id "+12065550100" \  
  --product-type "BusinessCalling"
```



```
--product-type "BusinessCalling"
```

输出：

```
{
  "PhoneNumber": {
    "PhoneNumberId": "%2B12065550100",
    "E164PhoneNumber": "+12065550100",
    "Type": "Local",
    "ProductType": "BusinessCalling",
    "Status": "Unassigned",
    "Capabilities": {
      "InboundCall": true,
      "OutboundCall": true,
      "InboundSMS": true,
      "OutboundSMS": true,
      "InboundMMS": true,
      "OutboundMMS": true
    },
    "Associations": [],
    "CallingName": "phonenumber1",
    "CreatedTimestamp": "2019-08-09T21:35:21.445Z",
    "UpdatedTimestamp": "2019-08-12T21:44:07.591Z"
  }
}
```

示例 2：更新电话号码的出站呼叫名称

以下update-phone-number示例更新指定电话号码的出站呼叫名称。

```
aws chime — phone-number-id "+12065550100" — calling update-phone-number -name
"phonenumber2"
```

输出：

```
{
  "PhoneNumber": {
    "PhoneNumberId": "%2B12065550100",
    "E164PhoneNumber": "+12065550100",
    "Type": "Local",
    "ProductType": "BusinessCalling",
    "Status": "Unassigned",
```

```
    "Capabilities": {
      "InboundCall": true,
      "OutboundCall": true,
      "InboundSMS": true,
      "OutboundSMS": true,
      "InboundMMS": true,
      "OutboundMMS": true
    },
    "Associations": [],
    "CallingName": "phonenumber2",
    "CreatedTimestamp": "2019-08-09T21:35:21.445Z",
    "UpdatedTimestamp": "2019-08-12T21:44:07.591Z"
  }
}
```

有关更多信息，请参阅 Amazon Chime 管理指南中的[使用电话号码](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdatePhoneNumber](#)中的。

## update-proxy-session

以下代码示例演示了如何使用 update-proxy-session。

### AWS CLI

#### 更新代理会话

以下 update-proxy-session 示例更新了代理会话功能。

```
aws chime update-proxy-session \
  --voice-connector-id abcdef1ghij2klmno3pqr4 \
  --proxy-session-id 123a4bc5-67d8-901e-2f3g-h4ghjk567891 \
  --capabilities "Voice"
```

输出：

```
{
  "ProxySession": {
    "VoiceConnectorId": "abcdef1ghij2klmno3pqr4",
    "ProxySessionId": "123a4bc5-67d8-901e-2f3g-h4ghjk567891",
    "Status": "Open",
    "ExpiryMinutes": 60,
  }
}
```

```

    "Capabilities": [
      "Voice"
    ],
    "CreatedTimestamp": "2020-04-15T16:10:10.288Z",
    "UpdatedTimestamp": "2020-04-15T16:10:10.288Z",
    "Participants": [
      {
        "PhoneNumber": "+12065550100",
        "ProxyPhoneNumber": "+19135550199"
      },
      {
        "PhoneNumber": "+14015550101",
        "ProxyPhoneNumber": "+19135550199"
      }
    ]
  }
}

```

有关更多信息，请参阅 Amazon Chime 开发者指南中的[代理电话会话](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateProxySession](#)中的。

## update-room-membership

以下代码示例演示了如何使用 update-room-membership。

### AWS CLI

#### 更新聊天室成员资格

以下 update-room-membership 示例将指定聊天室成员的角色修改为 Administrator

```

aws chime update-room-membership \
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \
  --room-id abcd1e2d-3e45-6789-01f2-3g45h67i890j \
  --member-id 1ab2345c-67de-8901-f23g-45h678901j2k \
  --role Administrator

```

#### 输出：

```

{
  "RoomMembership": {

```

```
"RoomId": "abcd1e2d-3e45-6789-01f2-3g45h67i890j",
"Member": {
  "MemberId": "1ab2345c-67de-8901-f23g-45h678901j2k",
  "MemberType": "User",
  "Email": "sofiamartinez@example.com",
  "FullName": "Sofia Martinez",
  "AccountId": "12a3456b-7c89-012d-3456-78901e23fg45"
},
"Role": "Administrator",
"InvitedBy": "arn:aws:iam::111122223333:user/admin",
"UpdatedTimestamp": "2019-12-02T22:40:22.931Z"
}
}
```

有关更多信息，请参阅 Amazon Chime 用户指南中的[创建聊天室](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateRoomMembership](#)中的。

## update-room

以下代码示例演示了如何使用 update-room。

### AWS CLI

#### 更新聊天室

以下 update-room 示例修改了指定聊天室的名称。

```
aws chime update-room \
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \
  --room-id abcd1e2d-3e45-6789-01f2-3g45h67i890j \
  --name teamRoom
```

输出：

```
{
  "Room": {
    "RoomId": "abcd1e2d-3e45-6789-01f2-3g45h67i890j",
    "Name": "teamRoom",
    "AccountId": "12a3456b-7c89-012d-3456-78901e23fg45",
    "CreatedBy": "arn:aws:iam::111122223333:user/alejandro",
    "CreatedTimestamp": "2019-12-02T22:29:31.549Z",
```

```
    "UpdatedTimestamp": "2019-12-02T22:33:19.310Z"  
  }  
}
```

有关更多信息，请参阅 Amazon Chime 用户指南中的[创建聊天室](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRoom](#)中的。

## update-user-settings

以下代码示例演示了如何使用 update-user-settings。

### AWS CLI

#### 更新用户设置

以下update-user-settings示例使指定用户能够拨打入站和出站呼叫以及发送和接收 SMS 消息。

```
aws chime update-user-settings \  
  --account-id 12a3456b-7c89-012d-3456-78901e23fg45 \  
  --user-id 1ab2345c-67de-8901-f23g-45h678901j2k \  
  --user-settings "Telephony={InboundCalling=true,OutboundCalling=true,SMS=true}"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Chime 管理指南中的管理[用户电话号码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUserSettings](#)中的。

## update-user

以下代码示例演示了如何使用 update-user。

### AWS CLI

#### 更新用户详细信息

此示例更新了指定用户的指定详细信息。

命令:

```
aws chime update-user \  
  --account-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \  
  --user-id a1b2c3d4-5678-90ab-cdef-22222EXAMPLE \  
  --license-type "Basic"
```

输出：

```
{  
  "User": {  
    "UserId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUser](#)中的。

## update-voice-connector-group

以下代码示例演示了如何使用 update-voice-connector-group。

### AWS CLI

更新 Amazon Chime Voice Connector 群组的详细信息

以下update-voice-connector-group示例更新了指定 Amazon Chime 语音连接器组的详细信息。

```
aws chime update-voice-connector-group \  
  --voice-connector-group-id 123a456b-c7d8-90e1-fg23-4h567jk18901 \  
  --name "newGroupName" \  
  --voice-connector-items VoiceConnectorId=abcdef1ghij2klmno3pqr4,Priority=1
```

输出：

```
{  
  "VoiceConnectorGroup": {  
    "VoiceConnectorGroupId": "123a456b-c7d8-90e1-fg23-4h567jk18901",  
    "Name": "newGroupName",  
    "VoiceConnectorItems": [  
      {  
        "VoiceConnectorId": "abcdef1ghij2klmno3pqr4",
```

```
        "Priority": 1
      }
    ],
    "CreatedTimestamp": "2019-09-18T16:38:34.734Z",
    "UpdatedTimestamp": "2019-10-28T19:00:57.081Z"
  }
}
```

有关更多信息，请参阅 [《亚马逊 Chime 管理指南》](#) 中的“使用亚马逊 Chime 语音连接器组”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateVoiceConnectorGroup](#) 中的。

## update-voice-connector

以下代码示例演示了如何使用 update-voice-connector。

### AWS CLI

更新 Amazon Chime 语音连接器的详细信息

以下 update-voice-connector 示例更新了指定的 Amazon Chime 语音连接器的名称。

```
aws chime update-voice-connector \
  --voice-connector-id abcdef1ghij2klmno3pqr4 \
  --name newName \
  --require-encryption
```

输出：

```
{
  "VoiceConnector": {
    "VoiceConnectorId": "abcdef1ghij2klmno3pqr4",
    "AwsRegion": "us-west-2",
    "Name": "newName",
    "OutboundHostName": "abcdef1ghij2klmno3pqr4.voiceconnector.chime.aws",
    "RequireEncryption": true,
    "CreatedTimestamp": "2019-09-18T20:34:01.352Z",
    "UpdatedTimestamp": "2019-09-18T20:40:52.895Z"
  }
}
```

有关更多信息，请参阅 [《亚马逊 Chime 管理指南》](#) 中的“使用 Amazon Chime 语音连接器”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateVoiceConnector](#)中的。

## 使用云控制 API 示例 AWS CLI

以下代码示例向您展示了如何使用 with Control API 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-resource**

以下代码示例演示了如何使用 create-resource。

AWS CLI

创建资源

以下create-resource示例创建了一个名为 AWS::Kinesis::Stream 的资源 ResourceExample，其保留期为 168 小时，分片计数为 3。

```
aws cloudcontrol create-resource \  
  --type-name AWS::Kinesis::Stream \  
  --desired-state "{\"Name\": \"ResourceExample\", \"RetentionPeriodHours\":168, \  
  \"ShardCount\":3}"
```

输出：

```
{  
  "ProgressEvent": {
```



```
    "EventTime": 1632506656.706,  
    "TypeName": "AWS::Kinesis::Stream",  
    "OperationStatus": "IN_PROGRESS",  
    "Operation": "CREATE",  
    "Identifier": "ResourceExample",  
    "RequestToken": "20999d87-e304-4725-ad84-832dcbfd7fc5"  
  }  
}
```

有关更多信息，请参阅《云控制 API 用户指南》中的[创建资源](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateResource](#) 中的。

## delete-resource

以下代码示例演示了如何使用 delete-resource。

### AWS CLI

#### 删除资源

以下 delete-resource 示例从您的账户中删除带有标识符 ResourceExample 的 AWS::Kinesis::Stream 资源。AWS

```
aws cloudcontrol delete-resource \  
  --type-name AWS::Kinesis::Stream \  
  --identifier ResourceExample
```

输出：

```
{  
  "ProgressEvent": {  
    "TypeName": "AWS::Kinesis::Stream",  
    "Identifier": "ResourceExample",  
    "RequestToken": "e48f26ff-d0f9-4ab8-a878-120db1edf111",  
    "Operation": "DELETE",  
    "OperationStatus": "IN_PROGRESS",  
    "EventTime": 1632950300.14  
  }  
}
```

有关更多信息，请参阅《云控制 API 用户指南》中的[删除资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteResource](#)中的。

## get-resource-request-status

以下代码示例演示了如何使用 `get-resource-request-status`。

### AWS CLI

获取资源请求的状态信息

以下`get-resource-request-status`示例返回有关指定资源请求的状态信息。

```
aws cloudcontrol get-resource-request-status \  
  --request-token "e1a6b86e-46bd-41ac-bfba-001234567890"
```

输出：

```
{  
  "ProgressEvent": {  
    "TypeName": "AWS::Kinesis::Stream",  
    "Identifier": "Demo",  
    "RequestToken": "e1a6b86e-46bd-41ac-bfba-001234567890",  
    "Operation": "CREATE",  
    "OperationStatus": "FAILED",  
    "EventTime": 1632950268.481,  
    "StatusMessage": "Resource of type 'AWS::Kinesis::Stream' with identifier  
'Demo' already exists.",  
    "ErrorCode": "AlreadyExists"  
  }  
}
```

有关更多信息，请参阅 Cloud Control API 用户指南中的[管理资源操作请求](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetResourceRequestStatus](#)中的。

## get-resource

以下代码示例演示了如何使用 `get-resource`。

### AWS CLI

获取资源的当前状态

以下 `get-resource` 示例返回名为 `:: Kinesis AWS:: Stream` 资源的当前状态。 `ResourceExample`

```
aws cloudcontrol get-resource \  
  --type-name AWS::Kinesis::Stream \  
  --identifier ResourceExample
```

输出：

```
{  
  "TypeName": "AWS::Kinesis::Stream",  
  "ResourceDescription": {  
    "Identifier": "ResourceExample",  
    "Properties": "{\"Arn\":\"arn:aws:kinesis:us-west-2:099908667365:stream/ResourceExample\", \"RetentionPeriodHours\":168, \"Name\":\"ResourceExample\", \"ShardCount\":3}"  
  }  
}
```

有关更多信息，请参阅 [Cloud Control API 用户指南中的读取资源的当前状态](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `GetResource`](#) 中的。

## list-resource-requests

以下代码示例演示了如何使用 `list-resource-requests`。

AWS CLI

列出活动资源操作请求

以下 `list-resource-requests` 示例列出了您的 AWS 账户中已失败的创建和更新操作的资源请求。

```
aws cloudcontrol list-resource-requests \  
  --resource-request-status-filter Operations=CREATE,OperationStatuses=FAILED
```

输出：

```
{  
  "ResourceRequestStatusSummaries": [  

```

```
{
  "TypeName": "AWS::Kinesis::Stream",
  "Identifier": "Demo",
  "RequestToken": "e1a6b86e-46bd-41ac-bfba-633abcdfdbd7",
  "Operation": "CREATE",
  "OperationStatus": "FAILED",
  "EventTime": 1632950268.481,
  "StatusMessage": "Resource of type 'AWS::Kinesis::Stream' with
identifier 'Demo' already exists.",
  "ErrorCode": "AlreadyExists"
}
]
```

有关更多信息，请参阅 Cloud Control API 用户指南中的[管理资源操作请求](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListResourceRequests](#)中的。

## list-resources

以下代码示例演示了如何使用 list-resources。

### AWS CLI

列出给定类型的资源

以下 list-resources 示例列出了在您的账户中配置的 AWS::Kinesis::Stream 资源。AWS

```
aws cloudcontrol list-resources \
  --type-name AWS::Kinesis::Stream
```

输出：

```
{
  "TypeName": "AWS::Kinesis::Stream",
  "ResourceDescriptions": [
    {
      "Identifier": "MyKinesisStream",
      "Properties": "{\"Name\": \"MyKinesisStream\"}"
    },
    {
      "Identifier": "AnotherStream",
```

```

        "Properties": "{\"Name\":\"AnotherStream\"}"
      }
    ]
  }

```

有关更多信息，请参阅《云控制 API 用户指南》中的[发现资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResources](#)中的。

## update-resource

以下代码示例演示了如何使用 update-resource。

### AWS CLI

更新现有资源的属性

以下update-resource示例将名为:: Logs AWS:: LogGroup 资源的保留策略更新 ExampleLogGroup 为 90 天。

```

aws cloudcontrol update-resource \
  --type-name AWS::Logs::LogGroup \
  --identifier ExampleLogGroup \
  --patch-document "[{\"op\":\"replace\",\"path\":\"/RetentionInDays\",\"value\":"
  \":90}]"

```

输出：

```

{
  "ProgressEvent": {
    "EventTime": "2021-08-09T18:17:15.219Z",
    "TypeName": "AWS::Logs::LogGroup",
    "OperationStatus": "IN_PROGRESS",
    "Operation": "UPDATE",
    "Identifier": "ExampleLogGroup",
    "RequestToken": "5f40c577-3534-4b20-9599-0b0123456789"
  }
}

```

有关更多信息，请参阅《云控制 API 用户指南》中的[更新资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateResource](#)中的。

## AWS Cloud Map 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS Cloud Map。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-private-dns-namespace**

以下代码示例演示了如何使用 `create-private-dns-namespace`。

AWS CLI

创建私有 DNS 命名空间

以下 `create-private-dns-namespace` 示例创建了一个私有 DNS 命名空间。

```
aws servicediscovery create-private-dns-namespace \  
  --name example.com \  
  --vpc vpc-1c56417b
```

输出：

```
{  
  "OperationId": "gv4g5meo7ndmeh4fqskygvk23d2fijwa-k9302yzd"  
}
```

要确认操作成功，可以运行 `get-operation`。有关更多信息，请参阅 [获取操作](#)。

有关更多信息，请参阅 AWS Cloud Map 开发者指南中的[创建命名空间](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreatePrivateDnsNamespace](#)中的。

## create-service

以下代码示例演示了如何使用 create-service。

### AWS CLI

#### 创建服务

以下 create-service 示例创建了一个服务。

```
aws servicediscovery create-service \  
  --name myservice \  
  --namespace-id ns-ylexjili4cdxy3xm \  
  --dns-config "NamespaceId=ns-  
ylexjili4cdxy3xm,RoutingPolicy=MULTIVALUE,DnsRecords=[{Type=A,TTL=60}]"
```

输出：

```
{  
  "Service": {  
    "Id": "srv-p5zdwlg5uvvzjita",  
    "Arn": "arn:aws:servicediscovery:us-west-2:803642222207:service/srv-  
p5zdwlg5uvvzjita",  
    "Name": "myservice",  
    "NamespaceId": "ns-ylexjili4cdxy3xm",  
    "DnsConfig": {  
      "NamespaceId": "ns-ylexjili4cdxy3xm",  
      "RoutingPolicy": "MULTIVALUE",  
      "DnsRecords": [  
        {  
          "Type": "A",  
          "TTL": 60  
        }  
      ]  
    },  
    "CreateDate": 1587081768.334,  
    "CreatorRequestId": "567c1193-6b00-4308-bd57-ad38a8822d25"  
  }  
}
```

有关更多信息，请参阅 AWS Cloud Map 开发者指南中的[创建服务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateService](#)中的。

## delete-namespace

以下代码示例演示了如何使用 delete-namespace。

### AWS CLI

#### 删除命名空间

以下delete-namespace示例删除命名空间。

```
aws servicediscovery delete-namespace \  
  --id ns-ylexjili4cdxy3xm
```

输出：

```
{  
  "OperationId": "gv4g5meo7ndmeh4fqskygvk23d2fijwa-k98y6drk"  
}
```

要确认操作成功，可以运行get-operation。有关更多信息，请参阅[获取操作](#)。

有关更多信息，请参阅 AWS Cloud Map 开发者指南中的[删除命名空间](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNamespace](#)中的。

## delete-service

以下代码示例演示了如何使用 delete-service。

### AWS CLI

#### 删除服务

以下delete-service示例删除服务。

```
aws servicediscovery delete-service \  
  --id srv-p5zdwlg5uvvzjita
```



此命令不生成任何输出。

有关更多信息，请参阅 AWS Cloud Map 开发者指南中的[删除服务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteService](#)中的。

## deregister-instance

以下代码示例演示了如何使用 deregister-instance。

### AWS CLI

#### 取消注册服务实例

以下deregister-instance示例取消注册服务实例。

```
aws servicediscovery deregister-instance \  
  --service-id srv-p5zdwlg5uvvzjita \  
  --instance-id myservice-53
```

输出：

```
{  
  "OperationId": "4yejorelbukcjzpnr6t1mrghsjwpngf4-k98rnaiq"  
}
```

要确认操作成功，可以运行get-operation。有关更多信息，请参阅[获取操作](#)。

有关更多信息，请参阅《AWS Cloud Map 开发者指南》中的[取消注册服务实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterInstance](#)中的。

## discover-instances

以下代码示例演示了如何使用 discover-instances。

### AWS CLI

#### 发现注册的实例

以下discover-instances示例发现注册的实例。

```
aws servicediscovery discover-instances \  
  --service-id myservice-53
```

```
--namespace-name example.com \  
--service-name myservice \  
--max-results 10 \  
--health-status ALL
```

输出：

```
{  
  "Instances": [  
    {  
      "InstanceId": "myservice-53",  
      "NamespaceName": "example.com",  
      "ServiceName": "myservice",  
      "HealthStatus": "UNKNOWN",  
      "Attributes": {  
        "AWS_INSTANCE_IPV4": "172.2.1.3",  
        "AWS_INSTANCE_PORT": "808"  
      }  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DiscoverInstances](#)中的。

## get-operation

以下代码示例演示了如何使用 get-operation。

### AWS CLI

获取操作结果

以下get-operation示例获取操作的结果。

```
aws servicediscovery get-operation \  
  --operation-id gv4g5meo7ndmeh4fqskygvk23d2fijwa-k9302yzd
```

输出：

```
{  
  "Operation": {  
    "Id": "gv4g5meo7ndmeh4fqskygvk23d2fijwa-k9302yzd",
```

```
    "Type": "CREATE_NAMESPACE",
    "Status": "SUCCESS",
    "CreateDate": 1587055860.121,
    "UpdateDate": 1587055900.469,
    "Targets": {
      "NAMESPACE": "ns-ylexjili4cdxy3xm"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetOperation](#)中的。

## list-instances

以下代码示例演示了如何使用 list-instances。

### AWS CLI

#### 列出服务实例

以下list-instances示例列出了服务实例。

```
aws servicediscovery list-instances \
  --service-id srv-qzpwvt2tfqcegapy
```

输出：

```
{
  "Instances": [
    {
      "Id": "i-06bdabbae60f65a4e",
      "Attributes": {
        "AWS_INSTANCE_IPV4": "172.2.1.3",
        "AWS_INSTANCE_PORT": "808"
      }
    }
  ]
}
```

有关更多信息，请参阅 [《AWS Cloud Map 开发者指南》中的查看服务实例列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListInstances](#)中的。

## list-namespaces

以下代码示例演示了如何使用 `list-namespaces`。

### AWS CLI

#### 列出命名空间

以下 `list-namespaces` 示例列出了命名空间。

```
aws servicediscovery list-namespaces
```

输出：

```
{
  "Namespaces": [
    {
      "Arn": "arn:aws:servicediscovery:us-west-2:123456789012:namespace/ns-a3ccy2e7e3a7rile",
      "CreateDate": 1585354387.357,
      "Id": "ns-a3ccy2e7e3a7rile",
      "Name": "local",
      "Properties": {
        "DnsProperties": {
          "HostedZoneId": "Z06752353VBUDTC32S84S"
        },
        "HttpProperties": {
          "HttpName": "local"
        }
      },
      "Type": "DNS_PRIVATE"
    },
    {
      "Arn": "arn:aws:servicediscovery:us-west-2:123456789012:namespace/ns-pocfyjtrsmwtvcxx",
      "CreateDate": 1586468974.698,
      "Description": "My second namespace",
      "Id": "ns-pocfyjtrsmwtvcxx",
      "Name": "My-second-namespace",
      "Properties": {
        "DnsProperties": {},
        "HttpProperties": {
          "HttpName": "My-second-namespace"
        }
      }
    }
  ]
}
```

```

        }
      },
      "Type": "HTTP"
    },
    {
      "Arn": "arn:aws:servicediscovery:us-west-2:123456789012:namespace/ns-
ylexjili4cdxy3xm",
      "CreateDate": 1587055896.798,
      "Id": "ns-ylexjili4cdxy3xm",
      "Name": "example.com",
      "Properties": {
        "DnsProperties": {
          "HostedZoneId": "Z09983722P0QME1B3KC8I"
        },
        "HttpProperties": {
          "HttpName": "example.com"
        }
      },
      "Type": "DNS_PRIVATE"
    }
  ]
}

```

有关更多信息，请参阅《AWS Cloud Map 开发者[指南](#)》中的[查看命名空间列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListNamespaces](#)中的。

## list-services

以下代码示例演示了如何使用 list-services。

### AWS CLI

#### 列出服务

以下list-services示例列出了服务。

```
aws servicediscovery list-services
```

输出：

```
{
```

```
"Services": [
  {
    "Id": "srv-p5zdwlg5uvvzjita",
    "Arn": "arn:aws:servicediscovery:us-west-2:123456789012:service/srv-
p5zdwlg5uvvzjita",
    "Name": "myservice",
    "DnsConfig": {
      "RoutingPolicy": "MULTIVALUE",
      "DnsRecords": [
        {
          "Type": "A",
          "TTL": 60
        }
      ]
    },
    "CreateDate": 1587081768.334
  }
]
```

有关更多信息，请参阅 [《AWS Cloud Map 开发者指南》中的查看服务列表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListServices](#) 中的。

## register-instance

以下代码示例演示了如何使用 register-instance。

### AWS CLI

#### 注册服务实例

以下 register-instance 示例注册了一个服务实例。

```
aws servicediscovery register-instance \
  --service-id srv-p5zdwlg5uvvzjita \
  --instance-id myservice-53 \
  --attributes=AWS_INSTANCE_IPV4=172.2.1.3,AWS_INSTANCE_PORT=808
```

输出：

```
{
  "OperationId": "4yejorelbukcjpnr6t1mrghsjwpngf4-k95yg2u7"
```

```
}
```

要确认操作成功，可以运行`get-operation`。有关更多信息，请参阅[获取操作](#)。

有关更多信息，请参阅 AWS Cloud Map 开发者指南中的[注册实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RegisterInstance](#)中的。

## AWS Cloud9 使用示例 AWS CLI

以下代码示例向您展示了如何使用`with`来执行操作和实现常见场景 AWS Cloud9。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-environment-ec2**

以下代码示例演示了如何使用 `create-environment-ec2`。

AWS CLI

创建 AWS Cloud9 EC2 开发环境

以下`create-environment-ec2`示例使用指定设置创建一个 AWS Cloud9 开发环境，启动亚马逊弹性计算云 (Amazon EC2) 实例，然后从该实例连接到环境。

```
aws cloud9 create-environment-ec2 \  
  --name my-demo-env \  
  --description "My demonstration development environment." \  
  --instance-type t2.micro --image-id amazonlinux-2023-x86_64 \  
  --
```

```
--subnet-id subnet-1fab8aEX \  
--automatic-stop-time-minutes 60 \  
--owner-arn arn:aws:iam::123456789012:user/MyDemoUser
```

输出：

```
{  
  "environmentId": "8a34f51ce1e04a08882f1e811bd706EX"  
}
```

有关更多信息，请参阅 AWS Cloud9 用户 [指南中的创建 EC2 环境](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [CreateEnvironmentEc2](#)。

## create-environment-membership

以下代码示例演示了如何使用 create-environment-membership。

### AWS CLI

向 AWS Cloud9 开发环境添加环境成员

此示例将指定的环境成员添加到指定的 AWS Cloud9 开发环境中。

命令：

```
aws cloud9 create-environment-membership --environment-id  
8a34f51ce1e04a08882f1e811bd706EX --user-arn arn:aws:iam::123456789012:user/  
AnotherDemoUser --permissions read-write
```

输出：

```
{  
  "membership": {  
    "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",  
    "userId": "AIDAJ3LOROMOUXTBSU6EX",  
    "userArn": "arn:aws:iam::123456789012:user/AnotherDemoUser",  
    "permissions": "read-write"  
  }  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateEnvironmentMembership](#) 中的。



## delete-environment-membership

以下代码示例演示了如何使用 delete-environment-membership。

### AWS CLI

从 AWS Cloud9 开发环境中删除环境成员

此示例从指定的 AWS Cloud9 开发环境中删除指定的环境成员。

命令:

```
aws cloud9 delete-environment-membership --environment-id
8a34f51ce1e04a08882f1e811bd706EX --user-arn arn:aws:iam::123456789012:user/
AnotherDemoUser
```

输出:

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteEnvironmentMembership](#)中的。

## delete-environment

以下代码示例演示了如何使用 delete-environment。

### AWS CLI

删除 C AWS Cloud9 开发环境

此示例删除了指定的 AWS Cloud9 开发环境。如果 Amazon EC2 实例已连接到环境，也会终止该实例。

命令:

```
aws cloud9 delete-environment --environment-id 8a34f51ce1e04a08882f1e811bd706EX
```

输出:

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteEnvironment](#)中的。

## describe-environment-memberships

以下代码示例演示了如何使用 `describe-environment-memberships`。

### AWS CLI

获取有关 AWS Cloud9 开发环境的环境成员的信息

此示例获取有关指定 AWS Cloud9 开发环境的环境成员的信息。

命令:

```
aws cloud9 describe-environment-memberships --environment-id
8a34f51ce1e04a08882f1e811bd706EX
```

输出:

```
{
  "memberships": [
    {
      "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
      "userId": "AIDAJ3LOROMOUCTBSU6EX",
      "userArn": "arn:aws:iam::123456789012:user/AnotherDemoUser",
      "permissions": "read-write"
    },
    {
      "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
      "userId": "AIDAJNUEDQAQWFELJDLEX",
      "userArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "permissions": "owner"
    }
  ]
}
```

获取有关 AWS Cloud9 开发环境所有者的信息

此示例获取有关指定 AWS Cloud9 开发环境所有者的信息。

命令:

```
aws cloud9 describe-environment-memberships --environment-id
8a34f51ce1e04a08882f1e811bd706EX --permissions owner
```

输出：

```
{
  "memberships": [
    {
      "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
      "userId": "AIDAJNUEDQAQWFELJDLEX",
      "userArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "permissions": "owner"
    }
  ]
}
```

获取有关多个 AWS Cloud9 开发环境的环境成员的信息

此示例获取有关多个 AWS Cloud9 开发环境的指定环境成员的信息。

命令：

```
aws cloud9 describe-environment-memberships --user-arn
arn:aws:iam::123456789012:user/MyDemoUser
```

输出：

```
{
  "memberships": [
    {
      "environmentId": "10a75714bd494714929e7f5ec4125aEX",
      "lastAccess": 1516213427.0,
      "userId": "AIDAJNUEDQAQWFELJDLEX",
      "userArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "permissions": "owner"
    },
    {
      "environmentId": "1980b80e5f584920801c09086667f0EX",
      "lastAccess": 1516144884.0,
      "userId": "AIDAJNUEDQAQWFELJDLEX",
      "userArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "permissions": "owner"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEnvironmentMemberships](#)中的。

## describe-environment-status

以下代码示例演示了如何使用 `describe-environment-status`。

### AWS CLI

获取 AWS Cloud9 开发环境的状态信息

此示例获取指定 AWS Cloud9 开发环境的状态信息。

命令:

```
aws cloud9 describe-environment-status --environment-id
685f892f431b45c2b28cb69eadcdb0EX
```

输出：

```
{
  "status": "ready",
  "message": "Environment is ready to use"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEnvironmentStatus](#)中的。

## describe-environments

以下代码示例演示了如何使用 `describe-environments`。

### AWS CLI

获取有关 AWS Cloud9 开发环境的信息

此示例获取有关指定 AWS Cloud9 开发环境的信息。

命令:

```
aws cloud9 describe-environments --environment-ids 685f892f431b45c2b28cb69eadcdb0EX
349c86d4579e4e7298d500ff57a6b2EX
```

输出：

```
{
  "environments": [
    {
      "id": "685f892f431b45c2b28cb69eadcdb0EX",
      "name": "my-demo-ec2-env",
      "description": "Created from CodeStar.",
      "type": "ec2",
      "arn": "arn:aws:cloud9:us-east-1:123456789012:environment:685f892f431b45c2b28cb69eadcdb0EX",
      "ownerArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "lifecycle": {
        "status": "CREATED"
      }
    },
    {
      "id": "349c86d4579e4e7298d500ff57a6b2EX",
      "name": "my-demo-ssh-env",
      "description": "",
      "type": "ssh",
      "arn": "arn:aws:cloud9:us-east-1:123456789012:environment:349c86d4579e4e7298d500ff57a6b2EX",
      "ownerArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "lifecycle": {
        "status": "CREATED"
      }
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEnvironments](#)中的。

## list-environments

以下代码示例演示了如何使用 list-environments。

### AWS CLI

获取可用的 AWS Cloud9 开发环境标识符列表

此示例获取可用的 AWS Cloud9 开发环境标识符列表。

命令:

```
aws cloud9 list-environments
```

输出:

```
{
  "environmentIds": [
    "685f892f431b45c2b28cb69eadcdb0EX",
    "1980b80e5f584920801c09086667f0EX"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListEnvironments](#)中的。

## update-environment-membership

以下代码示例演示了如何使用 update-environment-membership。

AWS CLI

更改 AWS Cloud9 开发环境中现有环境成员的设置

此示例更改了指定 AWS Cloud9 开发环境的指定现有环境成员的设置。

命令:

```
aws cloud9 update-environment-membership --environment-id
8a34f51ce1e04a08882f1e811bd706EX --user-arn arn:aws:iam::123456789012:user/
AnotherDemoUser --permissions read-only
```

输出:

```
{
  "membership": {
    "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
    "userId": "AIDAJ3LOROMOUXTBSUGEX",
    "userArn": "arn:aws:iam::123456789012:user/AnotherDemoUser",
    "permissions": "read-only"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateEnvironmentMembership](#)中的。

## update-environment

以下代码示例演示了如何使用 update-environment。

### AWS CLI

更改现有 AWS Cloud9 开发环境的设置

此示例更改了指定现有 AWS Cloud9 开发环境的指定设置。

命令:

```
aws cloud9 update-environment --environment-id 8a34f51ce1e04a08882f1e811bd706EX
--name my-changed-demo-env --description "My changed demonstration development
environment."
```

输出 :

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateEnvironment](#)中的。

## AWS CloudFormation 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS CloudFormation。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

## 操作

### activate-type

以下代码示例演示了如何使用 activate-type。

#### AWS CLI

##### 激活类型

以下activate-type示例激活了公共第三方扩展，使其可用于堆栈模板。

```
aws cloudformation activate-type \  
  --region us-west-2 \  
  --type RESOURCE \  
  --type-name Example::Test::1234567890abcdef0 \  
  --type-name-alias Example::Test::Alias
```

##### 输出：

```
{  
  "Arn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/Example-  
Test-Alias"  
}
```

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》中的使用 AWS CloudFormation 注册表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ActivateType](#)中的。

### batch-describe-type-configurations

以下代码示例演示了如何使用 batch-describe-type-configurations。

#### AWS CLI

##### 批量描述类型配置

以下batch-describe-type-configurations示例为该类型配置数据。

```
aws cloudformation batch-describe-type-configurations \  
  --region us-west-2 \  
  --type-name Example::Test::1234567890abcdef0
```



```
--type-configuration-identifiers TypeArn="arn:aws:cloudformation:us-west-2:123456789012:type/resource/Example-Test-Type,TypeConfigurationAlias=MyConfiguration"
```

输出：

```
{
  "Errors": [],
  "UnprocessedTypeConfigurations": [],
  "TypeConfigurations": [
    {
      "Arn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/Example-Test-Type",
      "Alias": "MyConfiguration",
      "Configuration": "{\n  \"Example\": {\n    \"ApiKey\": \"examplekey\", \n    \"ApplicationKey\": \"examplekey1\", \n    \"ApiURL\": \"exampleurl\" \n  }\n}",
      "LastUpdated": "2021-10-01T15:25:46.210000+00:00",
      "TypeArn": "arn:aws:cloudformation:us-east-1:123456789012:type/resource/Example-Test-Type"
    }
  ]
}
```

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》](#) 中的使用 AWS CloudFormation 注册表。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchDescribeTypeConfigurations](#) 中的。

## cancel-update-stack

以下代码示例演示了如何使用 cancel-update-stack。

### AWS CLI

取消正在进行的堆栈更新

以下 cancel-update-stack 命令会取消 myteststack 堆栈上的堆栈更新：

```
aws cloudformation cancel-update-stack --stack-name myteststack
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CancelUpdateStack](#) 中的。

## continue-update-rollback

以下代码示例演示了如何使用 `continue-update-rollback`。

### AWS CLI

#### 重试更新回滚

以下 `continue-update-rollback` 示例从之前失败的堆栈更新中恢复回滚操作。

```
aws cloudformation continue-update-rollback \  
  --stack-name my-stack
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ContinueUpdateRollback](#)中的。

## create-change-set

以下代码示例演示了如何使用 `create-change-set`。

### AWS CLI

#### 创建更改集

以下`create-change-set`示例创建了一个具有该CAPABILITY\_IAM功能的更改集。该文件`template.yaml`是当前文件夹中的一个 AWS CloudFormation 模板，用于定义包含 IAM 资源的堆栈。

```
aws cloudformation create-change-set \  
  --stack-name my-application \  
  --change-set-name my-change-set \  
  --template-body file://template.yaml \  
  --capabilities CAPABILITY_IAM
```

输出：

```
{  
  "Id": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-change-set/  
bc9555ba-a949-xmpl-bfb8-f41d04ec5784",  
  "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-application/  
d0a825a0-e4cd-xmpl-b9fb-061c69e99204"
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateChangeSet](#)中的。

## create-stack-instances

以下代码示例演示了如何使用 `create-stack-instances`。

### AWS CLI

#### 创建堆栈实例

以下`create-stack-instances`示例在两个账户和四个区域中创建堆栈集的实例。容错设置可确保在所有账户和地区尝试更新，即使某些堆栈无法创建。

```
aws cloudformation create-stack-instances \  
  --stack-set-name my-stack-set \  
  --accounts 123456789012 223456789012 \  
  --regions us-east-1 us-east-2 us-west-1 us-west-2 \  
  --operation-preferences FailureToleranceCount=7
```

输出：

```
{  
  "OperationId": "d7995c31-83c2-xmpl-a3d4-e9ca2811563f"  
}
```

要创建堆栈集，请使用`create-stack-set`命令。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateStackInstances](#)中的。

## create-stack-set

以下代码示例演示了如何使用 `create-stack-set`。

### AWS CLI

#### 创建堆栈集

以下`create-stack-set`示例使用指定的 YAML 文件模板创建堆栈集。`template.yaml`是当前文件夹中用于定义堆栈的 AWS CloudFormation 模板。

```
aws cloudformation create-stack-set \  
  --stack-set-name my-stack-set \  
  --template-body file://template.yaml \  
  --description "SNS topic"
```

输出：

```
{  
  "StackSetId": "my-stack-set:8d0f160b-d157-xmpl-a8e6-c0ce8e5d8cc1"  
}
```

要将堆栈实例添加到堆栈集中，请使用 `create-stack-instances` 命令。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateStackSet](#) 中的。

## create-stack

以下代码示例演示了如何使用 `create-stack`。

### AWS CLI

创建 AWS CloudFormation 堆栈

以下 `create-stacks` 命令使用 `sampletemplate.json` 模板创建了名为 `myteststack` 的堆栈：

```
aws cloudformation create-stack --stack-name myteststack --template-body file://  
sampletemplate.json --parameters ParameterKey=KeyValuePair,ParameterValue=TestKey  
ParameterKey=SubnetIDs,ParameterValue=SubnetID1\\,SubnetID2
```

输出：

```
{  
  "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/  
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"  
}
```

有关更多信息，请参阅《AWS CloudFormation 用户指南》中的堆栈。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateStack](#) 中的。

## deactivate-type

以下代码示例演示了如何使用 deactivate-type。

### AWS CLI

#### 停用文字

以下deactivate-type示例停用了先前在此账户和区域中激活的公共扩展程序。

```
aws cloudformation deactivate-type \  
  --region us-west-2 \  
  --type MODULE \  
  --type-name Example::Test::Type::MODULE
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》中的使用 AWS CloudFormation 注册表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeactivateType](#)中的。

## delete-change-set

以下代码示例演示了如何使用 delete-change-set。

### AWS CLI

#### 删除更改集

以下delete-change-set示例通过指定更改集名称和堆栈名称来删除更改集。

```
aws cloudformation delete-change-set \  
  --stack-name my-stack \  
  --change-set-name my-change-set
```

此命令不生成任何输出。

以下delete-change-set示例通过指定变更集的完整 ARN 来删除变更集。

```
aws cloudformation delete-change-set \  
  --change-set-name arn:aws:cloudformation:us-east-2:123456789012:changeSet/my-  
change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteChangeSet](#)中的。

## delete-stack-instances

以下代码示例演示了如何使用 delete-stack-instances。

### AWS CLI

#### 删除堆栈实例

以下delete-stack-instances示例删除两个区域中两个账户中的堆栈集实例并终止堆栈。

```
aws cloudformation delete-stack-instances \  
  --stack-set-name my-stack-set \  
  --accounts 123456789012 567890123456 \  
  --regions us-east-1 us-west-1 \  
  --no-retain-stacks
```

输出：

```
{  
  "OperationId": "ad49f10c-fd1d-413f-a20a-8de6e2fa8f27"  
}
```

要删除空堆栈集，请使用delete-stack-set命令。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteStackInstances](#)中的。

## delete-stack-set

以下代码示例演示了如何使用 delete-stack-set。

### AWS CLI

#### 删除堆栈集

以下命令删除指定的空堆栈集。堆栈集必须为空。

```
aws cloudformation delete-stack-set \  
  --stack-set-name my-stack-set
```

```
--stack-set-name my-stack-set
```

此命令不生成任何输出。

要从堆栈集中删除实例，请使用 `delete-stack-instances` 命令。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteStackSet](#) 中的。

## delete-stack

以下代码示例演示了如何使用 `delete-stack`。

### AWS CLI

#### 删除堆栈

以下 `delete-stack` 示例将删除指定的堆栈。

```
aws cloudformation delete-stack \  
  --stack-name my-stack
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteStack](#) 中的。

## deploy

以下代码示例演示了如何使用 `deploy`。

### AWS CLI

以下命令将名为的模板部署 `template.json` 到名为的堆栈中：`my-new-stack`

```
aws cloudformation deploy --template-file /path_to_template/template.json --stack-  
name my-new-stack --parameter-overrides Key1=Value1 Key2=Value2 --tags Key1=Value1  
Key2=Value2
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [“部署”](#)。

## deregister-type

以下代码示例演示了如何使用 `deregister-type`。

## AWS CLI

### 取消注册类型版本

以下deregister-type示例将指定类型的版本从 CloudFormation 注册表中移除，使其无法再用于 CloudFormation 操作。

```
aws cloudformation deregister-type \  
  --type RESOURCE \  
  --type-name My::Logs::LogGroup \  
  --version-id 00000002
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》中的使用 CloudFormation 注册表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterType](#)中的。

## describe-account-limits

以下代码示例演示了如何使用 describe-account-limits。

### AWS CLI

获取有关您的账户限额的信息

以下命令检索当前账户的区域限制列表。

```
aws cloudformation describe-account-limits
```

输出：

```
{  
  "AccountLimits": [  
    {  
      "Name": "StackLimit",  
      "Value": 200  
    },  
    {  
      "Name": "StackOutputsLimit",  
      "Value": 60  
    }  
  ]  
}
```



```

    },
    {
      "Name": "ConcurrentResourcesLimit",
      "Value": 2500
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAccountLimits](#)中的。

## describe-change-set

以下代码示例演示了如何使用 describe-change-set。

### AWS CLI

获取有关变更集的信息

以下describe-change-set示例显示了由更改集名称和堆栈名称指定的更改集的详细信息。

```

aws cloudformation describe-change-set \
  --change-set-name my-change-set \
  --stack-name my-stack

```

以下describe-change-set示例显示了由变更集的完整 ARN 指定的变更集的详细信息：

```

aws cloudformation describe-change-set \
  --change-set-name arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-
change-set/bc9555ba-a949-xmpl-bfb8-f41d04ec5784

```

输出：

```

{
  "Changes": [
    {
      "Type": "Resource",
      "ResourceChange": {
        "Action": "Modify",
        "LogicalResourceId": "function",
        "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
        "ResourceType": "AWS::Lambda::Function",

```

```

        "Replacement": "False",
        "Scope": [
            "Properties"
        ],
        "Details": [
            {
                "Target": {
                    "Attribute": "Properties",
                    "Name": "Timeout",
                    "RequiresRecreation": "Never"
                },
                "Evaluation": "Static",
                "ChangeSource": "DirectModification"
            }
        ]
    }
}
],
"ChangeSetName": "my-change-set",
"ChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-
change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
"StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/
d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
"StackName": "my-stack",
"Description": null,
"Parameters": null,
"CreationTime": "2019-10-02T05:20:56.651Z",
"ExecutionStatus": "AVAILABLE",
"Status": "CREATE_COMPLETE",
"StatusReason": null,
"NotificationARNs": [],
"RollbackConfiguration": {},
"Capabilities": [
    "CAPABILITY_IAM"
],
"Tags": null
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeChangeSet](#)中的。

## describe-publisher

以下代码示例演示了如何使用 describe-publisher。

## AWS CLI

### 描述出版商

以下describe-publisher示例为发布者配置信息。

```
aws cloudformation describe-publisher \  
  --region us-west-2 \  
  --publisher-id 000q6TfUovXsEMmgKowxDZLLwqr2QUsh
```

输出：

```
{  
  "PublisherId": "000q6TfUovXsEMmgKowxDZLLwqr2QUshd2e75c8c",  
  "PublisherStatus": "VERIFIED",  
  "IdentityProvider": "AWS_Marketplace",  
  "PublisherProfile": "https://aws.amazon.com/marketplace/seller-profile?  
id=2c5dc1f0-17cd-4259-8e46-822a83gdtegd"  
}
```

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》中的使用 AWS CloudFormation 注册表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePublisher](#)中的。

## describe-stack-drift-detection-status

以下代码示例演示了如何使用 describe-stack-drift-detection-status。

### AWS CLI

#### 检查偏移检测操作的状态

以下describe-stack-drift-detection-status示例显示了偏差检测操作的状态。获取运行detect-stack-drift命令的 by ID。

```
aws cloudformation describe-stack-drift-detection-status \  
  --stack-drift-detection-id 1a229160-e4d9-xmpl-ab67-0a4f93df83d4
```

输出：

```
{
  "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
  "StackDriftDetectionId": "1a229160-e4d9-xmpl-ab67-0a4f93df83d4",
  "StackDriftStatus": "DRIFTED",
  "DetectionStatus": "DETECTION_COMPLETE",
  "DriftedStackResourceCount": 1,
  "Timestamp": "2019-10-02T05:54:30.902Z"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStackDriftDetectionStatus](#)中的。

## describe-stack-events

以下代码示例演示了如何使用 describe-stack-events。

### AWS CLI

#### 描述堆栈事件

以下 describe-stack-events 示例显示了指定堆栈的 2 个最新事件。

```
aws cloudformation describe-stack-events \
  --stack-name my-stack \
  --max-items 2

{
  "StackEvents": [
    {
      "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
      "EventId": "4e1516d0-e4d6-xmpl-b94f-0a51958a168c",
      "StackName": "my-stack",
      "LogicalResourceId": "my-stack",
      "PhysicalResourceId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
      "ResourceType": "AWS::CloudFormation::Stack",
      "Timestamp": "2019-10-02T05:34:29.556Z",
      "ResourceStatus": "UPDATE_COMPLETE"
    },
    {
      "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
```

```

        "EventId": "4dd3c810-e4d6-xmpl-bade-0aaf8b31ab7a",
        "StackName": "my-stack",
        "LogicalResourceId": "my-stack",
        "PhysicalResourceId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
        "ResourceType": "AWS::CloudFormation::Stack",
        "Timestamp": "2019-10-02T05:34:29.127Z",
        "ResourceStatus": "UPDATE_COMPLETE_CLEANUP_IN_PROGRESS"
    }
],
    "NextToken": "eyJ0ZXh0VG9XMPLi0iBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ=="
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStackEvents](#)中的。

## describe-stack-instance

以下代码示例演示了如何使用 describe-stack-instance。

### AWS CLI

#### 描述堆栈实例

以下命令描述了指定账户和区域中指定堆栈集的实例。堆栈集在当前区域和账户中，实例在账户中的us-west-2区域中123456789012。：

```

aws cloudformation describe-stack-instance \
  --stack-set-name my-stack-set \
  --stack-instance-account 123456789012 \
  --stack-instance-region us-west-2

```

输出：

```

{
  "StackInstance": {
    "StackSetId": "enable-config:296a3360-xmpl-40af-be78-9341e95bf743",
    "Region": "us-west-2",
    "Account": "123456789012",
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
StackSet-enable-config-e6cac20f-xmpl-46e9-8314-53e0d4591532/4287f9a0-e615-
xmpl-894a-12b31d3117be",
    "ParameterOverrides": [],

```

```

    "Status": "OUTDATED",
    "StatusReason": "ResourceLogicalId:ConfigBucket,
ResourceType:AWS::S3::Bucket, ResourceStatusReason:You have attempted to create
more buckets than allowed (Service: Amazon S3; Status Code: 400; Error Code:
TooManyBuckets; Request ID: F7F21CXMPL580224; S3 Extended Request ID: egd/
Fdt89BXMPLYiqbMNLjVvk55Yqqvi3NYW2nKLUVWhUGEHnfCmZdyj9671hriaG/dWMobS040o=)."
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStackInstance](#)中的。

## describe-stack-resource-drifts

以下代码示例演示了如何使用 `describe-stack-resource-drifts`。

### AWS CLI

获取有关偏离堆栈定义的资源的信息

以下命令显示有关指定堆栈的漂移资源的信息。要启动偏移检测，请使用`detect-stack-drift`命令。：

```
aws cloudformation describe-stack-resource-drifts \
  --stack-name my-stack
```

输出显示了一个经过修改的 AWS Lambda 函数：out-of-band

```

{
  "StackResourceDrifts": [
    {
      "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
      "LogicalResourceId": "function",
      "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
      "ResourceType": "AWS::Lambda::Function",
      "ExpectedProperties": "{\"Description\":\"Write a file to S3.\",
\\\"Environment\\\":{\\\"Variables\\\":{\\\"bucket\\\":\\\"my-stack-bucket-1vc62xmplgguf
\\\"}},\\\"Handler\\\":\\\"index.handler\\\",\\\"MemorySize\\\":128,\\\"Role\\\":
\\\"arn:aws:iam:123456789012:role/my-functionRole-HIZXMPLE0M9E\\\",\\\"Runtime\\\":
\\\"nodejs10.x\\\",\\\"Tags\\\":[{\\\"Key\\\":\\\"lambda:createdBy\\\",\\\"Value\\\":\\\"SAM\\\"}],\\\"Timeout
\\\":900,\\\"TracingConfig\\\":{\\\"Mode\\\":\\\"Active\\\"}}",

```

```

    "ActualProperties": "{\"Description\": \"Write a file to S3.\",
  \Environment\": {\"Variables\": {\"bucket\": \"my-stack-bucket-1vc62xmplgguf
  \}}, \"Handler\": \"index.handler\", \"MemorySize\": 256, \"Role\":
  \"arn:aws:iam::123456789012:role/my-functionRole-HIZXMPLE0M9E\", \"Runtime\":
  \"nodejs10.x\", \"Tags\": [{\"Key\": \"lambda:createdBy\", \"Value\": \"SAM\"}], \"Timeout
  \": 22, \"TracingConfig\": {\"Mode\": \"Active\"}}\",
    "PropertyDifferences": [
      {
        "PropertyPath": "/MemorySize",
        "ExpectedValue": "128",
        "ActualValue": "256",
        "DifferenceType": "NOT_EQUAL"
      },
      {
        "PropertyPath": "/Timeout",
        "ExpectedValue": "900",
        "ActualValue": "22",
        "DifferenceType": "NOT_EQUAL"
      }
    ],
    "StackResourceDriftStatus": "MODIFIED",
    "Timestamp": "2019-10-02T05:54:44.064Z"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStackResourceDrifts](#)中的。

## describe-stack-resource

以下代码示例演示了如何使用 describe-stack-resource。

### AWS CLI

获取有关堆栈资源的信息

以下 describe-stack-resource 示例显示了指定堆栈中名为 MyFunction 的资源的详细信息。

```

aws cloudformation describe-stack-resource \
  --stack-name MyStack \
  --logical-resource-id MyFunction

```

输出：

```
{
  "StackResourceDetail": {
    "StackName": "MyStack",
    "StackId": "arn:aws:cloudformation:us-east-2:123456789012:stack/MyStack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
    "LogicalResourceId": "MyFunction",
    "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
    "ResourceType": "AWS::Lambda::Function",
    "LastUpdatedTimestamp": "2019-10-02T05:34:27.989Z",
    "ResourceStatus": "UPDATE_COMPLETE",
    "Metadata": "{}",
    "DriftInformation": {
      "StackResourceDriftStatus": "IN_SYNC"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStackResource](#)中的。

## describe-stack-resources

以下代码示例演示了如何使用 describe-stack-resources。

### AWS CLI

获取有关堆栈资源的信息

以下 describe-stack-resources 示例显示了指定堆栈中的资源详细信息。

```
aws cloudformation describe-stack-resources \
  --stack-name my-stack
```

输出：

```
{
  "StackResources": [
    {
      "StackName": "my-stack",
      "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
```



```

    "LogicalResourceId": "bucket",
    "PhysicalResourceId": "my-stack-bucket-1vc62xmplgguf",
    "ResourceType": "AWS::S3::Bucket",
    "Timestamp": "2019-10-02T04:34:11.345Z",
    "ResourceStatus": "CREATE_COMPLETE",
    "DriftInformation": {
      "StackResourceDriftStatus": "IN_SYNC"
    }
  },
  {
    "StackName": "my-stack",
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
    "LogicalResourceId": "function",
    "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
    "ResourceType": "AWS::Lambda::Function",
    "Timestamp": "2019-10-02T05:34:27.989Z",
    "ResourceStatus": "UPDATE_COMPLETE",
    "DriftInformation": {
      "StackResourceDriftStatus": "IN_SYNC"
    }
  },
  {
    "StackName": "my-stack",
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
    "LogicalResourceId": "functionRole",
    "PhysicalResourceId": "my-functionRole-HIZXMPLE0M9E",
    "ResourceType": "AWS::IAM::Role",
    "Timestamp": "2019-10-02T04:34:06.350Z",
    "ResourceStatus": "CREATE_COMPLETE",
    "DriftInformation": {
      "StackResourceDriftStatus": "IN_SYNC"
    }
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStackResources](#)中的。

## describe-stack-set-operation

以下代码示例演示了如何使用 describe-stack-set-operation。

## AWS CLI

获取有关堆栈集操作的信息

以下 `describe-stack-set-operation` 示例显示了对指定堆栈集执行更新操作的详细信息。

```
aws cloudformation describe-stack-set-operation \  
  --stack-set-name enable-config \  
  --operation-id 35d45ebc-ed88-xmpl-ab59-0197a1fc83a0
```

输出：

```
{  
  "StackSetOperation": {  
    "OperationId": "35d45ebc-ed88-xmpl-ab59-0197a1fc83a0",  
    "StackSetId": "enable-config:296a3360-xmpl-40af-be78-9341e95bf743",  
    "Action": "UPDATE",  
    "Status": "SUCCEEDED",  
    "OperationPreferences": {  
      "RegionOrder": [  
        "us-east-1",  
        "us-west-2",  
        "eu-west-1",  
        "us-west-1"  
      ],  
      "FailureToleranceCount": 7,  
      "MaxConcurrentCount": 2  
    },  
    "AdministrationRoleARN": "arn:aws:iam::123456789012:role/  
AWSCloudFormationStackSetAdministrationRole",  
    "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole",  
    "CreationTimestamp": "2019-10-03T16:28:44.377Z",  
    "EndTimestamp": "2019-10-03T16:42:08.607Z"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStackSetOperation](#)中的。

## describe-stack-set

以下代码示例演示了如何使用 `describe-stack-set`。

## AWS CLI

获取有关堆栈集的信息

以下 `describe-stack-set` 示例显示有关指定堆栈集的详细信息。

```
aws cloudformation describe-stack-set \  
  --stack-set-name my-stack-set
```

输出：

```
{  
  "StackSet": {  
    "StackSetName": "my-stack-set",  
    "StackSetId": "my-stack-set:296a3360-xmpl-40af-be78-9341e95bf743",  
    "Description": "Create an Amazon SNS topic",  
    "Status": "ACTIVE",  
    "TemplateBody": "AWSTemplateFormatVersion: '2010-09-09'\nDescription: An AWS  
SNS topic\nResources:\n  topic:\n    Type: AWS::SNS::Topic",  
    "Parameters": [],  
    "Capabilities": [],  
    "Tags": [],  
    "StackSetARN": "arn:aws:cloudformation:us-west-2:123456789012:stackset/  
enable-config:296a3360-xmpl-40af-be78-9341e95bf743",  
    "AdministrationRoleARN": "arn:aws:iam::123456789012:role/  
AWSCloudFormationStackSetAdministrationRole",  
    "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStackSet](#)中的。

## describe-stacks

以下代码示例演示了如何使用 `describe-stacks`。

## AWS CLI

描述 AWS CloudFormation 堆栈

以下 `describe-stacks` 命令显示了 `myteststack` 堆栈的摘要信息：

```
aws cloudformation describe-stacks --stack-name myteststack
```

输出：

```
{
  "Stacks": [
    {
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
      "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample template showing how to create a publicly accessible S3 bucket. **WARNING** This template creates an S3 bucket. You will be billed for the AWS resources used if you create a stack from this template.",
      "Tags": [],
      "Outputs": [
        {
          "Description": "Name of S3 bucket to hold website content",
          "OutputKey": "BucketName",
          "OutputValue": "myteststack-s3bucket-jssofilzie2w"
        }
      ],
      "StackStatusReason": null,
      "CreationTime": "2013-08-23T01:02:15.422Z",
      "Capabilities": [],
      "StackName": "myteststack",
      "StackStatus": "CREATE_COMPLETE",
      "DisableRollback": false
    }
  ]
}
```

有关更多信息，请参阅《AWS CloudFormation 用户指南》中的堆栈。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStacks](#)中的。

## describe-type-registration

以下代码示例演示了如何使用 describe-type-registration。

AWS CLI

显示类型注册信息

以下describe-type-registration示例显示有关指定类型注册的信息，包括该类型的当前状态、类型和版本。

```
aws cloudformation describe-type-registration \  
  --registration-token a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "ProgressStatus": "COMPLETE",  
  "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/My-Logs-LogGroup",  
  "Description": "Deployment is currently in DEPLOY_STAGE of status COMPLETED; ",  
  "TypeVersionArn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/My-Logs-LogGroup/00000001"  
}
```

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》中的使用 CloudFormation 注册表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTypeRegistration](#)中的。

## describe-type

以下代码示例演示了如何使用 describe-type。

### AWS CLI

显示类型信息

以下describe-type示例显示了指定类型的信息。

```
aws cloudformation describe-type \  
  --type-name My::Logs::LogGroup \  
  --type RESOURCE
```

输出：

```
{  
  "SourceUrl": "https://github.com/aws-cloudformation/aws-cloudformation-resource-providers-logs.git",  
  "Description": "Customized resource derived from AWS::Logs::LogGroup",  
  "TimeCreated": "2019-12-03T23:29:33.321Z",  
}
```

```
"Visibility": "PRIVATE",
"TypeName": "My::Logs::LogGroup",
"LastUpdated": "2019-12-03T23:29:33.321Z",
"DeprecatedStatus": "LIVE",
"ProvisioningType": "FULLY_MUTABLE",
"Type": "RESOURCE",
"Arn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/My-Logs-
LogGroup/00000001",
"Schema": "[details omitted]"
}
```

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》中的使用 CloudFormation 注册表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeType](#) 中的。

## detect-stack-drift

以下代码示例演示了如何使用 detect-stack-drift。

### AWS CLI

#### 检测漂移的资源

以下 detect-stack-drift 示例启动指定堆栈的偏差检测。

```
aws cloudformation detect-stack-drift \
  --stack-name my-stack
```

输出：

```
{
  "StackDriftDetectionId": "1a229160-e4d9-xmpl-ab67-0a4f93df83d4"
}
```

然后，您可以将此 ID 与 describe-stack-resource-drifts 命令一起使用来描述漂移的资源。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DetectStackDrift](#) 中的。

## detect-stack-resource-drift

以下代码示例演示了如何使用 detect-stack-resource-drift。

## AWS CLI

### 检测资源的偏差

以下 `detect-stack-resource-drift` 示例检查在名为 `drift` 的堆栈 `MyFunction` 中命 `MyStack` 名的资源：

```
aws cloudformation detect-stack-resource-drift \  
  --stack-name MyStack \  
  --logical-resource-id MyFunction
```

输出显示了一个经过修改的 AWS Lambda 函数：`out-of-band`

```
{  
  "StackResourceDrift": {  
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/MyStack/  
d0a825a0-e4cd-xmpl-b9fb-061c69e99204",  
    "LogicalResourceId": "MyFunction",  
    "PhysicalResourceId": "my-function-SEZV4XMPL4S5",  
    "ResourceType": "AWS::Lambda::Function",  
    "ExpectedProperties": "{\"Description\":\"Write a file to S3.\",  
\\\"Environment\\\":{\\\"Variables\\\":{\\\"bucket\\\":\\\"my-stack-bucket-1vc62xmplgguf  
\\\"}},\\\"Handler\\\":\\\"index.handler\\\",\\\"MemorySize\\\":128,\\\"Role\\\":  
\\\"arn:aws:iam::123456789012:role/my-functionRole-HIZXMPLEOM9E\\\",\\\"Runtime\\\":  
\\\"nodejs10.x\\\",\\\"Tags\\\":[{\\\"Key\\\":\\\"lambda:createdBy\\\",\\\"Value\\\":\\\"SAM\\\"}],\\\"Timeout  
\\\":900,\\\"TracingConfig\\\":{\\\"Mode\\\":\\\"Active\\\"}}",  
    "ActualProperties": "{\"Description\":\"Write a file to S3.\",\\\"Environment  
\\\":{\\\"Variables\\\":{\\\"bucket\\\":\\\"my-stack-bucket-1vc62xmplgguf\\\"}},\\\"Handler\\\":  
\\\"index.handler\\\",\\\"MemorySize\\\":256,\\\"Role\\\":\\\"arn:aws:iam:123456789012:role/  
my-functionRole-HIZXMPLEOM9E\\\",\\\"Runtime\\\":\\\"nodejs10.x\\\",\\\"Tags\\\":[{\\\"Key\\\":  
\\\"lambda:createdBy\\\",\\\"Value\\\":\\\"SAM\\\"}],\\\"Timeout\\\":22,\\\"TracingConfig\\\":{\\\"Mode\\\":  
\\\"Active\\\"}}",  
    "PropertyDifferences": [  
      {  
        "PropertyPath": "/MemorySize",  
        "ExpectedValue": "128",  
        "ActualValue": "256",  
        "DifferenceType": "NOT_EQUAL"  
      },  
      {  
        "PropertyPath": "/Timeout",  
        "ExpectedValue": "900",  
        "ActualValue": "22",
```

```
        "DifferenceType": "NOT_EQUAL"
      }
    ],
    "StackResourceDriftStatus": "MODIFIED",
    "Timestamp": "2019-10-02T05:58:47.433Z"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetectStackResourceDrift](#)中的。

## detect-stack-set-drift

以下代码示例演示了如何使用 detect-stack-set-drift。

### AWS CLI

检测堆栈集和所有关联堆栈实例上的偏差

以下detect-stack-set-drift示例在指定的堆栈集（包括与该堆栈集关联的所有堆栈实例）上启动偏差检测操作，并返回一个可用于跟踪偏移操作状态的操作 ID。

```
aws cloudformation detect-stack-set-drift \
  --stack-set-name stack-set-drift-example
```

输出：

```
{
  "OperationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

有关更多信息，请参阅《AWS CloudFormation 用户指南》中的“[检测堆栈集中的非托管配置更改](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetectStackSetDrift](#)中的。

## estimate-template-cost

以下代码示例演示了如何使用 estimate-template-cost。

### AWS CLI

估算模板成本



以下 `estimate-template-cost` 示例为当前文件夹中名为 `template.yaml` 的模板生成了成本估算。

```
aws cloudformation estimate-template-cost \  
  --template-body file://template.yaml
```

输出：

```
{  
  "Url": "http://calculator.s3.amazonaws.com/calc5.html?  
key=cloudformation/7870825a-xmpl-4def-92e7-c4f8dd360cca"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EstimateTemplateCost](#)中的。

## execute-change-set

以下代码示例演示了如何使用 `execute-change-set`。

### AWS CLI

#### 执行更改集

以下 `execute-change-set` 示例执行由更改集名称和堆栈名称指定的更改集。

```
aws cloudformation execute-change-set \  
  --change-set-name my-change-set \  
  --stack-name my-stack
```

以下 `execute-change-set` 示例执行由变更集的完整 ARN 指定的变更集。

```
aws cloudformation execute-change-set \  
  --change-set-name arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-  
change-set/bc9555ba-a949-xmpl-bfb8-f41d04ec5784
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ExecuteChangeSet](#)中的。

## get-stack-policy

以下代码示例演示了如何使用 `get-stack-policy`。

## AWS CLI

### 查看堆栈策略

以下`get-stack-policy`示例显示了指定堆栈的堆栈策略。要将策略附加到堆栈，请使用`set-stack-policy`命令。

```
aws cloudformation get-stack-policy \  
  --stack-name my-stack
```

输出：

```
{  
  "StackPolicyBody": "{\  
    \"Statement\" : [\  
      {\  
        \"Effect\" :  
        \"Allow\",\  
        \"Action\" : \"Update:*\",\  
        \"Principal\" : \"*\",\  
        \"Resource\" : \"*\"\  
      },\  
      {\  
        \"Effect\" : \"Deny\",\  
        \"Action\" : \"Update:*\",\  
        \"Principal\" : \"*\",\  
        \"Resource\" :  
        \"LogicalResourceId/bucket\"\  
      }\  
    ]\  
  }"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetStackPolicy](#)中的。

## get-template-summary

以下代码示例演示了如何使用 `get-template-summary`。

## AWS CLI

### 显示模板摘要

以下命令显示有关指定模板文件的资源和元数据的摘要信息。

```
aws cloudformation get-template-summary \  
  --template-body file://template.yaml
```

输出：

```
{  
  "Parameters": [],  
  "Description": "A VPC and subnets.",  
  "ResourceTypes": [  
    "AWS::EC2::VPC",
```

```

    "AWS::EC2::Subnet",
    "AWS::EC2::Subnet",
    "AWS::EC2::RouteTable",
    "AWS::EC2::VPCEndpoint",
    "AWS::EC2::SubnetRouteTableAssociation",
    "AWS::EC2::SubnetRouteTableAssociation",
    "AWS::EC2::VPCEndpoint"
  ],
  "Version": "2010-09-09"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTemplateSummary](#)中的。

## get-template

以下代码示例演示了如何使用 get-template。

### AWS CLI

查看 AWS CloudFormation 堆栈的模板正文

以下 get-template 命令显示了 myteststack 堆栈的模板：

```
aws cloudformation get-template --stack-name myteststack
```

输出：

```

{
  "TemplateBody": {
    "AWSTemplateFormatVersion": "2010-09-09",
    "Outputs": {
      "BucketName": {
        "Description": "Name of S3 bucket to hold website content",
        "Value": {
          "Ref": "S3Bucket"
        }
      }
    }
  },
  "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample
template showing how to create a publicly accessible S3 bucket. **WARNING** This
template creates an S3 bucket. You will be billed for the AWS resources used if you
create a stack from this template.",

```

```
    "Resources": {
      "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
          "AccessControl": "PublicRead"
        }
      }
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTemplate](#)中的。

## list-change-sets

以下代码示例演示了如何使用 `list-change-sets`。

### AWS CLI

#### 列出变更集

以下`list-change-sets`示例显示了指定堆栈的待处理更改集的列表。

```
aws cloudformation list-change-sets \
  --stack-name my-stack
```

输出：

```
{
  "Summaries": [
    {
      "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
      "StackName": "my-stack",
      "ChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/
my-change-set/70160340-7914-xmpl-bcbf-128a1fa78b5d",
      "ChangeSetName": "my-change-set",
      "ExecutionStatus": "AVAILABLE",
      "Status": "CREATE_COMPLETE",
      "CreationTime": "2019-10-02T05:38:54.297Z"
    }
  ]
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListChangeSets](#)中的。

## list-exports

以下代码示例演示了如何使用 list-exports。

### AWS CLI

#### 列出导出

以下list-exports示例显示了从当前区域堆栈中导出的列表。

```
aws cloudformation list-exports
```

输出：

```
{
  "Exports": [
    {
      "ExportingStackId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/private-vpc/99764070-b56c-xmpl-bee8-062a88d1d800",
      "Name": "private-vpc-subnet-a",
      "Value": "subnet-07b410xmplddcfa03"
    },
    {
      "ExportingStackId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/private-vpc/99764070-b56c-xmpl-bee8-062a88d1d800",
      "Name": "private-vpc-subnet-b",
      "Value": "subnet-075ed3xmplabd2fb1"
    },
    {
      "ExportingStackId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/private-vpc/99764070-b56c-xmpl-bee8-062a88d1d800",
      "Name": "private-vpc-vpcid",
      "Value": "vpc-011d7xmpl1100e9841"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListExports](#)中的。

## list-imports

以下代码示例演示了如何使用 `list-imports`。

### AWS CLI

列出进口商品

以下 `list-imports` 示例列出了导入指定导出的堆栈。要获取可用导出的列表，请使用 `list-exports` 命令。

```
aws cloudformation list-imports \  
  --export-name private-vpc-vpcid
```

输出：

```
{  
  "Imports": [  
    "my-database-stack"  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListImports](#) 中的。

## list-stack-instances

以下代码示例演示了如何使用 `list-stack-instances`。

### AWS CLI

列出堆栈的实例

以下 `list-stack-instances` 示例列出了根据指定堆栈集创建的实例。

```
aws cloudformation list-stack-instances \  
  --stack-set-name enable-config
```

示例输出包括有关由于错误而无法更新的堆栈的详细信息：

```
{  
  "Summaries": [  

```

```

    {
      "StackSetId": "enable-config:296a3360-xmpl-40af-be78-9341e95bf743",
      "Region": "us-west-2",
      "Account": "123456789012",
      "StackId": "arn:aws:cloudformation:ap-northeast-1:123456789012:stack/
StackSet-enable-config-35a6ac50-d9f8-4084-86e4-7da34d5de4c4/a1631cd0-e5fb-xmpl-
b474-0aa20f14f06e",
      "Status": "CURRENT"
    },
    {
      "StackSetId": "enable-config:296a3360-xmpl-40af-be78-9341e95bf743",
      "Region": "us-west-2",
      "Account": "123456789012",
      "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
StackSet-enable-config-e6cac20f-xmpl-46e9-8314-53e0d4591532/eab53680-e5fa-xmpl-
ba14-0a522351f81e",
      "Status": "OUTDATED",
      "StatusReason": "ResourceLogicalId:ConfigDeliveryChannel,
ResourceType:AWS::Config::DeliveryChannel, ResourceStatusReason:Failed to put
delivery channel 'StackSet-enable-config-e6cac20f-xmpl-46e9-8314-53e0d4591532-
ConfigDeliveryChannel-10JWJ7XD59WR0' because the maximum number of delivery
channels: 1 is reached. (Service: AmazonConfig; Status Code: 400; Error Code:
MaxNumberOfDeliveryChannelsExceededException; Request ID: d14b34a0-ef7c-xmpl-
acf8-8a864370ae56)."

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStackInstances](#)中的。

## list-stack-resources

以下代码示例演示了如何使用 `list-stack-resources`。

### AWS CLI

列出堆栈中的资源

以下命令显示了指定堆栈中的资源列表。

```

aws cloudformation list-stack-resources \
  --stack-name my-stack

```

输出：

```
{
  "StackResourceSummaries": [
    {
      "LogicalResourceId": "bucket",
      "PhysicalResourceId": "my-stack-bucket-1vc62xmplgguf",
      "ResourceType": "AWS::S3::Bucket",
      "LastUpdatedTimestamp": "2019-10-02T04:34:11.345Z",
      "ResourceStatus": "CREATE_COMPLETE",
      "DriftInformation": {
        "StackResourceDriftStatus": "IN_SYNC"
      }
    },
    {
      "LogicalResourceId": "function",
      "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
      "ResourceType": "AWS::Lambda::Function",
      "LastUpdatedTimestamp": "2019-10-02T05:34:27.989Z",
      "ResourceStatus": "UPDATE_COMPLETE",
      "DriftInformation": {
        "StackResourceDriftStatus": "IN_SYNC"
      }
    },
    {
      "LogicalResourceId": "functionRole",
      "PhysicalResourceId": "my-functionRole-HIZXMPLE0M9E",
      "ResourceType": "AWS::IAM::Role",
      "LastUpdatedTimestamp": "2019-10-02T04:34:06.350Z",
      "ResourceStatus": "CREATE_COMPLETE",
      "DriftInformation": {
        "StackResourceDriftStatus": "IN_SYNC"
      }
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStackResources](#)中的。

## list-stack-set-operation-results

以下代码示例演示了如何使用 list-stack-set-operation-results。



## AWS CLI

### 列出堆栈集操作结果

以下命令显示对指定堆栈集中的实例执行更新操作的结果。

```
aws cloudformation list-stack-set-operation-results \
  --stack-set-name enable-config \
  --operation-id 35d45ebc-ed88-xmpl-ab59-0197a1fc83a0
```

输出：

```
{
  "Summaries": [
    {
      "Account": "223456789012",
      "Region": "us-west-2",
      "Status": "SUCCEEDED",
      "AccountGateResult": {
        "Status": "SKIPPED",
        "StatusReason": "Function not found: arn:aws:lambda:eu-west-1:223456789012:function:AWSCloudFormationStackSetAccountGate"
      }
    },
    {
      "Account": "223456789012",
      "Region": "ap-south-1",
      "Status": "CANCELLED",
      "StatusReason": "Cancelled since failure tolerance has exceeded"
    }
  ]
}
```

注意：除非您创建账户门禁函数，否则的SKIPPED状态为AccountGateResult成功操作的预期。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStackSetOperationResults](#)中的。

### list-stack-set-operations

以下代码示例演示了如何使用 list-stack-set-operations。

## AWS CLI

### 列出堆栈集操作

以下`list-stack-set-operations`示例显示了对指定堆栈集的最新操作列表。

```
aws cloudformation list-stack-set-operations \  
  --stack-set-name my-stack-set
```

输出：

```
{  
  "Summaries": [  
    {  
      "OperationId": "35d45ebc-ed88-xmpl-ab59-0197a1fc83a0",  
      "Action": "UPDATE",  
      "Status": "SUCCEEDED",  
      "CreationTimestamp": "2019-10-03T16:28:44.377Z",  
      "EndTimestamp": "2019-10-03T16:42:08.607Z"  
    },  
    {  
      "OperationId": "891aa98f-7118-xmpl-00b2-00954d1dd0d6",  
      "Action": "UPDATE",  
      "Status": "FAILED",  
      "CreationTimestamp": "2019-10-03T15:43:53.916Z",  
      "EndTimestamp": "2019-10-03T15:45:58.925Z"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStackSetOperations](#)中的。

## list-stack-sets

以下代码示例演示了如何使用 `list-stack-sets`。

## AWS CLI

### 列出堆栈集

以下`list-stack-sets`示例显示了当前区域和账户中的堆栈集列表。

```
aws cloudformation list-stack-sets
```

输出：

```
{
  "Summaries": [
    {
      "StackSetName": "enable-config",
      "StackSetId": "enable-config:296a3360-xmpl-40af-be78-9341e95bf743",
      "Description": "Enable AWS Config",
      "Status": "ACTIVE"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStackSets](#)中的。

## list-stacks

以下代码示例演示了如何使用 list-stacks。

### AWS CLI

列出 AWS CloudFormation 堆栈

以下 list-stacks 命令显示了具有 CREATE\_COMPLETE 状态的所有堆栈的摘要信息：

```
aws cloudformation list-stacks --stack-status-filter CREATE_COMPLETE
```

输出：

```
[
  {
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
    "TemplateDescription": "AWS CloudFormation Sample Template S3_Bucket: Sample template showing how to create a publicly accessible S3 bucket. **WARNING** This template creates an S3 bucket. You will be billed for the AWS resources used if you create a stack from this template.",
    "StackStatusReason": null,
    "CreationTime": "2013-08-26T03:27:10.190Z",
    "StackName": "myteststack",
  }
]
```

```
    "StackStatus": "CREATE_COMPLETE"
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStacks](#)中的。

## list-type-registrations

以下代码示例演示了如何使用 `list-type-registrations`。

### AWS CLI

列出某一类型的已完成注册

以下`list-type-registrations`示例显示了指定类型的已完成类型注册的列表。

```
aws cloudformation list-type-registrations \
  --type RESOURCE \
  --type-name My::Logs::LogGroup \
  --registration-status-filter COMPLETE
```

输出：

```
{
  "RegistrationTokenList": [
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
  ]
}
```

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》中的使用 CloudFormation 注册表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTypeRegistrations](#)中的。

## list-type-versions

以下代码示例演示了如何使用 `list-type-versions`。

### AWS CLI

列出扩展程序的版本

以下 `list-type-versions` 示例返回有关扩展版本的摘要信息。

```
aws cloudformation list-type-versions \  
  --endpoint https://example.com \  
  --region us-west-2 \  
  --type RESOURCE \  
  --type-name My::Resource::Example \  
  --publisher-id 123456789012
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》中的使用 AWS CloudFormation 注册表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTypeVersions](#) 中的。

## list-types

以下代码示例演示了如何使用 `list-types`。

### AWS CLI

列出账户中的私有资源类型

以下 `list-types` 示例显示当前在当前 AWS 账户中注册的私有资源类型的列表。

```
aws cloudformation list-types
```

输出：

```
{  
  "TypeSummaries": [  
    {  
      "Description": "WordPress blog resource for internal use",  
      "LastUpdated": "2019-12-04T18:28:15.059Z",  
      "TypeName": "My::WordPress::BlogExample",  
      "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/  
My-WordPress-BlogExample",  
      "DefaultVersionId": "00000005",  
      "Type": "RESOURCE"  
    },  
    {  
      "Description": "Customized resource derived from AWS::Logs::LogGroup",
```

```
        "LastUpdated": "2019-12-04T18:28:15.059Z",
        "TypeName": "My::Logs::LogGroup",
        "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/
My-Logs-LogGroup",
        "DefaultVersionId": "00000003",
        "Type": "RESOURCE"
    }
]
}
```

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》](#) 中的 [使用 CloudFormation 注册表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTypes](#) 中的。

## package

以下代码示例演示了如何使用 package。

### AWS CLI

以下命令将 `template.json` 通过将本地项目上传到 S3 存储桶来导出一个名为的模板，`bucket-name` 并将导出的模板写入到 `packaged-template.json`：

```
aws cloudformation package --template-file /path_to_template/template.json --s3-
bucket bucket-name --output-template-file packaged-template.json --use-json
```

- 有关 API 的详细信息，请参阅 [《AWS CLI 命令参考》](#) 中的 [Pack age](#)。

## publish-type

以下代码示例演示了如何使用 publish-type。

### AWS CLI

#### 发布扩展

以下 `publish-type` 示例将指定的扩展作为该区域的公共扩展发布到 CloudFormation 注册表。

```
aws cloudformation publish-type \  
  --region us-west-2 \  
  --type RESOURCE \  
  --type-name Example::Test::1234567890abcdef0
```

输出：

```
{
  "PublicTypeArn": "arn:aws:cloudformation:us-west-2::type/
resource/000q6TfUovXsEMmgKowxDZLLwqr2QUshd2e75c8c/Example-
Test-1234567890abcdef0/1.0.0"
}
```

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》](#) 中的使用 AWS CloudFormation 注册表。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PublishType](#) 中的。

## register-publisher

以下代码示例演示了如何使用 register-publisher。

AWS CLI

注册出版商

以下 register-publisher 示例注册发布者并接受条款和条件参数。

```
aws cloudformation register-publisher \
  --region us-west-2 \
  --accept-terms-and-conditions
```

输出：

```
{
  "PublisherId": "000q6TfUovXsEMmgKowxDZLLwqr2QUshd2e75c8c"
}
```

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》](#) 中的使用 AWS CloudFormation 注册表。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterPublisher](#) 中的。

## register-type

以下代码示例演示了如何使用 register-type。

## AWS CLI

### 注册资源类型

以下register-type示例将指定的资源类型注册为用户账户中的私有资源类型。

```
aws cloudformation register-type \  
  --type-name My::Organization::ResourceName \  
  --schema-handler-package s3://bucket_name/my-organization-resource_name.zip \  
  --type RESOURCE
```

输出：

```
{  
  "RegistrationToken": "f5525280-104e-4d35-bef5-8f1f1example"  
}
```

有关更多信息，请参阅《类型开发CloudFormation 命令行界面用户指南》中的[注册资源提供者](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterType](#)中的。

## set-stack-policy

以下代码示例演示了如何使用 set-stack-policy。

### AWS CLI

#### 应用堆栈策略

以下set-stack-policy示例禁用指定堆栈中指定资源的更新。 stack-policy.json是一个JSON 文档，它定义了允许对堆栈中的资源执行的操作。

```
aws cloudformation set-stack-policy \  
  --stack-name my-stack \  
  --stack-policy-body file://stack-policy.json
```

输出：

```
{  
  "Statement" : [  
    {  
      "Effect" : "Allow",
```



```

    "Action" : "Update:*",
    "Principal": "*",
    "Resource" : "*"
  },
  {
    "Effect" : "Deny",
    "Action" : "Update:*",
    "Principal": "*",
    "Resource" : "LogicalResourceId/bucket"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetStackPolicy](#)中的。

## set-type-configuration

以下代码示例演示了如何使用 set-type-configuration。

### AWS CLI

#### 配置数据

以下set-type-configuration示例指定了给定账户和区域中已注册 CloudFormation 扩展的配置数据。

```

aws cloudformation set-type-configuration \
  --region us-west-2 \
  --type RESOURCE \
  --type-name Example::Test::Type \
  --configuration-alias default \
  --configuration "{\"CredentialKey\": \"testUserCredential\"}"

```

输出：

```

{
  "ConfigurationArn": "arn:aws:cloudformation:us-west-2:123456789012:type-configuration/resource/Example-Test-Type/default"
}

```

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》](#) 中的使用 AWS CloudFormation 注册表。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetTypeConfiguration](#)中的。

## set-type-default-version

以下代码示例演示了如何使用 set-type-default-version。

### AWS CLI

设置类型的默认版本

以下set-type-default-version示例将指定类型的版本设置为该类型的默认版本。

```
aws cloudformation set-type-default-version \  
  --type RESOURCE \  
  --type-name My::Logs::LogGroup \  
  --version-id 00000003
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS CloudFormation 用户指南](#)》中的使用 [CloudFormation 注册表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetTypeDefaultVersion](#)中的。

## signal-resource

以下代码示例演示了如何使用 signal-resource。

### AWS CLI

向资源发出信号

以下signal-resource示例表示满足名为success的堆栈MyWaitCondition中命名的等待条件my-stack。

```
aws cloudformation signal-resource \  
  --stack-name my-stack \  
  --logical-resource-id MyWaitCondition \  
  --unique-id 1234 \  
  --status SUCCESS
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SignalResource](#)中的。

## stop-stack-set-operation

以下代码示例演示了如何使用 stop-stack-set-operation。

### AWS CLI

#### 停止堆栈集操作

以下stop-stack-set-operation示例停止了对指定堆栈集的正在进行的更新操作。

```
aws cloudformation stop-stack-set-operation \  
  --stack-set-name my-stack-set \  
  --operation-id 1261cd27-490b-xmpl-ab42-793a896c69e6
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopStackSetOperation](#)中的。

## test-type

以下代码示例演示了如何使用 test-type。

### AWS CLI

#### 测试扩展

以下test-type示例测试已注册的扩展，以确保其满足在注册 CloudFormation 表中发布的所有必要要求。

```
aws cloudformation test-type \  
  --arn arn:aws:cloudformation:us-west-2:123456789012:type/resource/Sample-Test-  
Resource123/00000001
```

输出：

```
{  
  "TypeVersionArn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/  
Sample-Test-Resource123/00000001"  
}
```

有关更多信息，请参阅 [《AWS CloudFormation 用户指南》](#) 中的使用 AWS CloudFormation 注册表。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TestType](#)中的。

## update-stack-instances

以下代码示例演示了如何使用 update-stack-instances。

### AWS CLI

#### 更新堆栈实例

以下update-stack-instances示例使用最新设置重试两个区域中两个账户的堆栈实例的更新。指定的容错设置可确保在所有账户和地区尝试更新，即使某些堆栈无法更新。

```
aws cloudformation update-stack-instances \  
  --stack-set-name my-stack-set \  
  --accounts 123456789012 567890123456 \  
  --regions us-east-1 us-west-2 \  
  --operation-preferences FailureToleranceCount=3
```

输出：

```
{  
  "OperationId": "103ebdf2-21ea-xmpl-8892-de5e30733132"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateStackInstances](#)中的。

## update-stack-set

以下代码示例演示了如何使用 update-stack-set。

### AWS CLI

#### 更新堆栈集

以下update-stack-set示例向指定堆栈集中的堆栈实例添加一个密钥名称Owner和值为的标签。IT

```
aws cloudformation update-stack-set \  
  --stack-set-name my-stack-set \  
  --use-previous-template \  
  --tags Key=Owner,Value=IT
```

```
--tags Key=Owner,Value=IT
```

输出：

```
{
  "OperationId": "e2b60321-6cab-xmpl-bde7-530c6f47950e"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateStackSet](#)中的。

## update-stack

以下代码示例演示了如何使用 update-stack。

### AWS CLI

更新 AWS CloudFormation 堆栈

以下 update-stack 命令更新了 mystack 堆栈的模板和输入参数：

```
aws cloudformation update-stack --stack-name mystack --
template-url https://s3.amazonaws.com/sample/updated.template --
parameters ParameterKey=KeyPairName,ParameterValue=SampleKeyPair
ParameterKey=SubnetIDs,ParameterValue=SampleSubnetID1\\,SampleSubnetID2
```

以下 update-stack 命令更新了 mystack 堆栈的 SubnetIDs 参数值：如果您没有指定参数值，则将使用模板中指定的默认值：

```
aws cloudformation update-stack --stack-name mystack --
template-url https://s3.amazonaws.com/sample/updated.template
--parameters ParameterKey=KeyPairName,UsePreviousValue=true
ParameterKey=SubnetIDs,ParameterValue=SampleSubnetID1\\,UpdatedSampleSubnetID2
```

以下 update-stack 命令向 mystack 堆栈添加了两个堆栈通知主题：

```
aws cloudformation update-stack --stack-name mystack --use-previous-template --
notification-arns "arn:aws:sns:use-east-1:123456789012:mytopic1" "arn:aws:sns:us-
east-1:123456789012:mytopic2"
```

有关更多信息，请参阅《AWS CloudFormation 用户指南》中的[AWS CloudFormation 堆栈更新](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateStack](#)中的。

## update-termination-protection

以下代码示例演示了如何使用 update-termination-protection。

### AWS CLI

启用终止保护

以下update-termination-protection示例在指定堆栈上启用终止保护。

```
aws cloudformation update-termination-protection \  
  --stack-name my-stack \  
  --enable-termination-protection
```

输出：

```
{  
  "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-stack/  
d0a825a0-e4cd-xmpl-b9fb-061c69e99204"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateTerminationProtection](#)中的。

## validate-template

以下代码示例演示了如何使用 validate-template。

### AWS CLI

验证 AWS CloudFormation 模板

以下 validate-template 命令将验证 sampletemplate.json 模板：

```
aws cloudformation validate-template --template-body file://sampletemplate.json
```

输出：

```
{  
  "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample template  
showing how to create a publicly accessible S3 bucket. **WARNING** This template  
creates an S3 bucket. You will be billed for the AWS resources used if you create a  
stack from this template.",
```

```
"Parameters": [],  
"Capabilities": []  
}
```

有关更多信息，请参阅《AWS CloudFormation 用户指南》中的使用 AWS CloudFormation 模板。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ValidateTemplate](#) 中的。

## CloudFront 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 CloudFront。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-cloud-front-origin-access-identity**

以下代码示例演示了如何使用 `create-cloud-front-origin-access-identity`。

AWS CLI

创建 CloudFront 源访问身份

以下示例通过提供 OAI 配置作为命令行参数来创建 CloudFront 源访问身份 (OAI)：

```
aws cloudfront create-cloud-front-origin-access-identity \  
  --cloud-front-origin-access-identity-config \  
    CallerReference="cli-example",Comment="Example OAI"
```

您可以通过在 JSON 文件中提供 OAI 配置来完成同样的事情，如以下示例所示：

```
aws cloudfront create-cloud-front-origin-access-identity \  
  --cloud-front-origin-access-identity-config file://OAI-config.json
```

该文件OAI-config.json是当前目录中的一个 JSON 文档，其中包含以下内容：

```
{  
  "CallerReference": "cli-example",  
  "Comment": "Example OAI"  
}
```

无论您为 OAI 配置提供命令行参数还是 JSON 文件，输出都是一样的：

```
{  
  "Location": "https://cloudfront.amazonaws.com/2019-03-26/origin-access-identity/  
cloudfront/E74FTE3AEXAMPLE",  
  "ETag": "E2QWRUHEXAMPLE",  
  "CloudFrontOriginAccessIdentity": {  
    "Id": "E74FTE3AEXAMPLE",  
    "S3CanonicalUserId":  
"cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",  
    "CloudFrontOriginAccessIdentityConfig": {  
      "CallerReference": "cli-example",  
      "Comment": "Example OAI"  
    }  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCloudFrontOriginAccessIdentity](#)中的。

## create-distribution-with-tags

以下代码示例演示了如何使用 create-distribution-with-tags。

### AWS CLI

创建带有标签的 CloudFront 分配

以下示例通过在名为的 JSON 文件中提供分发配置和标签来创建带有两个标签的分配dist-config-with-tags.json：

```
aws cloudfront create-distribution-with-tags \  
  --tags "tag-key=tag-value" "tag-key=tag-value" \  
  --distribution-configuration file://dist-config-with-tags.json
```



```
--distribution-config-with-tags file://dist-config-with-tags.json
```

文件 `dist-config-with-tags.json` 是当前文件夹中包含以下内容的 JSON 文档：请注意文件顶部的 `Tags` 对象，其中包含两个标签：

```
Name = ExampleDistributionProject = ExampleProject
```

```
{
  "Tags": {
    "Items": [
      {
        "Key": "Name",
        "Value": "ExampleDistribution"
      },
      {
        "Key": "Project",
        "Value": "ExampleProject"
      }
    ]
  },
  "DistributionConfig": {
    "CallerReference": "cli-example",
    "Aliases": {
      "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
      "Quantity": 1,
      "Items": [
        {
          "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
          "DomainName": "awsexamplebucket.s3.amazonaws.com",
          "OriginPath": "",
          "CustomHeaders": {
            "Quantity": 0
          },
          "S3OriginConfig": {
            "OriginAccessIdentity": ""
          }
        }
      ]
    },
    "OriginGroups": {
```

```
    "Quantity": 0
  },
  "DefaultCacheBehavior": {
    "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",
    "ForwardedValues": {
      "QueryString": false,
      "Cookies": {
        "Forward": "none"
      },
      "Headers": {
        "Quantity": 0
      },
      "QueryStringCacheKeys": {
        "Quantity": 0
      }
    },
    "TrustedSigners": {
      "Enabled": false,
      "Quantity": 0
    },
    "ViewerProtocolPolicy": "allow-all",
    "MinTTL": 0,
    "AllowedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ],
      "CachedMethods": {
        "Quantity": 2,
        "Items": [
          "HEAD",
          "GET"
        ]
      }
    },
    "SmoothStreaming": false,
    "DefaultTTL": 86400,
    "MaxTTL": 31536000,
    "Compress": false,
    "LambdaFunctionAssociations": {
      "Quantity": 0
    },
    "FieldLevelEncryptionId": ""
  }
}
```

```

    },
    "CacheBehaviors": {
      "Quantity": 0
    },
    "CustomErrorResponses": {
      "Quantity": 0
    },
    "Comment": "",
    "Logging": {
      "Enabled": false,
      "IncludeCookies": false,
      "Bucket": "",
      "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": true,
    "ViewerCertificate": {
      "CloudFrontDefaultCertificate": true,
      "MinimumProtocolVersion": "TLSv1",
      "CertificateSource": "cloudfront"
    },
    "Restrictions": {
      "GeoRestriction": {
        "RestrictionType": "none",
        "Quantity": 0
      }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
  }
}

```

输出：

```

{
  "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/EDFDVBD6EXAMPLE",
  "ETag": "E2QWRUHEXAMPLE",
  "Distribution": {
    "Id": "EDFDVBD6EXAMPLE",
    "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
    "Status": "InProgress",

```

```
"LastModifiedTime": "2019-12-04T23:35:41.433Z",
"InProgressInvalidationBatches": 0,
"DomainName": "d1111111abcdef8.cloudfront.net",
"ActiveTrustedSigners": {
  "Enabled": false,
  "Quantity": 0
},
"DistributionConfig": {
  "CallerReference": "cli-example",
  "Aliases": {
    "Quantity": 0
  },
  "DefaultRootObject": "index.html",
  "Origins": {
    "Quantity": 1,
    "Items": [
      {
        "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
        "DomainName": "awsexamplebucket.s3.amazonaws.com",
        "OriginPath": "",
        "CustomHeaders": {
          "Quantity": 0
        },
        "S3OriginConfig": {
          "OriginAccessIdentity": ""
        }
      }
    ]
  },
  "OriginGroups": {
    "Quantity": 0
  },
  "DefaultCacheBehavior": {
    "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",
    "ForwardedValues": {
      "QueryString": false,
      "Cookies": {
        "Forward": "none"
      },
      "Headers": {
        "Quantity": 0
      },
      "QueryStringCacheKeys": {
        "Quantity": 0
      }
    }
  }
}
```

```
    }
  },
  "TrustedSigners": {
    "Enabled": false,
    "Quantity": 0
  },
  "ViewerProtocolPolicy": "allow-all",
  "MinTTL": 0,
  "AllowedMethods": {
    "Quantity": 2,
    "Items": [
      "HEAD",
      "GET"
    ],
    "CachedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ]
    }
  },
  "SmoothStreaming": false,
  "DefaultTTL": 86400,
  "MaxTTL": 31536000,
  "Compress": false,
  "LambdaFunctionAssociations": {
    "Quantity": 0
  },
  "FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
  "Quantity": 0
},
"CustomErrorResponses": {
  "Quantity": 0
},
"Comment": "",
"Logging": {
  "Enabled": false,
  "IncludeCookies": false,
  "Bucket": "",
  "Prefix": ""
},
},
```

```
    "PriceClass": "PriceClass_All",
    "Enabled": true,
    "ViewerCertificate": {
      "CloudFrontDefaultCertificate": true,
      "MinimumProtocolVersion": "TLSv1",
      "CertificateSource": "cloudfront"
    },
    "Restrictions": {
      "GeoRestriction": {
        "RestrictionType": "none",
        "Quantity": 0
      }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDistributionWithTags](#)中的。

## create-distribution

以下代码示例演示了如何使用 create-distribution。

### AWS CLI

#### 创建分 CloudFront 配

以下示例使用命令行参数为名为 awsexamplebucket 的 S3 存储桶创建分配，并将 index.html 指定为默认根对象：

```
aws cloudfront create-distribution \
  --origin-domain-name awsexamplebucket.s3.amazonaws.com \
  --default-root-object index.html
```

您可以在 JSON 文件中提供分配配置，而不必使用命令行参数，如以下示例所示：

```
aws cloudfront create-distribution \
  --distribution-config file://dist-config.json
```

文件 `dist-config.json` 是当前文件夹中包含以下内容的 JSON 文档：

```
{
  "CallerReference": "cli-example",
  "Aliases": {
    "Quantity": 0
  },
  "DefaultRootObject": "index.html",
  "Origins": {
    "Quantity": 1,
    "Items": [
      {
        "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
        "DomainName": "awsexamplebucket.s3.amazonaws.com",
        "OriginPath": "",
        "CustomHeaders": {
          "Quantity": 0
        },
        "S3OriginConfig": {
          "OriginAccessIdentity": ""
        }
      }
    ]
  },
  "OriginGroups": {
    "Quantity": 0
  },
  "DefaultCacheBehavior": {
    "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",
    "ForwardedValues": {
      "QueryString": false,
      "Cookies": {
        "Forward": "none"
      },
      "Headers": {
        "Quantity": 0
      },
      "QueryStringCacheKeys": {
        "Quantity": 0
      }
    },
    "TrustedSigners": {
      "Enabled": false,
      "Quantity": 0
    }
  }
}
```

```
    },
    "ViewerProtocolPolicy": "allow-all",
    "MinTTL": 0,
    "AllowedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ],
    },
    "CachedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ]
    }
  },
  "SmoothStreaming": false,
  "DefaultTTL": 86400,
  "MaxTTL": 31536000,
  "Compress": false,
  "LambdaFunctionAssociations": {
    "Quantity": 0
  },
  "FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
  "Quantity": 0
},
"CustomErrorResponses": {
  "Quantity": 0
},
"Comment": "",
"Logging": {
  "Enabled": false,
  "IncludeCookies": false,
  "Bucket": "",
  "Prefix": ""
},
"PriceClass": "PriceClass_All",
"Enabled": true,
"ViewerCertificate": {
  "CloudFrontDefaultCertificate": true,
  "MinimumProtocolVersion": "TLSv1",
```



```

    "CertificateSource": "cloudfront"
  },
  "Restrictions": {
    "GeoRestriction": {
      "RestrictionType": "none",
      "Quantity": 0
    }
  },
  "WebACLId": "",
  "HttpVersion": "http2",
  "IsIPV6Enabled": true
}

```

无论您使用命令行参数还是 JSON 文件提供分配信息，输出都相同：

```

{
  "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/EMLARXS9EXAMPLE",
  "ETag": "E9LHASXEXAMPLE",
  "Distribution": {
    "Id": "EMLARXS9EXAMPLE",
    "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
    "Status": "InProgress",
    "LastModifiedTime": "2019-11-22T00:55:15.705Z",
    "InProgressInvalidationBatches": 0,
    "DomainName": "d111111abcdef8.cloudfront.net",
    "ActiveTrustedSigners": {
      "Enabled": false,
      "Quantity": 0
    }
  },
  "DistributionConfig": {
    "CallerReference": "cli-example",
    "Aliases": {
      "Quantity": 0
    }
  },
  "DefaultRootObject": "index.html",
  "Origins": {
    "Quantity": 1,
    "Items": [
      {
        "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
        "DomainName": "awsexamplebucket.s3.amazonaws.com",
        "OriginPath": "",

```

```
        "CustomHeaders": {
            "Quantity": 0
        },
        "S3OriginConfig": {
            "OriginAccessIdentity": ""
        }
    }
]
},
"OriginGroups": {
    "Quantity": 0
},
"DefaultCacheBehavior": {
    "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",
    "ForwardedValues": {
        "QueryString": false,
        "Cookies": {
            "Forward": "none"
        },
        "Headers": {
            "Quantity": 0
        },
    },
    "QueryStringCacheKeys": {
        "Quantity": 0
    }
},
"TrustedSigners": {
    "Enabled": false,
    "Quantity": 0
},
"ViewerProtocolPolicy": "allow-all",
"MinTTL": 0,
"AllowedMethods": {
    "Quantity": 2,
    "Items": [
        "HEAD",
        "GET"
    ],
    "CachedMethods": {
        "Quantity": 2,
        "Items": [
            "HEAD",
            "GET"
        ]
    }
]
```

```
    }
    },
    "SmoothStreaming": false,
    "DefaultTTL": 86400,
    "MaxTTL": 31536000,
    "Compress": false,
    "LambdaFunctionAssociations": {
      "Quantity": 0
    },
    "FieldLevelEncryptionId": ""
  },
  "CacheBehaviors": {
    "Quantity": 0
  },
  "CustomErrorResponses": {
    "Quantity": 0
  },
  "Comment": "",
  "Logging": {
    "Enabled": false,
    "IncludeCookies": false,
    "Bucket": "",
    "Prefix": ""
  },
  "PriceClass": "PriceClass_All",
  "Enabled": true,
  "ViewerCertificate": {
    "CloudFrontDefaultCertificate": true,
    "MinimumProtocolVersion": "TLSv1",
    "CertificateSource": "cloudfront"
  },
  "Restrictions": {
    "GeoRestriction": {
      "RestrictionType": "none",
      "Quantity": 0
    }
  },
  "WebACLId": "",
  "HttpVersion": "http2",
  "IsIPV6Enabled": true
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDistribution](#)中的。

## create-field-level-encryption-config

以下代码示例演示了如何使用 create-field-level-encryption-config。

### AWS CLI

#### 创建 CloudFront 字段级加密配置

以下示例通过在名为的 JSON 文件中提供配置参数来创建字段级加密配置。fle-config.json在创建字段级加密配置之前，必须具有字段级加密配置文件。要创建配置文件，请参阅 create-field-level-encryption-profile 命令。

有关 CloudFront 字段级加密的更多信息，请参阅 Amazon 开发者指南[中的使用字段级加密来帮助保护敏感数据](#)。CloudFront

```
aws cloudfront create-field-level-encryption-config \  
  --field-level-encryption-config file://fle-config.json
```

文件 fle-config.json 是当前文件夹中包含以下内容的 JSON 文档：

```
{  
  "CallerReference": "cli-example",  
  "Comment": "Example FLE configuration",  
  "QueryArgProfileConfig": {  
    "ForwardWhenQueryArgProfileIsUnknown": true,  
    "QueryArgProfiles": {  
      "Quantity": 0  
    }  
  },  
  "ContentTypeProfileConfig": {  
    "ForwardWhenContentTypeIsUnknown": true,  
    "ContentTypeProfiles": {  
      "Quantity": 1,  
      "Items": [  
        {  
          "Format": "URLEncoded",  
          "ProfileId": "P280MFCLSY0CVU",  
          "ContentType": "application/x-www-form-urlencoded"  
        }  
      ]  
    }  
  }  
}
```

```

    ]
  }
}

```

输出：

```

{
  "Location": "https://cloudfront.amazonaws.com/2019-03-26/field-level-encryption/
C3KM2WVD605UAY",
  "ETag": "E2P4Z4VU7TY5SG",
  "FieldLevelEncryption": {
    "Id": "C3KM2WVD605UAY",
    "LastModifiedTime": "2019-12-10T21:30:18.974Z",
    "FieldLevelEncryptionConfig": {
      "CallerReference": "cli-example",
      "Comment": "Example FLE configuration",
      "QueryArgProfileConfig": {
        "ForwardWhenQueryArgProfileIsUnknown": true,
        "QueryArgProfiles": {
          "Quantity": 0,
          "Items": []
        }
      },
      "ContentTypeProfileConfig": {
        "ForwardWhenContentTypeIsUnknown": true,
        "ContentTypeProfiles": {
          "Quantity": 1,
          "Items": [
            {
              "Format": "URLEncoded",
              "ProfileId": "P280MFCLSYOCVU",
              "ContentType": "application/x-www-form-urlencoded"
            }
          ]
        }
      }
    }
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFieldLevelEncryptionConfig](#)中的。

## create-field-level-encryption-profile

以下代码示例演示了如何使用 create-field-level-encryption-profile。

### AWS CLI

创建 CloudFront 字段级加密配置文件

以下示例通过在名为的 JSON 文件中提供参数来创建字段级加密配置文件。fle-profile-config.json在创建字段级加密配置文件之前，您必须拥有 CloudFront 公钥。要创建 CloudFront 公钥，请参阅 create-public-key 命令。

有关 CloudFront 字段级加密的更多信息，请参阅 Amazon 开发者指南[中的使用字段级加密来帮助保护敏感数据](#)。CloudFront

```
aws cloudfront create-field-level-encryption-profile \  
  --field-level-encryption-profile-config file://fle-profile-config.json
```

文件 fle-profile-config.json 是当前文件夹中包含以下内容的 JSON 文档：

```
{  
  "Name": "ExampleFLEProfile",  
  "CallerReference": "cli-example",  
  "Comment": "FLE profile for AWS CLI example",  
  "EncryptionEntities": [  
    {  
      "Quantity": 1,  
      "Items": [  
        {  
          "PublicKeyId": "K2K8NC4HVFE3M0",  
          "ProviderId": "ExampleFLEProvider",  
          "FieldPatterns": {  
            "Quantity": 1,  
            "Items": [  
              "ExampleSensitiveField"  
            ]  
          }  
        }  
      ]  
    }  
  ]  
}
```

输出：

```
{
  "Location": "https://cloudfront.amazonaws.com/2019-03-26/field-level-encryption-
profile/PPK0U0SIF5WSV",
  "ETag": "E2QWRUHEXAMPLE",
  "FieldLevelEncryptionProfile": {
    "Id": "PPK0U0SIF5WSV",
    "LastModifiedTime": "2019-12-10T01:03:16.537Z",
    "FieldLevelEncryptionProfileConfig": {
      "Name": "ExampleFLEProfile",
      "CallerReference": "cli-example",
      "Comment": "FLE profile for AWS CLI example",
      "EncryptionEntities": {
        "Quantity": 1,
        "Items": [
          {
            "PublicKeyId": "K2K8NC4HVFE3M0",
            "ProviderId": "ExampleFLEProvider",
            "FieldPatterns": {
              "Quantity": 1,
              "Items": [
                "ExampleSensitiveField"
              ]
            }
          ]
        ]
      }
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFieldLevelEncryptionProfile](#)中的。

## create-invalidation

以下代码示例演示了如何使用 create-invalidation。

### AWS CLI

为分配创建失效状态 CloudFront

以下create-invalidation示例为指定 CloudFront 发行版中的指定文件创建失效：

```
aws cloudfront create-invalidation \
```

```
--distribution-id EDFDVBD6EXAMPLE \  
--paths "/example-path/example-file.jpg" "/example-path/example-file2.png"
```

输出：

```
{  
  "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/  
EDFDVBD6EXAMPLE/invalidation/I1JLWSDAP8FU89",  
  "Invalidation": {  
    "Id": "I1JLWSDAP8FU89",  
    "Status": "InProgress",  
    "CreateTime": "2019-12-05T18:24:51.407Z",  
    "InvalidationBatch": {  
      "Paths": {  
        "Quantity": 2,  
        "Items": [  
          "/example-path/example-file2.png",  
          "/example-path/example-file.jpg"  
        ]  
      },  
      "CallerReference": "cli-1575570291-670203"  
    }  
  }  
}
```

在前面的示例中，AWS CLI 自动生成了一个随机值 CallerReference。要指定自己的 CallerReference，或者为了避免将失效参数作为命令行参数传递，可以使用 JSON 文件。以下示例通过在名为 inv-batch.json 的 JSON 文件中提供失效参数，从而为两个文件创建失效：

```
aws cloudfront create-invalidation \  
--distribution-id EDFDVBD6EXAMPLE \  
--invalidation-batch file://inv-batch.json
```

inv-batch.json 的内容：

```
{  
  "Paths": {  
    "Quantity": 2,  
    "Items": [  
      "/example-path/example-file.jpg",  
      "/example-path/example-file2.png"  
    ]  
  }  
}
```



```
  },
  "CallerReference": "cli-example"
}
```

输出：

```
{
  "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/EDFDVBD6EXAMPLE/invalidation/I2J0I21PCUY0IK",
  "Invalidation": {
    "Id": "I2J0I21PCUY0IK",
    "Status": "InProgress",
    "CreateTime": "2019-12-05T18:40:49.413Z",
    "InvalidationBatch": {
      "Paths": {
        "Quantity": 2,
        "Items": [
          "/example-path/example-file.jpg",
          "/example-path/example-file2.png"
        ]
      }
    },
    "CallerReference": "cli-example"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateInvalidation](#)中的。

## create-public-key

以下代码示例演示了如何使用 create-public-key。

### AWS CLI

#### 创建 CloudFront 公钥

以下示例通过在名为的 JSON 文件中提供参数来创建 CloudFront 公钥pub-key-config.json。必须先拥有 PEM 编码的公有密钥，然后才能使用此命令。有关更多信息，请参阅《Amazon CloudFront 开发者指南》中的[创建 RSA 密钥对](#)。

```
aws cloudfront create-public-key \
  --public-key-config file://pub-key-config.json
```

文件 `pub-key-config.json` 是当前文件夹中包含以下内容的 JSON 文档：请注意，公有密钥以 PEM 格式编码。

```
{
  "CallerReference": "cli-example",
  "Name": "ExampleKey",
  "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAA0CAQ8AMIIBCgKCAQEAxPmbCA2Ks0lnd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1U0WcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAz03ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95y1UQd9LfYTBzVZYG2tSesplc0kjM3\n2Uu
+oMwxQAw1NINnSLPinMVsutJy6Zq1V3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nq
+kGZ2NQ0FyIyT2eiLK0X5RgB/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nrWIDAQAB\n-----END
PUBLIC KEY-----\n",
  "Comment": "example public key"
}
```

输出：

```
{
  "Location": "https://cloudfront.amazonaws.com/2019-03-26/public-key/
KDFB19YGCR002",
  "ETag": "E2QWRUHEXAMPLE",
  "PublicKey": {
    "Id": "KDFB19YGCR002",
    "CreatedTime": "2019-12-05T18:51:43.781Z",
    "PublicKeyConfig": {
      "CallerReference": "cli-example",
      "Name": "ExampleKey",
      "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAA0CAQ8AMIIBCgKCAQEAxPmbCA2Ks0lnd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1U0WcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAz03ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95y1UQd9LfYTBzVZYG2tSesplc0kjM3\n2Uu
+oMwxQAw1NINnSLPinMVsutJy6Zq1V3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nq
+kGZ2NQ0FyIyT2eiLK0X5RgB/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nrWIDAQAB\n-----END
PUBLIC KEY-----\n",
      "Comment": "example public key"
    }
  }
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreatePublicKey](#) 中的。

## delete-cloud-front-origin-access-identity

以下代码示例演示了如何使用 delete-cloud-front-origin-access-identity。

### AWS CLI

#### 删除 CloudFront 源访问身份

以下示例删除带有 ID E74FTE3AEXAMPLE 的源访问身份 (OAI)。要删除 OAI，您必须拥有 OAI 的 ID 和 ETag。ETag 将在 `delete-cloud-front-origin-access-identity` 和 `create-cloud-front-origin-access-identitions` 命令的输出中返回。list-cloud-front-origin 要获取 ETag，请使用 `get-cloud-front-origin-access-identity` 或 `get-cloud-front-origin-identity` 命令。access-identity-config 使用 `--if-match` 选项提供 OAI。ETag

```
aws cloudfront delete-cloud-front-origin-access-identity \
  --id E74FTE3AEXAMPLE \
  --if-match E2QWRUHEXAMPLE
```

成功时，此命令没有输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteCloudFrontOriginAccessIdentity](#) 中的。

## delete-distribution

以下代码示例演示了如何使用 delete-distribution。

### AWS CLI

#### 删除 CloudFront 分配

以下示例删除标识为 ID 的 CloudFront 分配 EDFDVBD6EXAMPLE。删除分配之前，必须先禁用它。要禁用分配，请使用 `update-distribution` 命令。有关更多信息，请参阅 `update-distribution` 示例。

分配已禁用，您可以将其删除。要删除分配，您必须使用 `--if-match` 选项来提供分配的 ETag。要获取 ETag，请使用 `get-distribution` 或 `get-distribution-config` 命令。

```
aws cloudfront delete-distribution \
  --id EDFDVBD6EXAMPLE \
  --if-match E2QWRUHEXAMPLE
```

成功时，此命令没有输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteDistribution](#) 中的。

## delete-field-level-encryption-config

以下代码示例演示了如何使用 delete-field-level-encryption-config。

### AWS CLI

#### 删除 CloudFront 字段级加密配置

以下示例删除带有 CloudFront ID 的字段级加密配置。C3KM2WVD605UAY 要删除字段级加密配置，必须拥有其 ID 和 ETag。该 ID 将在 create-field-level-encryption 和 list-field-level-encryption-configs 命令的输出中返回。要获取 ETag，请使用 get-field-level-encryption 或 get-field-level-encryption-config 命令。使用 --if-match 选项提供配置 ETag。

```
aws cloudfront delete-field-level-encryption-config \  
  --id C3KM2WVD605UAY \  
  --if-match E26M4BIAV81ZF6
```

成功时，此命令没有输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteFieldLevelEncryptionConfig](#) 中的。

## delete-field-level-encryption-profile

以下代码示例演示了如何使用 delete-field-level-encryption-profile。

### AWS CLI

#### 删除 CloudFront 字段级加密配置文件

以下示例删除带有 CloudFront ID 的字段级加密配置文件。PPK0UOSIF5WSV 要删除字段级加密配置文件，您必须拥有其 ID 和 ETag。该 ID 将在 profile 和 create-field-level-encryption list-field-level-encryption-profiles 命令的输出中返回。要获取 ETag，请使用 profile 或 get-field-level-encryption-profile 或 get-field-level-encryption-profile-config 命令。使用 --if-match 选项提供个人资料 ETag。

```
aws cloudfront delete-field-level-encryption-profile \  
  --id PPK0UOSIF5WSV \  
  --if-match EJETYFJ9CL66D
```

成功时，此命令没有输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteFieldLevelEncryptionProfile](#) 中的。

## delete-public-key

以下代码示例演示了如何使用 delete-public-key。

### AWS CLI

#### 删除 CloudFront 公钥

以下示例删除 ID 为的 CloudFront 公钥KDFB19YGCR002。要删除公钥，您必须拥有其 ID 和ETag。ID 将在create-public-key 和list-public-keys 命令的输出中返回。要获取ETag，请使用 get-public-key 或get-public-key-config 命令。使用--if-match选项提供公钥ETag。

```
aws cloudfront delete-public-key \  
  --id KDFB19YGCR002 \  
  --if-match E2QWRUHEXAMPLE
```

成功时，此命令没有输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePublicKey](#)中的。

## get-cloud-front-origin-access-identity-config

以下代码示例演示了如何使用 get-cloud-front-origin-access-identity-config。

### AWS CLI

#### 获取 CloudFront 源站访问身份配置

以下示例获取有关带有 ID 的 CloudFront 源访问身份 (OAI) 的元数据E74FTE3AEXAMPLE，包括其ETag。OAI ID 将在-access-identity 和 create-cloud-front-origin-access-identitions 命令的输出中返回。list-cloud-front-origin

```
aws cloudfront get-cloud-front-origin-access-identity-config --id E74FTE3AEXAMPLE
```

输出：

```
{  
  "ETag": "E2QWRUHEXAMPLE",  
  "CloudFrontOriginAccessIdentityConfig": {  
    "CallerReference": "cli-example",  
    "Comment": "Example OAI"  }  
}
```

```
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCloudFrontOriginAccessIdentityConfig](#)中的。

## get-cloud-front-origin-access-identity

以下代码示例演示了如何使用 `get-cloud-front-origin-access-identity`。

### AWS CLI

获取 CloudFront 源站访问身份

以下示例获取带有 ID 的 CloudFront 源访问身份 (OAI)E74FTE3AEXAMPLE，包括其ETag和关联的 S3 规范 ID。OAI ID 将在 `-access-identity` 和 `create-cloud-front-origin-access-identitions` 命令的输出中返回。`list-cloud-front-origin`

```
aws cloudfront get-cloud-front-origin-access-identity --id E74FTE3AEXAMPLE
```

输出：

```
{
  "ETag": "E2QWRUHEXAMPLE",
  "CloudFrontOriginAccessIdentity": {
    "Id": "E74FTE3AEXAMPLE",
    "S3CanonicalUserId":
"cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",
    "CloudFrontOriginAccessIdentityConfig": {
      "CallerReference": "cli-example",
      "Comment": "Example OAI"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCloudFrontOriginAccessIdentity](#)中的。

## get-distribution-config

以下代码示例演示了如何使用 `get-distribution-config`。

## AWS CLI

### 获取分 CloudFront 发配置

以下示例获取有关带有 ID 的 CloudFront 分配的元数据EDFDVBD6EXAMPLE，包括其ETag。分配 ID 将在 create-distribution 和 list-distributions 命令中返回。

```
aws cloudfront get-distribution-config --id EDFDVBD6EXAMPLE
```

输出：

```
{
  "ETag": "E2QWRUHEXAMPLE",
  "DistributionConfig": {
    "CallerReference": "cli-example",
    "Aliases": {
      "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
      "Quantity": 1,
      "Items": [
        {
          "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
          "DomainName": "awsexamplebucket.s3.amazonaws.com",
          "OriginPath": "",
          "CustomHeaders": {
            "Quantity": 0
          },
          "S3OriginConfig": {
            "OriginAccessIdentity": ""
          }
        }
      ]
    },
    "OriginGroups": {
      "Quantity": 0
    },
    "DefaultCacheBehavior": {
      "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",
      "ForwardedValues": {
        "QueryString": false,
        "Cookies": {
```

```
        "Forward": "none"
      },
      "Headers": {
        "Quantity": 0
      },
      "QueryStringCacheKeys": {
        "Quantity": 0
      }
    },
    "TrustedSigners": {
      "Enabled": false,
      "Quantity": 0
    },
    "ViewerProtocolPolicy": "allow-all",
    "MinTTL": 0,
    "AllowedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ],
    },
    "CachedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ]
    }
  },
  "SmoothStreaming": false,
  "DefaultTTL": 86400,
  "MaxTTL": 31536000,
  "Compress": false,
  "LambdaFunctionAssociations": {
    "Quantity": 0
  },
  "FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
  "Quantity": 0
},
"CustomErrorResponses": {
  "Quantity": 0
},
},
```



```

    "Comment": "",
    "Logging": {
      "Enabled": false,
      "IncludeCookies": false,
      "Bucket": "",
      "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": true,
    "ViewerCertificate": {
      "CloudFrontDefaultCertificate": true,
      "MinimumProtocolVersion": "TLSv1",
      "CertificateSource": "cloudfront"
    },
    "Restrictions": {
      "GeoRestriction": {
        "RestrictionType": "none",
        "Quantity": 0
      }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDistributionConfig](#)中的。

## get-distribution

以下代码示例演示了如何使用 get-distribution。

### AWS CLI

要获得分 CloudFront 发

以下示例获取带有 ID 的 CloudFront 分配EDFDVBD6EXAMPLE，包括其ETag。分配 ID 将在 create-distribution 和 list-distributions 命令中返回。

```
aws cloudfront get-distribution --id EDFDVBD6EXAMPLE
```

输出：

```
{
  "ETag": "E2QWRUHEXAMPLE",
  "Distribution": {
    "Id": "EDFDVBD6EXAMPLE",
    "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
    "Status": "Deployed",
    "LastModifiedTime": "2019-12-04T23:35:41.433Z",
    "InProgressInvalidationBatches": 0,
    "DomainName": "d111111abcdef8.cloudfront.net",
    "ActiveTrustedSigners": {
      "Enabled": false,
      "Quantity": 0
    },
    "DistributionConfig": {
      "CallerReference": "cli-example",
      "Aliases": {
        "Quantity": 0
      },
      "DefaultRootObject": "index.html",
      "Origins": {
        "Quantity": 1,
        "Items": [
          {
            "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
            "DomainName": "awsexamplebucket.s3.amazonaws.com",
            "OriginPath": "",
            "CustomHeaders": {
              "Quantity": 0
            },
            "S3OriginConfig": {
              "OriginAccessIdentity": ""
            }
          }
        ]
      },
      "OriginGroups": {
        "Quantity": 0
      },
      "DefaultCacheBehavior": {
        "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",
        "ForwardedValues": {
          "QueryString": false,
          "Cookies": {
```

```
        "Forward": "none"
    },
    "Headers": {
        "Quantity": 0
    },
    "QueryStringCacheKeys": {
        "Quantity": 0
    }
},
"TrustedSigners": {
    "Enabled": false,
    "Quantity": 0
},
"ViewerProtocolPolicy": "allow-all",
"MinTTL": 0,
"AllowedMethods": {
    "Quantity": 2,
    "Items": [
        "HEAD",
        "GET"
    ],
},
"CachedMethods": {
    "Quantity": 2,
    "Items": [
        "HEAD",
        "GET"
    ]
}
},
"SmoothStreaming": false,
"DefaultTTL": 86400,
"MaxTTL": 31536000,
"Compress": false,
"LambdaFunctionAssociations": {
    "Quantity": 0
},
"FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
    "Quantity": 0
},
"CustomErrorResponses": {
    "Quantity": 0
},
}
```

```

    "Comment": "",
    "Logging": {
      "Enabled": false,
      "IncludeCookies": false,
      "Bucket": "",
      "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": true,
    "ViewerCertificate": {
      "CloudFrontDefaultCertificate": true,
      "MinimumProtocolVersion": "TLSv1",
      "CertificateSource": "cloudfront"
    },
    "Restrictions": {
      "GeoRestriction": {
        "RestrictionType": "none",
        "Quantity": 0
      }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
  }
}
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDistribution](#)中的。

## get-field-level-encryption-config

以下代码示例演示了如何使用 `get-field-level-encryption-config`。

### AWS CLI

获取有关 CloudFront 字段级加密配置的元数据

以下示例获取有关带有 ID 的 CloudFront 字段级加密配置的元数据 `C3KM2WVD605UAY`，包括其 `ETag`

```
aws cloudfront get-field-level-encryption-config --id C3KM2WVD605UAY
```

输出：

```
{
  "ETag": "E2P4Z4VU7TY5SG",
  "FieldLevelEncryptionConfig": {
    "CallerReference": "cli-example",
    "Comment": "Example FLE configuration",
    "QueryArgProfileConfig": {
      "ForwardWhenQueryArgProfileIsUnknown": true,
      "QueryArgProfiles": {
        "Quantity": 0,
        "Items": []
      }
    },
    "ContentTypeProfileConfig": {
      "ForwardWhenContentTypeIsUnknown": true,
      "ContentTypeProfiles": {
        "Quantity": 1,
        "Items": [
          {
            "Format": "URLEncoded",
            "ProfileId": "P280MFCLSYOCVU",
            "ContentType": "application/x-www-form-urlencoded"
          }
        ]
      }
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFieldLevelEncryptionConfig](#)中的。

## get-field-level-encryption-profile-config

以下代码示例演示了如何使用 `get-field-level-encryption-profile-config`。

### AWS CLI

获取 CloudFront 字段级加密配置文件配置

以下示例获取有关带有 ID 的 CloudFront 字段级加密配置文件的元数据 PPK0U0SIF5WSV，包括其 ETag

```
aws cloudfront get-field-level-encryption-profile-config --id PPK0U0SIF5WSV
```

输出：

```
{
  "ETag": "E1QQG65FS2L2GC",
  "FieldLevelEncryptionProfileConfig": {
    "Name": "ExampleFLEProfile",
    "CallerReference": "cli-example",
    "Comment": "FLE profile for AWS CLI example",
    "EncryptionEntities": {
      "Quantity": 1,
      "Items": [
        {
          "PublicKeyId": "K2K8NC4HVFE3M0",
          "ProviderId": "ExampleFLEProvider",
          "FieldPatterns": {
            "Quantity": 1,
            "Items": [
              "ExampleSensitiveField"
            ]
          }
        }
      ]
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFieldLevelEncryptionProfileConfig](#)中的。

## get-field-level-encryption-profile

以下代码示例演示了如何使用 `get-field-level-encryption-profile`。

AWS CLI

获取 CloudFront 字段级加密配置文件

以下示例获取带有 ID 的 CloudFront 字段级加密配置文件 PPK0U0SIF5WSV，包括其 ETag

```
aws cloudfront get-field-level-encryption-profile --id PPK0U0SIF5WSV
```

输出：

```
{
  "ETag": "E1QQG65FS2L2GC",
  "FieldLevelEncryptionProfile": {
    "Id": "PPK0UOSIF5WSV",
    "LastModifiedTime": "2019-12-10T01:03:16.537Z",
    "FieldLevelEncryptionProfileConfig": {
      "Name": "ExampleFLEProfile",
      "CallerReference": "cli-example",
      "Comment": "FLE profile for AWS CLI example",
      "EncryptionEntities": {
        "Quantity": 1,
        "Items": [
          {
            "PublicKeyId": "K2K8NC4HVFE3M0",
            "ProviderId": "ExampleFLEProvider",
            "FieldPatterns": {
              "Quantity": 1,
              "Items": [
                "ExampleSensitiveField"
              ]
            }
          ]
        ]
      }
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFieldLevelEncryptionProfile](#)中的。

## get-field-level-encryption

以下代码示例演示了如何使用 get-field-level-encryption。

### AWS CLI

获取 CloudFront 字段级加密配置

以下示例获取带有 ID 的 CloudFront 字段级加密配置C3KM2WVD605UAY，包括其 ETag

```
aws cloudfront get-field-level-encryption --id C3KM2WVD605UAY
```

输出：

```
{
  "ETag": "E2P4Z4VU7TY5SG",
  "FieldLevelEncryption": {
    "Id": "C3KM2WVD605UAY",
    "LastModifiedTime": "2019-12-10T21:30:18.974Z",
    "FieldLevelEncryptionConfig": {
      "CallerReference": "cli-example",
      "Comment": "Example FLE configuration",
      "QueryArgProfileConfig": {
        "ForwardWhenQueryArgProfileIsUnknown": true,
        "QueryArgProfiles": {
          "Quantity": 0,
          "Items": []
        }
      },
      "ContentTypeProfileConfig": {
        "ForwardWhenContentTypeIsUnknown": true,
        "ContentTypeProfiles": {
          "Quantity": 1,
          "Items": [
            {
              "Format": "URLEncoded",
              "ProfileId": "P280MFCLSYOCVU",
              "ContentType": "application/x-www-form-urlencoded"
            }
          ]
        }
      }
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFieldLevelEncryption](#)中的。

## get-invalidations

以下代码示例演示了如何使用 get-invalidations。

### AWS CLI

要获得无 CloudFront 效



以下示例使用带有 ID 的 CloudFront 分配的 ID I2J0I21PCUY0IK 获取失效信息：EDFDVBD6EXAMPLE

```
aws cloudfront get-invalidation --id I2J0I21PCUY0IK --distribution-id
EDFDVBD6EXAMPLE
```

输出：

```
{
  "Invalidation": {
    "Status": "Completed",
    "InvalidationBatch": {
      "Paths": {
        "Items": [
          "/example-path/example-file.jpg",
          "/example-path/example-file-2.jpg"
        ],
        "Quantity": 2
      },
      "CallerReference": "cli-example"
    },
    "Id": "I2J0I21PCUY0IK",
    "CreateTime": "2019-12-05T18:40:49.413Z"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInvalidation](#)中的。

## get-public-key-config

以下代码示例演示了如何使用 get-public-key-config。

### AWS CLI

获取 CloudFront 公钥配置

以下示例获取有关带有 ID 的 CloudFront 公钥的元数据KDFB19YGCR002，包括其ETag。公钥 ID 将在create-public-key 和 list-public-keys 命令中返回。

```
aws cloudfront get-public-key-config --id KDFB19YGCR002
```

输出：

```
{
  "ETag": "E2QWRUHEXAMPLE",
  "PublicKeyConfig": {
    "CallerReference": "cli-example",
    "Name": "ExampleKey",
    "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAA0CAQ8AMIIBCgKCAQEAxPmCA2Ks01nd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1U0WcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBaz03ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95y1UQd9LfYTBzVZYG2tSesplc0kjM3\n2Uu
+oMwxQAw1NINnSLPinMVsutJy6Zq1V3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nnq
+kGZ2NQ0FyIyT2eiLK0X5RgB/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nnrwIDAQAB\n-----END
PUBLIC KEY-----\n",
    "Comment": "example public key"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPublicKeyConfig](#)中的。

## get-public-key

以下代码示例演示了如何使用 get-public-key。

### AWS CLI

#### 获取 CloudFront 公钥

以下示例获取带有 ID 的 CloudFront 公钥KDFB19YGCR002，包括其ETag。公钥 ID 将在create-public-key 和 list-public-keys 命令中返回。

```
aws cloudfront get-public-key --id KDFB19YGCR002
```

输出：

```
{
  "ETag": "E2QWRUHEXAMPLE",
  "PublicKey": {
    "Id": "KDFB19YGCR002",
    "CreatedTime": "2019-12-05T18:51:43.781Z",
    "PublicKeyConfig": {
      "CallerReference": "cli-example",
      "Name": "ExampleKey",
```

```

      "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAA0CAQ8AMIIBCgKCAQEAxPmBCA2Ks0lnd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1U0WcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAz03ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95y1UQd9LfYTBzVZYG2tSesplc0kjM3\n2Uu
+oMwXQAw1NINnSLPinMVsutJy6Zq1V3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nq
+kGZ2NQ0FyIyT2eiLK0X5RgB/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nnrwIDAQAB\n-----END
PUBLIC KEY-----\n",
      "Comment": "example public key"
    }
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPublicKey](#)中的。

## list-cloud-front-origin-access-identities

以下代码示例演示了如何使用 `list-cloud-front-origin-access-identities`。

### AWS CLI

列出 CloudFront 源访问身份

以下示例获取了您 AWS 账户中的 CloudFront 源站访问身份 (OAI) 列表：

```
aws cloudfront list-cloud-front-origin-access-identities
```

输出：

```

{
  "CloudFrontOriginAccessIdentityList": {
    "Items": [
      {
        "Id": "E74FTE3AEXAMPLE",
        "S3CanonicalUserId":
"cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",
        "Comment": "Example OAI"
      },
      {
        "Id": "EH1HDMBEXAMPLE",
        "S3CanonicalUserId":
"1489f6f2e6faacaae7ff64c4c3e6956c24f78788abfc1718c3527c263bf7a17EXAMPLE",
        "Comment": "Test OAI"
      }
    ]
  }
}

```

```
    },
    {
      "Id": "E2X2C9EXAMPLE",
      "S3CanonicalUserId":
"cbfeebb915a64749f9be546a45b3fcfd3a31c779673c13c4dd460911ae402c2EXAMPLE",
      "Comment": "Example OAI #2"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCloudFrontOriginAccessIdentities](#)中的。

## list-distributions

以下代码示例演示了如何使用 list-distributions。

### AWS CLI

列出 CloudFront 发行版

以下示例获取了您 AWS 账户中的 CloudFront 分配列表：

```
aws cloudfront list-distributions
```

输出：

```
{
  "DistributionList": {
    "Items": [
      {
        "Id": "EMLARXS9EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/
EMLARXS9EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-11-22T00:55:15.705Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
          "Enabled": false,
          "Quantity": 0
        },
        "DistributionConfig": {
```

```
"CallerReference": "cli-example",
"Aliases": {
  "Quantity": 0
},
"DefaultRootObject": "index.html",
"Origins": {
  "Quantity": 1,
  "Items": [
    {
      "Id": "awsexamplebucket.s3.amazonaws.com-cli-
example",
      "DomainName": "awsexamplebucket.s3.amazonaws.com",
      "OriginPath": "",
      "CustomHeaders": {
        "Quantity": 0
      },
      "S3OriginConfig": {
        "OriginAccessIdentity": ""
      }
    }
  ]
},
"OriginGroups": {
  "Quantity": 0
},
"DefaultCacheBehavior": {
  "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-
example",
  "ForwardedValues": {
    "QueryString": false,
    "Cookies": {
      "Forward": "none"
    },
    "Headers": {
      "Quantity": 0
    },
    "QueryStringCacheKeys": {
      "Quantity": 0
    }
  },
  "TrustedSigners": {
    "Enabled": false,
    "Quantity": 0
  },
}
```

```
    "ViewerProtocolPolicy": "allow-all",
    "MinTTL": 0,
    "AllowedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ],
      "CachedMethods": {
        "Quantity": 2,
        "Items": [
          "HEAD",
          "GET"
        ]
      }
    },
    "SmoothStreaming": false,
    "DefaultTTL": 86400,
    "MaxTTL": 31536000,
    "Compress": false,
    "LambdaFunctionAssociations": {
      "Quantity": 0
    },
    "FieldLevelEncryptionId": ""
  },
  "CacheBehaviors": {
    "Quantity": 0
  },
  "CustomErrorResponses": {
    "Quantity": 0
  },
  "Comment": "",
  "Logging": {
    "Enabled": false,
    "IncludeCookies": false,
    "Bucket": "",
    "Prefix": ""
  },
  "PriceClass": "PriceClass_All",
  "Enabled": true,
  "ViewerCertificate": {
    "CloudFrontDefaultCertificate": true,
    "MinimumProtocolVersion": "TLSv1",
    "CertificateSource": "cloudfront"
```

```

    },
    "Restrictions": {
      "GeoRestriction": {
        "RestrictionType": "none",
        "Quantity": 0
      }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
  }
},
{
  "Id": "EDFDVBD6EXAMPLE",
  "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
  "Status": "InProgress",
  "LastModifiedTime": "2019-12-04T23:35:41.433Z",
  "InProgressInvalidationBatches": 0,
  "DomainName": "d930174dauwrn8.cloudfront.net",
  "ActiveTrustedSigners": {
    "Enabled": false,
    "Quantity": 0
  },
  "DistributionConfig": {
    "CallerReference": "cli-example",
    "Aliases": {
      "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
      "Quantity": 1,
      "Items": [
        {
          "Id": "awsexamplebucket1.s3.amazonaws.com-cli-example",
          "DomainName": "awsexamplebucket1.s3.amazonaws.com",
          "OriginPath": "",
          "CustomHeaders": {
            "Quantity": 0
          },
          "S3OriginConfig": {
            "OriginAccessIdentity": ""
          }
        }
      ]
    }
  }
}

```

```
        }
      ]
    },
    "OriginGroups": {
      "Quantity": 0
    },
    "DefaultCacheBehavior": {
      "TargetOriginId": "awsexamplebucket1.s3.amazonaws.com-cli-
example",
      "ForwardedValues": {
        "QueryString": false,
        "Cookies": {
          "Forward": "none"
        },
        "Headers": {
          "Quantity": 0
        },
        "QueryStringCacheKeys": {
          "Quantity": 0
        }
      },
      "TrustedSigners": {
        "Enabled": false,
        "Quantity": 0
      },
      "ViewerProtocolPolicy": "allow-all",
      "MinTTL": 0,
      "AllowedMethods": {
        "Quantity": 2,
        "Items": [
          "HEAD",
          "GET"
        ],
        "CachedMethods": {
          "Quantity": 2,
          "Items": [
            "HEAD",
            "GET"
          ]
        }
      },
      "SmoothStreaming": false,
      "DefaultTTL": 86400,
      "MaxTTL": 31536000,
```



```

        "Compress": false,
        "LambdaFunctionAssociations": {
            "Quantity": 0
        },
        "FieldLevelEncryptionId": ""
    },
    "CacheBehaviors": {
        "Quantity": 0
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
    "Comment": "",
    "Logging": {
        "Enabled": false,
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": true,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
    },
    "Restrictions": {
        "GeoRestriction": {
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
    }
},
{
    "Id": "E1X5IZQEXAMPLE",
    "ARN": "arn:aws:cloudfront::123456789012:distribution/
E1X5IZQEXAMPLE",
    "Status": "Deployed",
    "LastModifiedTime": "2019-11-06T21:31:48.864Z",
    "DomainName": "d2e04y12345678.cloudfront.net",

```

```
    "Aliases": {
      "Quantity": 0
    },
    "Origins": {
      "Quantity": 1,
      "Items": [
        {
          "Id": "awsexamplebucket2",
          "DomainName": "awsexamplebucket2.s3.us-
west-2.amazonaws.com",
          "OriginPath": "",
          "CustomHeaders": {
            "Quantity": 0
          },
          "S3OriginConfig": {
            "OriginAccessIdentity": ""
          }
        }
      ]
    },
    "OriginGroups": {
      "Quantity": 0
    },
    "DefaultCacheBehavior": {
      "TargetOriginId": "awsexamplebucket2",
      "ForwardedValues": {
        "QueryString": false,
        "Cookies": {
          "Forward": "none"
        }
      },
      "Headers": {
        "Quantity": 0
      },
      "QueryStringCacheKeys": {
        "Quantity": 0
      }
    },
    "TrustedSigners": {
      "Enabled": false,
      "Quantity": 0
    },
    "ViewerProtocolPolicy": "allow-all",
    "MinTTL": 0,
    "AllowedMethods": {
```

```
        "Quantity": 2,
        "Items": [
            "HEAD",
            "GET"
        ],
        "CachedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ]
        }
    },
    "SmoothStreaming": false,
    "DefaultTTL": 86400,
    "MaxTTL": 31536000,
    "Compress": false,
    "LambdaFunctionAssociations": {
        "Quantity": 0
    },
    "FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
    "Quantity": 0
},
"CustomErrorResponses": {
    "Quantity": 0
},
"Comment": "",
"PriceClass": "PriceClass_All",
"Enabled": true,
"ViewerCertificate": {
    "CloudFrontDefaultCertificate": true,
    "MinimumProtocolVersion": "TLSv1",
    "CertificateSource": "cloudfront"
},
"Restrictions": {
    "GeoRestriction": {
        "RestrictionType": "none",
        "Quantity": 0
    }
},
"WebACLId": "",
"HttpVersion": "HTTP1_1",
```

```

        "IsIPV6Enabled": true
      }
    ]
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDistributions](#)中的。

## list-field-level-encryption-configs

以下代码示例演示了如何使用 list-field-level-encryption-configs。

### AWS CLI

列出 CloudFront 字段级加密配置

以下示例获取了您 AWS 账户中 CloudFront 字段级加密配置的列表：

```
aws cloudfront list-field-level-encryption-configs
```

输出：

```

{
  "FieldLevelEncryptionList": {
    "MaxItems": 100,
    "Quantity": 1,
    "Items": [
      {
        "Id": "C3KM2WVD605UAY",
        "LastModifiedTime": "2019-12-10T21:30:18.974Z",
        "Comment": "Example FLE configuration",
        "QueryArgProfileConfig": {
          "ForwardWhenQueryArgProfileIsUnknown": true,
          "QueryArgProfiles": {
            "Quantity": 0,
            "Items": []
          }
        },
        "ContentTypeProfileConfig": {
          "ForwardWhenContentTypeIsUnknown": true,
          "ContentTypeProfiles": {
            "Quantity": 1,

```

```

        "Items": [
            {
                "Format": "URLEncoded",
                "ProfileId": "P280MFCLSY0CVU",
                "ContentType": "application/x-www-form-urlencoded"
            }
        ]
    }
}

```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListFieldLevelEncryptionConfigs](#) 中的。

## list-field-level-encryption-profiles

以下代码示例演示了如何使用 `list-field-level-encryption-profiles`。

### AWS CLI

列出 CloudFront 字段级加密配置文件

以下示例获取了您 AWS 账户中的 CloudFront 字段级加密配置文件列表：

```
aws cloudfront list-field-level-encryption-profiles
```

输出：

```

{
  "FieldLevelEncryptionProfileList": {
    "MaxItems": 100,
    "Quantity": 2,
    "Items": [
      {
        "Id": "P280MFCLSY0CVU",
        "LastModifiedTime": "2019-12-05T01:05:39.896Z",
        "Name": "ExampleFLEProfile",
        "EncryptionEntities": {
          "Quantity": 1,
          "Items": [
            {

```

```

        "PublicKeyId": "K2K8NC4HVFE3M0",
        "ProviderId": "ExampleFLEProvider",
        "FieldPatterns": {
            "Quantity": 1,
            "Items": [
                "ExampleSensitiveField"
            ]
        }
    ],
    },
    "Comment": "FLE profile for AWS CLI example"
},
{
    "Id": "PPK0UOSIF5WSV",
    "LastModifiedTime": "2019-12-10T01:03:16.537Z",
    "Name": "ExampleFLEProfile2",
    "EncryptionEntities": {
        "Quantity": 1,
        "Items": [
            {
                "PublicKeyId": "K2ABC10EXAMPLE",
                "ProviderId": "ExampleFLEProvider2",
                "FieldPatterns": {
                    "Quantity": 1,
                    "Items": [
                        "ExampleSensitiveField2"
                    ]
                }
            }
        ]
    },
    "Comment": "FLE profile #2 for AWS CLI example"
}
]
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFieldLevelEncryptionProfiles](#)中的。

## list-invalidations

以下代码示例演示了如何使用 list-invalidations。

## AWS CLI

列出 CloudFront 失效情况

以下示例获取了带有 ID 的 CloudFront分配的失效列表：EDFDVBD6EXAMPLE

```
aws cloudfront list-invalidations --distribution-id EDFDVBD6EXAMPLE
```

输出：

```
{
  "InvalidationList": {
    "Marker": "",
    "Items": [
      {
        "Status": "Completed",
        "Id": "YNY2LI2BVJ4NJU",
        "CreateTime": "2019-08-31T21:15:52.042Z"
      }
    ],
    "IsTruncated": false,
    "MaxItems": 100,
    "Quantity": 1
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListInvalidations](#)中的。

## list-public-keys

以下代码示例演示了如何使用 list-public-keys。

## AWS CLI

列出 CloudFront 公钥

以下示例获取了您 AWS 账户中的 CloudFront 公钥列表：

```
aws cloudfront list-public-keys
```

输出：

```

{
  "PublicKeyList": {
    "MaxItems": 100,
    "Quantity": 2,
    "Items": [
      {
        "Id": "K2K8NC4HVFE3M0",
        "Name": "ExampleKey",
        "CreatedTime": "2019-12-05T01:04:28.818Z",
        "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAA0CAQ8AMIIBCgKCAQEAxPmBCA2Ks01nd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1U0WcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAz03ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAUQPnhinFJ8Wdy8YBXSPpy7r95y1UQd9LfYTBzVZYG2tSesplc0kjm3\n2Uu
+oMWxQAw1NINnSLPinMVsutJy6Zq1V3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nnq
+kGZ2NQ0FyIyT2eiLK0X5RgB/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nnrwIDAQAB\n-----END
PUBLIC KEY-----\n",
        "Comment": "example public key"
      },
      {
        "Id": "K1S0LWQ2L5HTBU",
        "Name": "ExampleKey2",
        "CreatedTime": "2019-12-09T23:28:11.110Z",
        "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAA0CAQ8AMIIBCgKCAQEAp0CAg88A8+f4dujn9Izt
\n26LxtgAkn2opGgo/NKpMiaisyw5qlg3f1gol7FV6pYN178iJg3E08JBbwt1H
+cR9\nLGSf60NDeVhm760c39Np/vWg0dsGQcRbi9WmKZeS0DqjQGzVZWqPmito3FzWV6b
\nfVY5N36U/RdbVAJm95Km+qaMY1bIdF40t72bi3IkKYV5h1B2XoDjlQ9F6ajQKyTB
\nMHa3SN8q+3ZjQ4sJJ7D1V6r4wR8jDcFVD5NckWJmmgIVnk0QM37NYeoDnka0uTpu\nnha/
+3b8t0b2z3LBVHPkp85zJRA0XacSwf5rZtPYKBNFsixTa2n55k2r218m0kMC4\nUwIDAQAB\n-----END
PUBLIC KEY-----",
        "Comment": "example public key #2"
      }
    ]
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPublicKeys](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 list-tags-for-resource。



## AWS CLI

### 列出 CloudFront 分配的标签

以下示例获取 CloudFront 分配的标签列表：

```
aws cloudfront list-tags-for-resource \  
  --resource arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE
```

输出：

```
{  
  "Tags": {  
    "Items": [  
      {  
        "Key": "DateCreated",  
        "Value": "2019-12-04"  
      },  
      {  
        "Key": "Name",  
        "Value": "Example name"  
      },  
      {  
        "Key": "Project",  
        "Value": "Example project"  
      }  
    ]  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## sign

以下代码示例演示了如何使用 sign。

## AWS CLI

### 为 CloudFront 网址签名

以下示例为 CloudFront URL 签名。要签名 URL，您需要密钥对 ID（在 AWS 管理控制台中称为访问密钥 ID）和可信签名者的密钥对的私 CloudFront 钥。有关签名网址的更多信息，请参阅《亚马逊 CloudFront 开发者指南》中的[使用签名网址和签名 Cookie 提供私有内容](#)。

```
aws cloudfront sign \  
  --url https://d111111abcdef8.cloudfront.net/private-content/private-file.html \  
  --key-pair-id APKAEIBAERJR2EXAMPLE \  
  --private-key file://cf-signer-priv-key.pem \  
  --date-less-than 2020-01-01
```

输出：

```
https://d111111abcdef8.cloudfront.net/private-content/private-  
file.html?Expires=1577836800&Signature=nEXK7Kby47XKeZQKvc6pwkif6oZc-  
JWSpDkH0UH7EBGGqvgurkecCbgL5VfUAXyLQuJxFwRQWscz-  
owcq9KpmewCXrXQbPaJZNi9XSNwf4YKurPDQYaRQawKoeenH0GFteRf9ELK-  
Bs3nljTLjtbgzIUt7QJNKXcWr8AuUYikzGdJ4-qzx6WnxXfH~fxg4-  
GG16l2kgCpXUB6Jx6K~Y3kpV0dzUP0IqFLHAnJojbhxqrVejomZZ2XrquDvNUCCIbePGnR3d24UPaLXG4FK0qNEaWDIB  
GNvjRjXqWf93uMobeM0iVYahb-e0KIItiQewGcm0eLZQ__&Key-Pair-Id=APKAEIBAERJR2EXAMPLE
```

- 有关 API 的详细信息，请参阅[登录AWS CLI命令参考](#)。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为 CloudFront 分配添加标签

以下 tag-resource 示例向指定的 CloudFront 发行版添加了两个标签。

```
aws cloudfront tag-resource \  
  --resource arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE \  
  --tags 'Items=[{Key=Name,Value="Example name"},{Key=Project,Value="Example  
project"}]'
```

您可以在 JSON 文件中提供标签，而不必使用命令行参数，如以下示例所示：

```
aws cloudfront tag-resource \  
  --resource arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE \  
  --tags file://tags.json
```

```
--tags file://tags.json
```

tags.json 的内容：

```
{
  "Items": [
    {
      "Key": "Name",
      "Value": "Example name"
    },
    {
      "Key": "Project",
      "Value": "Example project"
    }
  ]
}
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从 CloudFront 分配中移除标签

以下示例使用命令行参数从 CloudFront 发行版中删除两个标签：

```
aws cloudfront untag-resource \
  --resource arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE \
  --tag-keys Items=Name,Project
```

您可以在 JSON 文件中提供标签密钥，而不必使用命令行参数，如以下示例所示：

```
aws cloudfront untag-resource \
  --resource arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE \
  --tag-keys file://tag-keys.json
```

文件 tag-keys.json 是当前文件夹中包含以下内容的 JSON 文档：

```
{
  "Items": [
    "Name",
    "Project"
  ]
}
```

成功时，此命令没有输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-cloud-front-origin-access-identity

以下代码示例演示了如何使用 `update-cloud-front-origin-access-identity`。

### AWS CLI

#### 更新 CloudFront 源站访问身份

以下示例使用 ID `E74FTE3AEXAMPLE` 更新源访问身份 (OAI)。您可以更新的唯一字段是 OAI 字段。Comment

要更新 OAI，您必须拥有 OAI 的 ID 和 ETag。ETag OAI ID 将在 `-access-identity` 和 `create-cloud-front-origin-access-identitions` 命令的输出中返回。list-cloud-front-origin要获取ETag，请使用 `get-cloud-front-origin-access-identity` 或 `get-cloud-front-origin-identity` 命令。access-identity-config 使用 `--if-match` 选项提供 OAI。ETag

```
aws cloudfront update-cloud-front-origin-access-identity \
  --id E74FTE3AEXAMPLE \
  --if-match E2QWRUHEXAMPLE \
  --cloud-front-origin-access-identity-config \
    CallerReference=cli-example,Comment="Example OAI Updated"
```

您可以通过在 JSON 文件中提供 OAI 配置来完成同样的事情，如以下示例所示：

```
aws cloudfront update-cloud-front-origin-access-identity \
  --id E74FTE3AEXAMPLE \
  --if-match E2QWRUHEXAMPLE \
  --cloud-front-origin-access-identity-config file://OAI-config.json
```

该文件 `OAI-config.json` 是当前目录中的一个 JSON 文档，其中包含以下内容：

```
{
  "CallerReference": "cli-example",
  "Comment": "Example OAI Updated"
}
```

无论您为 OAI 配置提供命令行参数还是 JSON 文件，输出都是一样的：

```
{
  "ETag": "E9LHASXEXAMPLE",
  "CloudFrontOriginAccessIdentity": {
    "Id": "E74FTE3AEXAMPLE",
    "S3CanonicalUserId":
    "cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",
    "CloudFrontOriginAccessIdentityConfig": {
      "CallerReference": "cli-example",
      "Comment": "Example OAI Updated"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateCloudFrontOriginAccessIdentity](#)中的。

## update-distribution

以下代码示例演示了如何使用 update-distribution。

### AWS CLI

更新 CloudFront 分配的默认根对象

以下示例将CloudFront 分配的默认根对象更新为 IDEFDVBD6EXAMPLE : index.html

```
aws cloudfront update-distribution --id EDFDVBD6EXAMPLE \
  --default-root-object index.html
```

输出：

```
{
  "ETag": "E2QWRUHEXAMPLE",
  "Distribution": {
    "Id": "EDFDVBD6EXAMPLE",
    "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
```

```
"Status": "InProgress",
"LastModifiedTime": "2019-12-06T18:55:39.870Z",
"InProgressInvalidationBatches": 0,
"DomainName": "d111111abcdef8.cloudfront.net",
"ActiveTrustedSigners": {
  "Enabled": false,
  "Quantity": 0
},
"DistributionConfig": {
  "CallerReference": "6b10378d-49be-4c4b-a642-419ccaf8f3b5",
  "Aliases": {
    "Quantity": 0
  },
  "DefaultRootObject": "index.html",
  "Origins": {
    "Quantity": 1,
    "Items": [
      {
        "Id": "example-website",
        "DomainName": "www.example.com",
        "OriginPath": "",
        "CustomHeaders": {
          "Quantity": 0
        },
        "CustomOriginConfig": {
          "HTTPPort": 80,
          "HTTPSPort": 443,
          "OriginProtocolPolicy": "match-viewer",
          "OriginSslProtocols": {
            "Quantity": 2,
            "Items": [
              "SSLv3",
              "TLSv1"
            ]
          },
          "OriginReadTimeout": 30,
          "OriginKeepaliveTimeout": 5
        }
      }
    ]
  },
  "OriginGroups": {
    "Quantity": 0
  },
}
```

```
"DefaultCacheBehavior": {
  "TargetOriginId": "example-website",
  "ForwardedValues": {
    "QueryString": false,
    "Cookies": {
      "Forward": "none"
    },
    "Headers": {
      "Quantity": 1,
      "Items": [
        "*"
      ]
    },
    "QueryStringCacheKeys": {
      "Quantity": 0
    }
  },
  "TrustedSigners": {
    "Enabled": false,
    "Quantity": 0
  },
  "ViewerProtocolPolicy": "allow-all",
  "MinTTL": 0,
  "AllowedMethods": {
    "Quantity": 2,
    "Items": [
      "HEAD",
      "GET"
    ],
    "CachedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ]
    }
  },
  "SmoothStreaming": false,
  "DefaultTTL": 86400,
  "MaxTTL": 31536000,
  "Compress": false,
  "LambdaFunctionAssociations": {
    "Quantity": 0
  },
}
```

```

        "FieldLevelEncryptionId": ""
    },
    "CacheBehaviors": {
        "Quantity": 0
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
    "Comment": "",
    "Logging": {
        "Enabled": false,
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": true,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
    },
    "Restrictions": {
        "GeoRestriction": {
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http1.1",
    "IsIPV6Enabled": true
}
}
}

```

## 更新分 CloudFront 配

以下示例EMLARXS9EXAMPLE通过在名dist-config-disable.json为的 CloudFront JSON 文件中提供分发配置来禁用 ID 的分发。要更新分配，您必须使用 `--if-match` 选项来提供分配的 ETag。要获取ETag，请使用 `get-distribution` 或 `get-distribution-config` 命令。

使用以下示例禁用分配后，您可以使用 `delete-distribute` 命令将其删除。



```
aws cloudfront update-distribution \  
  --id EMLARXS9EXAMPLE \  
  --if-match E2QWRUHEXAMPLE \  
  --distribution-config file://dist-config-disable.json
```

文件 `dist-config-disable.json` 是当前文件夹中包含以下内容的 JSON 文档：请注意，`Enabled` 字段设置为 `false`：

```
{  
  "CallerReference": "cli-1574382155-496510",  
  "Aliases": {  
    "Quantity": 0  
  },  
  "DefaultRootObject": "index.html",  
  "Origins": {  
    "Quantity": 1,  
    "Items": [  
      {  
        "Id": "awsexamplebucket.s3.amazonaws.com-1574382155-273939",  
        "DomainName": "awsexamplebucket.s3.amazonaws.com",  
        "OriginPath": "",  
        "CustomHeaders": {  
          "Quantity": 0  
        },  
        "S3OriginConfig": {  
          "OriginAccessIdentity": ""  
        }  
      }  
    ]  
  },  
  "OriginGroups": {  
    "Quantity": 0  
  },  
  "DefaultCacheBehavior": {  
    "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-1574382155-273939",  
    "ForwardedValues": {  
      "QueryString": false,  
      "Cookies": {  
        "Forward": "none"  
      },  
      "Headers": {  
        "Quantity": 0  
      }  
    }  
  },  
}
```

```
    "QueryStringCacheKeys": {
      "Quantity": 0
    },
    "TrustedSigners": {
      "Enabled": false,
      "Quantity": 0
    },
    "ViewerProtocolPolicy": "allow-all",
    "MinTTL": 0,
    "AllowedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ],
      "CachedMethods": {
        "Quantity": 2,
        "Items": [
          "HEAD",
          "GET"
        ]
      }
    },
    "SmoothStreaming": false,
    "DefaultTTL": 86400,
    "MaxTTL": 31536000,
    "Compress": false,
    "LambdaFunctionAssociations": {
      "Quantity": 0
    },
    "FieldLevelEncryptionId": ""
  },
  "CacheBehaviors": {
    "Quantity": 0
  },
  "CustomErrorResponses": {
    "Quantity": 0
  },
  "Comment": "",
  "Logging": {
    "Enabled": false,
    "IncludeCookies": false,
    "Bucket": "",
```

```

    "Prefix": ""
  },
  "PriceClass": "PriceClass_All",
  "Enabled": false,
  "ViewerCertificate": {
    "CloudFrontDefaultCertificate": true,
    "MinimumProtocolVersion": "TLSv1",
    "CertificateSource": "cloudfront"
  },
  "Restrictions": {
    "GeoRestriction": {
      "RestrictionType": "none",
      "Quantity": 0
    }
  },
  "WebACLId": "",
  "HttpVersion": "http2",
  "IsIPV6Enabled": true
}

```

输出：

```

{
  "ETag": "E9LHASXEXAMPLE",
  "Distribution": {
    "Id": "EMLARXS9EXAMPLE",
    "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
    "Status": "InProgress",
    "LastModifiedTime": "2019-12-06T18:32:35.553Z",
    "InProgressInvalidationBatches": 0,
    "DomainName": "d111111abcdef8.cloudfront.net",
    "ActiveTrustedSigners": {
      "Enabled": false,
      "Quantity": 0
    }
  },
  "DistributionConfig": {
    "CallerReference": "cli-1574382155-496510",
    "Aliases": {
      "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
      "Quantity": 1,

```

```
    "Items": [
      {
        "Id": "awsexamplebucket.s3.amazonaws.com-1574382155-273939",
        "DomainName": "awsexamplebucket.s3.amazonaws.com",
        "OriginPath": "",
        "CustomHeaders": {
          "Quantity": 0
        },
        "S3OriginConfig": {
          "OriginAccessIdentity": ""
        }
      }
    ],
    "OriginGroups": {
      "Quantity": 0
    },
    "DefaultCacheBehavior": {
      "TargetOriginId":
"awsexamplebucket.s3.amazonaws.com-1574382155-273939",
      "ForwardedValues": {
        "QueryString": false,
        "Cookies": {
          "Forward": "none"
        },
        "Headers": {
          "Quantity": 0
        },
        "QueryStringCacheKeys": {
          "Quantity": 0
        }
      },
      "TrustedSigners": {
        "Enabled": false,
        "Quantity": 0
      },
      "ViewerProtocolPolicy": "allow-all",
      "MinTTL": 0,
      "AllowedMethods": {
        "Quantity": 2,
        "Items": [
          "HEAD",
          "GET"
        ],
      },
    },
  ],
}
```

```
        "CachedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ]
        }
    },
    "SmoothStreaming": false,
    "DefaultTTL": 86400,
    "MaxTTL": 31536000,
    "Compress": false,
    "LambdaFunctionAssociations": {
        "Quantity": 0
    },
    "FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
    "Quantity": 0
},
"CustomErrorResponses": {
    "Quantity": 0
},
"Comment": "",
"Logging": {
    "Enabled": false,
    "IncludeCookies": false,
    "Bucket": "",
    "Prefix": ""
},
"PriceClass": "PriceClass_All",
"Enabled": false,
"ViewerCertificate": {
    "CloudFrontDefaultCertificate": true,
    "MinimumProtocolVersion": "TLSv1",
    "CertificateSource": "cloudfront"
},
"Restrictions": {
    "GeoRestriction": {
        "RestrictionType": "none",
        "Quantity": 0
    }
},
"WebACLId": "",
```

```

        "HttpVersion": "http2",
        "IsIPV6Enabled": true
    }
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDistribution](#)中的。

## update-field-level-encryption-config

以下代码示例演示了如何使用 update-field-level-encryption-config。

### AWS CLI

#### 更新 CloudFront 字段级加密配置

以下示例C3KM2WVD605UAY通过在 JSON 文件中提供参数，使用 ID 更新Comment字段级加密配置的字段。

要更新字段级加密配置，您必须拥有该配置的 ID 和 ETag。ETag ID 将在-config create-field-level-encryption g 和 list-field-level-encryption-configs 命令的输出中返回。要获取ETag，请使用get-field-level-encryption 或 get-field-level-encryption-config 命令。使用--if-match选项提供配置ETag。

```

aws cloudfront update-field-level-encryption-config \
  --id C3KM2WVD605UAY \
  --if-match E2P4Z4VU7TY5SG \
  --field-level-encryption-config file://fle-config.json

```

该文件fle-config.json是当前目录中的一个 JSON 文档，其中包含以下内容：

```

{
  "CallerReference": "cli-example",
  "Comment": "Updated example FLE configuration",
  "QueryArgProfileConfig": {
    "ForwardWhenQueryArgProfileIsUnknown": true,
    "QueryArgProfiles": {
      "Quantity": 0
    }
  },
  "ContentTypeProfileConfig": {

```

```

    "ForwardWhenContentTypeIsUnknown": true,
    "ContentTypeProfiles": {
      "Quantity": 1,
      "Items": [
        {
          "Format": "URLEncoded",
          "ProfileId": "P280MFCLSYOCVU",
          "ContentType": "application/x-www-form-urlencoded"
        }
      ]
    }
  }
}

```

输出：

```

{
  "ETag": "E26M4BIAV81ZF6",
  "FieldLevelEncryption": {
    "Id": "C3KM2WVD605UAY",
    "LastModifiedTime": "2019-12-10T22:26:26.170Z",
    "FieldLevelEncryptionConfig": {
      "CallerReference": "cli-example",
      "Comment": "Updated example FLE configuration",
      "QueryArgProfileConfig": {
        "ForwardWhenQueryArgProfileIsUnknown": true,
        "QueryArgProfiles": {
          "Quantity": 0,
          "Items": []
        }
      }
    },
    "ContentTypeProfileConfig": {
      "ForwardWhenContentTypeIsUnknown": true,
      "ContentTypeProfiles": {
        "Quantity": 1,
        "Items": [
          {
            "Format": "URLEncoded",
            "ProfileId": "P280MFCLSYOCVU",
            "ContentType": "application/x-www-form-urlencoded"
          }
        ]
      }
    }
  }
}

```

```

    }
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateFieldLevelEncryptionConfig](#)中的。

## update-field-level-encryption-profile

以下代码示例演示了如何使用 `update-field-level-encryption-profile`。

### AWS CLI

#### 更新 CloudFront 字段级加密配置文件

以下示例使用 ID 更新字段级加密配置文件。PPK0U0SIF5WSV此示例通过在 JSON 文件中提供参数来更新配置文件的和，并添加第二FieldPatterns项。Name Comment

要更新字段级加密配置文件，您必须拥有该配置文件的 ID 和。ETag该 ID 将在-profile 和 create-field-level-encryption list-field-level-encryption-profiles 命令的输出中返回。要获取ETag，请使用-profile 或-p get-field-level-encryption rof get-field-level-encryption ile-config 命令。使用--if-match选项提供个人资料ETag。

```

aws cloudfront update-field-level-encryption-profile \
  --id PPK0U0SIF5WSV \
  --if-match E1QQG65FS2L2GC \
  --field-level-encryption-profile-config file://fle-profile-config.json

```

该文件fle-profile-config.json是当前目录中的一个 JSON 文档，其中包含以下内容：

```

{
  "Name": "ExampleFLEProfileUpdated",
  "CallerReference": "cli-example",
  "Comment": "Updated FLE profile for AWS CLI example",
  "EncryptionEntities": {
    "Quantity": 1,
    "Items": [
      {
        "PublicKeyId": "K2K8NC4HVFE3M0",
        "ProviderId": "ExampleFLEProvider",
        "FieldPatterns": {

```



```

        "Quantity": 2,
        "Items": [
            "ExampleSensitiveField",
            "SecondExampleSensitiveField"
        ]
    }
}

```

输出：

```

{
  "ETag": "EJETYFJ9CL66D",
  "FieldLevelEncryptionProfile": {
    "Id": "PPK0UOSIF5WSV",
    "LastModifiedTime": "2019-12-10T19:05:58.296Z",
    "FieldLevelEncryptionProfileConfig": {
      "Name": "ExampleFLEProfileUpdated",
      "CallerReference": "cli-example",
      "Comment": "Updated FLE profile for AWS CLI example",
      "EncryptionEntities": {
        "Quantity": 1,
        "Items": [
          {
            "PublicKeyId": "K2K8NC4HVFE3M0",
            "ProviderId": "ExampleFLEProvider",
            "FieldPatterns": {
              "Quantity": 2,
              "Items": [
                "ExampleSensitiveField",
                "SecondExampleSensitiveField"
              ]
            }
          }
        ]
      }
    }
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateFieldLevelEncryptionProfile](#)中的。

## 使用亚马逊的 CloudSearch 示例 AWS CLI

以下代码示例向您展示如何在 Amazon 中使用来执行操作和实现常见场景 CloudSearch。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### upload-documents

以下代码示例演示了如何使用 upload-documents。

AWS CLI

以下upload-documents命令将一批 JSON 文档上传到亚马逊 CloudSearch 域名：

```
aws cloudsearchdomain upload-documents --endpoint-url https://doc-my-domain.us-west-1.cloudsearch.amazonaws.com --content-type application/json --documents document-batch.json
```

输出：

```
{
  "status": "success",
  "adds": 5000,
  "deletes": 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UploadDocuments](#)中的。

## CloudTrail 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 CloudTrail。 AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### add-tags

以下代码示例演示了如何使用 add-tags。

AWS CLI

向跟踪添加标签

以下add-tags命令为以下命令添加标签Trail1：

```
aws cloudtrail add-tags --resource-id arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail1 --tags-list Key=name,Value=Alice
Key=location,Value=us
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddTags](#)中的。

### create-subscription

以下代码示例演示了如何使用 create-subscription。

AWS CLI

为跟踪创建和配置 AWS 资源

以下`create-subscription`命令为以下内容创建新的 S3 存储桶和 SNS 主题：Trail1

```
aws cloudtrail create-subscription --name Trail1 --s3-new-bucket my-bucket --sns-  
new-topic my-topic
```

输出：

```
Setting up new S3 bucket my-bucket...  
Setting up new SNS topic my-topic...  
Creating/updating CloudTrail configuration...  
CloudTrail configuration:  
{  
  "trailList": [  
    {  
      "IncludeGlobalServiceEvents": true,  
      "Name": "Trail1",  
      "TrailARN": "arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail1",  
      "LogFileValidationEnabled": false,  
      "IsMultiRegionTrail": false,  
      "S3BucketName": "my-bucket",  
      "SnsTopicName": "my-topic",  
      "HomeRegion": "us-east-1"  
    }  
  ],  
  "ResponseMetadata": {  
    "HTTPStatusCode": 200,  
    "RequestId": "f39e51f6-c615-11e5-85bd-d35ca21ee3e2"  
  }  
}  
Starting CloudTrail service...  
Logs will be delivered to my-bucket
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSubscription](#)中的。

## create-trail

以下代码示例演示了如何使用 `create-trail`。

### AWS CLI

#### 创建跟踪

以下`create-trail`命令创建名为的多区域跟踪Trail1并指定一个 S3 存储桶：

```
aws cloudtrail create-trail --name Trail1 --s3-bucket-name my-bucket --is-multi-region-trail
```

输出：

```
{
  "IncludeGlobalServiceEvents": true,
  "Name": "Trail1",
  "TrailARN": "arn:aws:cloudtrail:us-west-2:123456789012:trail/Trail1",
  "LogFileValidationEnabled": false,
  "IsMultiRegionTrail": true,
  "S3BucketName": "my-bucket"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTrail](#)中的。

## **delete-trail**

以下代码示例演示了如何使用 `delete-trail`。

### AWS CLI

#### 删除跟踪

以下`delete-trail`命令删除名为的跟踪Trail1：

```
aws cloudtrail delete-trail --name Trail1
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTrail](#)中的。

## **describe-trails**

以下代码示例演示了如何使用 `describe-trails`。

### AWS CLI

#### 描述一条小径

以下describe-trails命令返回Trail1和的设置Trail2：

```
aws cloudtrail describe-trails --trail-name-list Trail1 Trail2
```

输出：

```
{
  "trailList": [
    {
      "IncludeGlobalServiceEvents": true,
      "Name": "Trail1",
      "TrailARN": "arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail1",
      "LogFileValidationEnabled": false,
      "IsMultiRegionTrail": false,
      "S3BucketName": "my-bucket",
      "CloudWatchLogsRoleArn": "arn:aws:iam::123456789012:role/
CloudTrail_CloudWatchLogs_Role",
      "CloudWatchLogsLogGroupArn": "arn:aws:logs:us-east-1:123456789012:log-
group:CloudTrail:*",
      "SnsTopicName": "my-topic",
      "HomeRegion": "us-east-1"
    },
    {
      "IncludeGlobalServiceEvents": true,
      "Name": "Trail2",
      "S3KeyPrefix": "my-prefix",
      "TrailARN": "arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail2",
      "LogFileValidationEnabled": false,
      "IsMultiRegionTrail": false,
      "S3BucketName": "my-bucket",
      "KmsKeyId": "arn:aws:kms:us-
east-1:123456789012:key/4c5ae5ac-3c13-421e-8335-c7868ef6a769",
      "HomeRegion": "us-east-1"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTrails](#)中的。

## get-event-selectors

以下代码示例演示了如何使用 get-event-selectors。

## AWS CLI

### 查看跟踪的事件选择器设置

以下`get-event-selectors`命令返回的设置Trail1：

```
aws cloudtrail get-event-selectors --trail-name Trail1
```

输出：

```
{
  "EventSelectors": [
    {
      "IncludeManagementEvents": true,
      "DataResources": [],
      "ReadWriteType": "All"
    }
  ],
  "TrailARN": "arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail1"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetEventSelectors](#)中的。

## get-trail-status

以下代码示例演示了如何使用 `get-trail-status`。

### AWS CLI

#### 获取跟踪的状态

以下`get-trail-status`命令返回以下内容的交付和日志详细信息Trail1：

```
aws cloudtrail get-trail-status --name Trail1
```

输出：

```
{
  "LatestNotificationTime": 1454022144.869,
  "LatestNotificationAttemptSucceeded": "2016-01-28T23:02:24Z",
  "LatestDeliveryAttemptTime": "2016-01-28T23:02:24Z",
}
```

```

"LatestDeliveryTime": 1454022144.869,
"TimeLoggingStarted": "2015-11-06T18:36:38Z",
"LatestDeliveryAttemptSucceeded": "2016-01-28T23:02:24Z",
"IsLogging": true,
"LatestCloudWatchLogsDeliveryTime": 1454022144.918,
"StartLoggingTime": 1446834998.695,
"StopLoggingTime": 1446834996.933,
"LatestNotificationAttemptTime": "2016-01-28T23:02:24Z",
"TimeLoggingStopped": "2015-11-06T18:36:36Z"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTrailStatus](#)中的。

## list-public-keys

以下代码示例演示了如何使用 list-public-keys。

### AWS CLI

列出跟踪的所有公钥

以下list-public-keys命令返回在指定时间范围内使用其私钥对摘要文件进行签名的所有公钥：

```
aws cloudtrail list-public-keys --start-time 2016-01-01T20:30:00.000Z
```

输出：

```

{
  "PublicKeyList": [
    {
      "ValidityStartTime": 1453076702.0,
      "ValidityEndTime": 1455668702.0,
      "Value": "MIIBCgKCAQEA1SS3c192HDycr/MTj0mo0has8habjrraXw+Kz1WF0axSI2tcF
+3iJ9BKQAVSKxGwxwu3m0wG3J
+kU11xboEcEPHYoIYMbgfSw7KGnuDKwkLzsQWhUJ0cIb0HASox1vv/5fNXkrHhGbDCHeVXm804c83nvHUEFYThr1PfyP
+4WGDk+BGH5m9iuiAKkipEHWmU18/P7XpfpWQuk4h8g3pXZ0rNXr081bh4d39svj7Uqdhv0XoBISp9t/
EXYuePGEtBdrKD9Dz+VHwyUPtBQvYr9BnkF88qBnaPNhS44rzwIDAQAB",
      "Fingerprint": "7f3f401420072e50a65a141430817ab3"
    }
  ]
}

```



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPublicKeys](#)中的。

## list-tags

以下代码示例演示了如何使用 list-tags。

### AWS CLI

列出跟踪的标签

以下list-tags命令列出了Trail1和的标签Trail2：

```
aws cloudtrail list-tags --resource-id-list arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail1 arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail2
```

输出：

```
{
  "ResourceTagList": [
    {
      "ResourceId": "arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail1",
      "TagsList": [
        {
          "Value": "Alice",
          "Key": "name"
        },
        {
          "Value": "us",
          "Key": "location"
        }
      ]
    },
    {
      "ResourceId": "arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail2",
      "TagsList": [
        {
          "Value": "Bob",
          "Key": "name"
        }
      ]
    }
  ]
}
```

```
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTags](#)中的。

## lookup-events

以下代码示例演示了如何使用 lookup-events。

### AWS CLI

#### 查找轨迹的事件

以下lookup-events命令按属性查找 API 活动事件EventName：

```
aws cloudtrail lookup-events --lookup-attributes
AttributeKey=EventName,AttributeValue=ConsoleLogin
```

输出：

```
{
  "Events": [
    {
      "EventId": "654ccbc0-ba0d-486a-9076-dbf7274677a7",
      "Username": "my-session-name",
      "EventTime": "2021-11-18T09:41:02-08:00",
      "CloudTrailEvent": "{\"eventVersion\":\"1.02\",\"userIdentity\":{\"type\":\"AssumedRole\",\"principalId\":\"AR0AJIKPFTA72SWU4L7T4:my-session-name\",\"arn\":\"arn:aws:sts::123456789012:assumed-role/my-role/my-session-name\",\"accountId\":\"123456789012\",\"sessionContext\":{\"attributes\":{\"mfaAuthenticated\":\"false\"},\"creationDate\":\"2016-01-26T21:42:12Z\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId\":\"AR0AJIKPFTA72SWU4L7T4\",\"arn\":\"arn:aws:iam:123456789012:role/my-role\",\"accountId\":\"123456789012\",\"userName\":\"my-role\"}}},\"eventTime\":\"2016-01-26T21:42:12Z\",\"eventSource\":\"signin.amazonaws.com\",\"eventName\":\"ConsoleLogin\",\"awsRegion\":\"us-east-1\",\"sourceIPAddress\":\"72.21.198.70\",\"userAgent\":\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36\",\"requestParameters\":null,\"responseElements\":{\"ConsoleLogin\":{\"Success\"}},\"additionalEventData\":{\"MobileVersion\":\"No\",\"MFAUsed\":\"No\"},\"eventID\":\"654ccbc0-ba0d-486a-9076-dbf7274677a7\",\"eventType\":\"AwsConsoleSignIn\",\"recipientAccountId\":\"123456789012\"}",
      "EventName": "ConsoleLogin",
    }
  ]
}
```

```

        "Resources": []
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[LookupEvents](#)中的。

## put-event-selectors

以下代码示例演示了如何使用 put-event-selectors。

### AWS CLI

为跟踪配置事件选择器

要创建事件选择器，请运行“put-event-selectors”命令。当您的账户中发生事件时，CloudTrail 会评估您的跟踪配置。如果事件匹配跟踪的任何事件选择器，则跟踪将处理并记录事件。您可以为一个跟踪配置最多 5 个事件选择器和最多 250 个数据资源。

以下示例为名为“TrailName”的跟踪创建事件选择器，以包括只读和只写管理事件、两个 Amazon S3 存储桶/前缀组合的数据事件以及名为“”的单个 Lambda 函数 AWS 的数据事件：hello-world-python-function

```

aws cloudtrail put-event-selectors --trail-name TrailName --event-
selectors '[{"ReadWriteType": "All","IncludeManagementEvents":
  true,"DataResources": [{"Type":"AWS::S3::Object", "Values":
  ["arn:aws:s3:::mybucket/prefix","arn:aws:s3:::mybucket2/prefix2"]},
{"Type": "AWS::Lambda::Function","Values": ["arn:aws:lambda:us-
west-2:999999999999:function:hello-world-python-function"]}]]]'

```

输出：

```

{
  "EventSelectors": [
    {
      "IncludeManagementEvents": true,
      "DataResources": [
        {
          "Values": [
            "arn:aws:s3:::mybucket/prefix",
            "arn:aws:s3:::mybucket2/prefix2"
          ],

```

```

        "Type": "AWS::S3::Object"
      },
      {
        "Values": [
          "arn:aws:lambda:us-west-2:123456789012:function:hello-world-
python-function"
        ],
        "Type": "AWS::Lambda::Function"
      },
    ],
    "ReadWriteType": "All"
  }
],
"TrailARN": "arn:aws:cloudtrail:us-east-2:123456789012:trail/TrailName"
}

```

以下示例为名为“TrailName2”的跟踪创建事件选择器，其中包括所有事件，包括只读和只写管理事件，以及账户中所有 Amazon S3 存储桶和 Lambda 函数 AWS 的所有数据事件：AWS

```

aws cloudtrail put-event-selectors --trail-name TrailName2 --event-selectors
' [{"ReadWriteType": "All", "IncludeManagementEvents": true, "DataResources":
[{"Type": "AWS::S3::Object", "Values": ["arn:aws:s3:::"}], {"Type":
"AWS::Lambda::Function", "Values": ["arn:aws:lambda"]}]} ]'

```

输出：

```

{
  "EventSelectors": [
    {
      "IncludeManagementEvents": true,
      "DataResources": [
        {
          "Values": [
            "arn:aws:s3:::"
          ],
          "Type": "AWS::S3::Object"
        },
        {
          "Values": [
            "arn:aws:lambda"
          ],
          "Type": "AWS::Lambda::Function"
        }
      ]
    }
  ]
}

```

```
    ],  
    "ReadWriteType": "All"  
  }  
],  
"TrailARN": "arn:aws:cloudtrail:us-east-2:123456789012:trail/TrailName2"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutEventSelectors](#)中的。

## remove-tags

以下代码示例演示了如何使用 `remove-tags`。

### AWS CLI

移除跟踪的标签

以下`remove-tags`命令删除的指定标签`Trail1`：

```
aws cloudtrail remove-tags --resource-id arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail1 --tags-list Key=name Key=location
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveTags](#)中的。

## start-logging

以下代码示例演示了如何使用 `start-logging`。

### AWS CLI

开始记录跟踪

以下`start-logging`命令开启对以下各项的日志记录`Trail1`：

```
aws cloudtrail start-logging --name Trail1
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartLogging](#)中的。

## stop-logging

以下代码示例演示了如何使用 `stop-logging`。

## AWS CLI

### 停止记录跟踪

以下stop-logging命令关闭对的日志记录Trail1：

```
aws cloudtrail stop-logging --name Trail1
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopLogging](#)中的。

## update-subscription

以下代码示例演示了如何使用 update-subscription。

## AWS CLI

### 更新跟踪的配置设置

以下update-subscription命令更新跟踪以指定新的 S3 存储桶和 SNS 主题：

```
aws cloudtrail update-subscription --name Trail1 --s3-new-bucket my-bucket-new --  
sns-new-topic my-topic-new
```

输出：

```
Setting up new S3 bucket my-bucket-new...  
Setting up new SNS topic my-topic-new...  
Creating/updating CloudTrail configuration...  
CloudTrail configuration:  
{  
  "trailList": [  
    {  
      "IncludeGlobalServiceEvents": true,  
      "Name": "Trail1",  
      "TrailARN": "arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail1",  
      "LogFileValidationEnabled": false,  
      "IsMultiRegionTrail": false,  
      "S3BucketName": "my-bucket-new",  
      "SnsTopicName": "my-topic-new",  
      "HomeRegion": "us-east-1"  
    }  
  ],  
}
```

```
"ResponseMetadata": {  
  "HTTPStatusCode": 200,  
  "RequestId": "31126f8a-c616-11e5-9cc6-2fd637936879"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateSubscription](#)中的。

## update-trail

以下代码示例演示了如何使用 update-trail。

### AWS CLI

#### 更新跟踪

以下update-trail命令更新跟踪以使用现有存储桶进行日志传输：

```
aws cloudtrail update-trail --name Trail1 --s3-bucket-name my-bucket
```

输出：

```
{  
  "IncludeGlobalServiceEvents": true,  
  "Name": "Trail1",  
  "TrailARN": "arn:aws:cloudtrail:us-west-2:123456789012:trail/Trail1",  
  "LogFileValidationEnabled": false,  
  "IsMultiRegionTrail": true,  
  "S3BucketName": "my-bucket"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateTrail](#)中的。

## validate-logs

以下代码示例演示了如何使用 validate-logs。

### AWS CLI

#### 验证日志文件

以下validate-logs命令验证日志：Trail1

```
aws cloudtrail validate-logs --trail-arn arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail1 --start-time 20160129T19:00:00Z
```

输出：

```
Validating log files for trail arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail1 between 2016-01-29T19:00:00Z and 2016-01-29T22:15:43Z
Results requested for 2016-01-29T19:00:00Z to 2016-01-29T22:15:43Z
Results found for 2016-01-29T19:24:57Z to 2016-01-29T21:24:57Z:
3/3 digest files valid
15/15 log files valid
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ValidateLogs](#)中的。

## CloudWatch 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 CloudWatch。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### delete-alarms

以下代码示例演示了如何使用 delete-alarms。

AWS CLI

删除警报

以下示例使用delete-alarms命令删除名为“myalarm”的亚马逊 CloudWatch 警报：



```
aws cloudwatch delete-alarms --alarm-names myalarm
```

输出：

```
This command returns to the prompt if successful.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAlarms](#)中的。

## describe-alarm-history

以下代码示例演示了如何使用 describe-alarm-history。

### AWS CLI

检索警报的历史记录

以下示例使用describe-alarm-history命令检索名为“myalarm”的亚马逊CloudWatch 警报的历史记录：

```
aws cloudwatch describe-alarm-history --alarm-name "myalarm" --history-item-type  
StateUpdate
```

输出：

```
{  
  "AlarmHistoryItems": [  
    {  
      "Timestamp": "2014-04-09T18:59:06.442Z",  
      "HistoryItemType": "StateUpdate",  
      "AlarmName": "myalarm",  
      "HistoryData": "{\"version\":\"1.0\",\"oldState\":{\"stateValue\":  
\\\"ALARM\\\",\\\"stateReason\\\":\\\"testing purposes\\\"},\\\"newState\":{\"stateValue\":  
\\\"OK\\\",\\\"stateReason\\\":\\\"Threshold Crossed: 2 datapoints were not greater than the  
threshold (70.0). The most recent datapoints: [38.958, 40.292].\\\",\\\"stateReasonData  
\\\":{\\\"version\\\":\\\"1.0\\\",\\\"queryDate\\\":\\\"2014-04-09T18:59:06.419+0000\\\",\\\"startDate  
\\\":\\\"2014-04-09T18:44:00.000+0000\\\",\\\"statistic\\\":\\\"Average\\\",\\\"period\\\":300,  
\\\"recentDatapoints\\\":[38.958,40.292],\\\"threshold\\\":70.0}}}\",  
      "HistorySummary": "Alarm updated from ALARM to OK"  
    },  
    {  
      "Timestamp": "2014-04-09T18:59:05.805Z",
```

```

        "HistoryItemType": "StateUpdate",
        "AlarmName": "myalarm",
        "HistoryData": "{\"version\":\"1.0\",\"oldState\":{\"stateValue\":"
    \":\"OK\",\"stateReason\":\"Threshold Crossed: 2 datapoints were
    not greater than the threshold (70.0). The most recent datapoints:
    [38.839999999999996, 39.714].\",\"stateReasonData\":{\"version\":
    \"1.0\",\"queryDate\":\"2014-03-11T22:45:41.569+0000\",\"startDate\":
    \"2014-03-11T22:30:00.000+0000\",\"statistic\":\"Average\",\"period\":300,
    \"recentDatapoints\":[38.839999999999996,39.714],\"threshold\":70.0}},\"newState\":
    {\"stateValue\":\"ALARM\",\"stateReason\":\"testing purposes\"}}",
        "HistorySummary": "Alarm updated from OK to ALARM"
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAlarmHistory](#)中的。

## describe-alarms-for-metric

以下代码示例演示了如何使用 `describe-alarms-for-metric`。

### AWS CLI

显示与指标关联的警报的相关信息

以下示例使用 `describe-alarms-for-metric` 命令显示与 Amazon EC2 CPUUtilization 指标和 ID 为 `i-0c986c72` 的实例关联的任何警报的相关信息：

```
aws cloudwatch describe-alarms-for-metric --metric-name CPUUtilization --namespace
AWS/EC2 --dimensions Name=InstanceId,Value=i-0c986c72
```

输出：

```

{
  "MetricAlarms": [
    {
      "EvaluationPeriods": 10,
      "AlarmArn": "arn:aws:cloudwatch:us-
east-1:111122223333:alarm:myHighCpuAlarm2",
      "StateUpdatedTimestamp": "2013-10-30T03:03:51.479Z",
      "AlarmConfigurationUpdatedTimestamp": "2013-10-30T03:03:50.865Z",
      "ComparisonOperator": "GreaterThanOrEqualToThreshold",

```

```

    "AlarmActions": [
      "arn:aws:sns:us-east-1:111122223333:NotifyMe"
    ],
    "Namespace": "AWS/EC2",
    "AlarmDescription": "CPU usage exceeds 70 percent",
    "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\\\"2013-10-30T03:03:51.479+0000\\\", \"startDate\": \"2013-10-30T02:08:00.000+0000\\\",
\\\"statistic\": \"Average\\\", \"period\": 300, \"recentDatapoints\":
[40.698, 39.612, 42.432, 39.796, 38.816, 42.28, 42.854, 40.088, 40.760000000000005, 41.316],
\\\"threshold\": 70.0}\",
    "Period": 300,
    "StateValue": "OK",
    "Threshold": 70.0,
    "AlarmName": "myHighCpuAlarm2",
    "Dimensions": [
      {
        "Name": "InstanceId",
        "Value": "i-0c986c72"
      }
    ],
    "Statistic": "Average",
    "StateReason": "Threshold Crossed: 10 datapoints were not greater than
or equal to the threshold (70.0). The most recent datapoints: [40.760000000000005,
41.316].",
    "InsufficientDataActions": [],
    "OKActions": [],
    "ActionsEnabled": true,
    "MetricName": "CPUUtilization"
  },
  {
    "EvaluationPeriods": 2,
    "AlarmArn": "arn:aws:cloudwatch:us-
east-1:111122223333:alarm:myHighCpuAlarm",
    "StateUpdatedTimestamp": "2014-04-09T18:59:06.442Z",
    "AlarmConfigurationUpdatedTimestamp": "2014-04-09T22:26:05.958Z",
    "ComparisonOperator": "GreaterThanThreshold",
    "AlarmActions": [
      "arn:aws:sns:us-east-1:111122223333:HighCPUAlarm"
    ],
    "Namespace": "AWS/EC2",
    "AlarmDescription": "CPU usage exceeds 70 percent",
    "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\\\"2014-04-09T18:59:06.419+0000\\\", \"startDate\": \"2014-04-09T18:44:00.000+0000\\\",

```

```

{"statistic": "Average", "period": 300, "recentDatapoints": [38.958, 40.292],
 "threshold": 70.0},
  {
    "Period": 300,
    "StateValue": "OK",
    "Threshold": 70.0,
    "AlarmName": "myHighCpuAlarm",
    "Dimensions": [
      {
        "Name": "InstanceId",
        "Value": "i-0c986c72"
      }
    ],
    "Statistic": "Average",
    "StateReason": "Threshold Crossed: 2 datapoints were not greater than
the threshold (70.0). The most recent datapoints: [38.958, 40.292].",
    "InsufficientDataActions": [],
    "OKActions": [],
    "ActionsEnabled": false,
    "MetricName": "CPUUtilization"
  }
]
}

```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAlarmsForMetric](#) 中的。

## describe-alarms

以下代码示例演示了如何使用 describe-alarms。

### AWS CLI

列出有关警报的信息

以下示例使用 describe-alarms 命令提供名为“myalarm”的警报的相关信息：

```
aws cloudwatch describe-alarms --alarm-names "myalarm"
```

输出：

```

{
  "MetricAlarms": [
    {
      "EvaluationPeriods": 2,

```

```

    "AlarmArn": "arn:aws:cloudwatch:us-east-1:123456789012:alarm:myalarm",
    "StateUpdatedTimestamp": "2014-04-09T18:59:06.442Z",
    "AlarmConfigurationUpdatedTimestamp": "2012-12-27T00:49:54.032Z",
    "ComparisonOperator": "GreaterThanThreshold",
    "AlarmActions": [
      "arn:aws:sns:us-east-1:123456789012:myHighCpuAlarm"
    ],
    "Namespace": "AWS/EC2",
    "AlarmDescription": "CPU usage exceeds 70 percent",
    "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\\\"2014-04-09T18:59:06.419+0000\\\",\\\"startDate\\\":\\\"2014-04-09T18:44:00.000+0000\\\",
\\\"statistic\\\":\\\"Average\\\",\\\"period\\\":300,\\\"recentDatapoints\\\":[38.958,40.292],
\\\"threshold\\\":70.0}\",
    "Period": 300,
    "StateValue": "OK",
    "Threshold": 70.0,
    "AlarmName": "myalarm",
    "Dimensions": [
      {
        "Name": "InstanceId",
        "Value": "i-0c986c72"
      }
    ],
    "Statistic": "Average",
    "StateReason": "Threshold Crossed: 2 datapoints were not greater than
the threshold (70.0). The most recent datapoints: [38.958, 40.292].",
    "InsufficientDataActions": [],
    "OKActions": [],
    "ActionsEnabled": true,
    "MetricName": "CPUUtilization"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAlarms](#)中的。

## disable-alarm-actions

以下代码示例演示了如何使用 disable-alarm-actions。

### AWS CLI

#### 禁用警报的操作

以下示例使用 `disable-alarm-actions` 命令禁用名为 `myalarm` 的警报的所有操作：

```
aws cloudwatch disable-alarm-actions --alarm-names myalarm
```

如果成功，该命令将返回到提示符。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableAlarmActions](#)中的。

## **enable-alarm-actions**

以下代码示例演示了如何使用 `enable-alarm-actions`。

### AWS CLI

启用警报的所有操作

以下示例使用 `enable-alarm-actions` 命令启用名为 `myalarm` 的警报的所有操作：

```
aws cloudwatch enable-alarm-actions --alarm-names myalarm
```

如果成功，该命令将返回到提示符。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableAlarmActions](#)中的。

## **get-metric-statistics**

以下代码示例演示了如何使用 `get-metric-statistics`。

### AWS CLI

获取每个 EC2 实例的 CPU 利用率

以下示例使用 `get-metric-statistics` 命令获取 ID 为 `i-abcdef` 的 EC2 实例的 CPU 利用率。

```
aws cloudwatch get-metric-statistics --metric-name CPUUtilization --start-time
2014-04-08T23:18:00Z --end-time 2014-04-09T23:18:00Z --period 3600 --namespace AWS/
EC2 --statistics Maximum --dimensions Name=InstanceId,Value=i-abcdef
```

输出：

```
{
  "Datapoints": [
    {
```

```
    "Timestamp": "2014-04-09T11:18:00Z",
    "Maximum": 44.79,
    "Unit": "Percent"
  },
  {
    "Timestamp": "2014-04-09T20:18:00Z",
    "Maximum": 47.92,
    "Unit": "Percent"
  },
  {
    "Timestamp": "2014-04-09T19:18:00Z",
    "Maximum": 50.85,
    "Unit": "Percent"
  },
  {
    "Timestamp": "2014-04-09T09:18:00Z",
    "Maximum": 47.92,
    "Unit": "Percent"
  },
  {
    "Timestamp": "2014-04-09T03:18:00Z",
    "Maximum": 76.84,
    "Unit": "Percent"
  },
  {
    "Timestamp": "2014-04-09T21:18:00Z",
    "Maximum": 48.96,
    "Unit": "Percent"
  },
  {
    "Timestamp": "2014-04-09T14:18:00Z",
    "Maximum": 47.92,
    "Unit": "Percent"
  },
  {
    "Timestamp": "2014-04-09T08:18:00Z",
    "Maximum": 47.92,
    "Unit": "Percent"
  },
  {
    "Timestamp": "2014-04-09T16:18:00Z",
    "Maximum": 45.55,
    "Unit": "Percent"
  },
  },
```

```
{
  "Timestamp": "2014-04-09T06:18:00Z",
  "Maximum": 47.92,
  "Unit": "Percent"
},
{
  "Timestamp": "2014-04-09T13:18:00Z",
  "Maximum": 45.08,
  "Unit": "Percent"
},
{
  "Timestamp": "2014-04-09T05:18:00Z",
  "Maximum": 47.92,
  "Unit": "Percent"
},
{
  "Timestamp": "2014-04-09T18:18:00Z",
  "Maximum": 46.88,
  "Unit": "Percent"
},
{
  "Timestamp": "2014-04-09T17:18:00Z",
  "Maximum": 52.08,
  "Unit": "Percent"
},
{
  "Timestamp": "2014-04-09T07:18:00Z",
  "Maximum": 47.92,
  "Unit": "Percent"
},
{
  "Timestamp": "2014-04-09T02:18:00Z",
  "Maximum": 51.23,
  "Unit": "Percent"
},
{
  "Timestamp": "2014-04-09T12:18:00Z",
  "Maximum": 47.67,
  "Unit": "Percent"
},
{
  "Timestamp": "2014-04-08T23:18:00Z",
  "Maximum": 46.88,
  "Unit": "Percent"
}
```



```
    },
    {
      "Timestamp": "2014-04-09T10:18:00Z",
      "Maximum": 51.91,
      "Unit": "Percent"
    },
    {
      "Timestamp": "2014-04-09T04:18:00Z",
      "Maximum": 47.13,
      "Unit": "Percent"
    },
    {
      "Timestamp": "2014-04-09T15:18:00Z",
      "Maximum": 48.96,
      "Unit": "Percent"
    },
    {
      "Timestamp": "2014-04-09T00:18:00Z",
      "Maximum": 48.16,
      "Unit": "Percent"
    },
    {
      "Timestamp": "2014-04-09T01:18:00Z",
      "Maximum": 49.18,
      "Unit": "Percent"
    }
  ],
  "Label": "CPUUtilization"
}
```

## 指定多个维度

以下示例说明了如何指定多个维度。每个维度都指定为一个“名称/值”对，名称和值之间用逗号隔开。多个维度之间用空格隔开。如果单个指标包含多个维度，则必须为每个已定义的维度指定一个值。

有关使用该 `get-metric-statistics` 命令的更多示例，请参阅 Amazon CloudWatch 开发者指南中的获取指标统计信息。

```
aws cloudwatch get-metric-statistics --metric-name Buffers --namespace MyNameSpace
--dimensions Name=InstanceID,Value=i-abcdef Name=InstanceType,Value=m1.small --
start-time 2016-10-15T04:00:00Z --end-time 2016-10-19T07:00:00Z --statistics Average
--period 60
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMetricStatistics](#)中的。

## list-metrics

以下代码示例演示了如何使用 list-metrics。

### AWS CLI

列出 Amazon SNS 的指标

以下 list-metrics 示例显示 Amazon SNS 的指标。

```
aws cloudwatch list-metrics \  
  --namespace "AWS/SNS"
```

输出：

```
{  
  "Metrics": [  
    {  
      "Namespace": "AWS/SNS",  
      "Dimensions": [  
        {  
          "Name": "TopicName",  
          "Value": "NotifyMe"  
        }  
      ],  
      "MetricName": "PublishSize"  
    },  
    {  
      "Namespace": "AWS/SNS",  
      "Dimensions": [  
        {  
          "Name": "TopicName",  
          "Value": "CF0"  
        }  
      ],  
      "MetricName": "PublishSize"  
    },  
    {  
      "Namespace": "AWS/SNS",  
      "Dimensions": [  
        {
```

```
        "Name": "TopicName",
        "Value": "NotifyMe"
    }
],
"MetricName": "NumberOfNotificationsFailed"
},
{
    "Namespace": "AWS/SNS",
    "Dimensions": [
        {
            "Name": "TopicName",
            "Value": "NotifyMe"
        }
    ],
    "MetricName": "NumberOfNotificationsDelivered"
},
{
    "Namespace": "AWS/SNS",
    "Dimensions": [
        {
            "Name": "TopicName",
            "Value": "NotifyMe"
        }
    ],
    "MetricName": "NumberOfMessagesPublished"
},
{
    "Namespace": "AWS/SNS",
    "Dimensions": [
        {
            "Name": "TopicName",
            "Value": "CF0"
        }
    ],
    "MetricName": "NumberOfMessagesPublished"
},
{
    "Namespace": "AWS/SNS",
    "Dimensions": [
        {
            "Name": "TopicName",
            "Value": "CF0"
        }
    ],
    "MetricName": "NumberOfMessagesPublished"
},
],
```

```

        "MetricName": "NumberOfNotificationsDelivered"
    },
    {
        "Namespace": "AWS/SNS",
        "Dimensions": [
            {
                "Name": "TopicName",
                "Value": "CF0"
            }
        ],
        "MetricName": "NumberOfNotificationsFailed"
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListMetrics](#)中的。

## put-metric-alarm

以下代码示例演示了如何使用 put-metric-alarm。

### AWS CLI

在 CPU 利用率超过 70% 时发送 Amazon Simple Notification Service 电子邮件消息

以下示例使用 put-metric-alarm 命令在 CPU 利用率超过 70% 时发送 Amazon Simple Notification Service 电子邮件消息：

```

aws cloudwatch put-metric-alarm --alarm-name cpu-mon --alarm-description "Alarm
when CPU exceeds 70 percent" --metric-name CPUUtilization --namespace AWS/
EC2 --statistic Average --period 300 --threshold 70 --comparison-operator
GreaterThanThreshold --dimensions "Name=InstanceId,Value=i-12345678" --evaluation-
periods 2 --alarm-actions arn:aws:sns:us-east-1:111122223333:MyTopic --unit Percent

```

如果成功，该命令将返回到提示符。如果已存在同名警报，则该警报将被新警报覆盖。

### 指定多个维度

以下示例说明了如何指定多个维度。每个维度都指定为一个“名称/值”对，名称和值之间用逗号隔开。多个维度之间用空格隔开：

```

aws cloudwatch put-metric-alarm --alarm-name "Default_Test_Alarm3" --alarm-
description "The default example alarm" --namespace "CW EXAMPLE METRICS" --

```

```
metric-name Default_Test --statistic Average --period 60 --evaluation-periods 3
--threshold 50 --comparison-operator GreaterThanOrEqualToThreshold --dimensions
Name=key1,Value=value1 Name=key2,Value=value2
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutMetricAlarm](#)中的。

## put-metric-data

以下代码示例演示了如何使用 put-metric-data。

### AWS CLI

向 Amazon 发布自定义指标 CloudWatch

以下示例使用put-metric-data命令向 Amazon 发布自定义指标 CloudWatch：

```
aws cloudwatch put-metric-data --namespace "Usage Metrics" --metric-data file://
metric.json
```

指标本身的值存储在 JSON 文件 metric.json 中。

以下是该文件的内容：

```
[
  {
    "MetricName": "New Posts",
    "Timestamp": "Wednesday, June 12, 2013 8:28:20 PM",
    "Value": 0.50,
    "Unit": "Count"
  }
]
```

有关更多信息，请参阅《Amazon CloudWatch 开发者指南》中的发布自定义指标。

### 指定多个维度

以下示例说明了如何指定多个维度。每个维度都指定为一个 Name=Value 对。多个维度之间用逗号隔开：

```
aws cloudwatch put-metric-data --metric-name Buffers --namespace MyNameSpace --unit
Bytes --value 231434333 --dimensions InstanceID=1-23456789,InstanceType=m1.small
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutMetricData](#)中的。

## set-alarm-state

以下代码示例演示了如何使用 `set-alarm-state`。

### AWS CLI

#### 临时更改警报的状态

以下示例使用 `set-alarm-state` 命令临时更改名为 “myalarm” 的 Amazon CloudWatch 警报的状态，并将其设置为警报状态以供测试：

```
aws cloudwatch set-alarm-state --alarm-name "myalarm" --state-value ALARM --state-reason "testing purposes"
```

如果成功，该命令将返回到提示符。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetAlarmState](#)中的。

## CloudWatch 使用记录示例 AWS CLI

以下代码示例向您展示了如何使用 `with Logs` 来执行操作和实现常见场 CloudWatch 景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

## create-log-group

以下代码示例演示了如何使用 `create-log-group`。

## AWS CLI

以下命令将创建名为 my-logs 的日志组：

```
aws logs create-log-group --log-group-name my-logs
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLogGroup](#)中的。

### **create-log-stream**

以下代码示例演示了如何使用 create-log-stream。

## AWS CLI

以下命令将在日志组 my-logs 中创建一个名为 20150601 的日志流：

```
aws logs create-log-stream --log-group-name my-logs --log-stream-name 20150601
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLogStream](#)中的。

### **delete-log-group**

以下代码示例演示了如何使用 delete-log-group。

## AWS CLI

以下命令将删除名为 my-logs 的日志组：

```
aws logs delete-log-group --log-group-name my-logs
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLogGroup](#)中的。

### **delete-log-stream**

以下代码示例演示了如何使用 delete-log-stream。

## AWS CLI

以下命令20150531从名为的日志组中删除名为的日志流my-logs：

```
aws logs delete-log-stream --log-group-name my-logs --log-stream-name 20150531
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLogStream](#)中的。

## delete-retention-policy

以下代码示例演示了如何使用 delete-retention-policy。

### AWS CLI

以下命令删除先前应用于名为的日志组的保留策略my-logs：

```
aws logs delete-retention-policy --log-group-name my-logs
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRetentionPolicy](#)中的。

## describe-log-groups

以下代码示例演示了如何使用 describe-log-groups。

### AWS CLI

以下命令将描述名为 my-logs 的日志组：

```
aws logs describe-log-groups --log-group-name-prefix my-logs
```

输出：

```
{
  "logGroups": [
    {
      "storedBytes": 0,
      "metricFilterCount": 0,
      "creationTime": 1433189500783,
      "logGroupName": "my-logs",
      "retentionInDays": 5,
      "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:*"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLogGroups](#)中的。



## describe-log-streams

以下代码示例演示了如何使用 `describe-log-streams`。

### AWS CLI

以下命令显示日志组 `2015` 中以该前缀开头的日志流 `my-logs`：

```
aws logs describe-log-streams --log-group-name my-logs --log-stream-name-prefix 2015
```

输出：

```
{
  "logStreams": [
    {
      "creationTime": 1433189871774,
      "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-stream:20150531",
      "logStreamName": "20150531",
      "storedBytes": 0
    },
    {
      "creationTime": 1433189873898,
      "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-stream:20150601",
      "logStreamName": "20150601",
      "storedBytes": 0
    }
  ]
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeLogStreams](#) 中的。

## get-log-events

以下代码示例演示了如何使用 `get-log-events`。

### AWS CLI

以下命令从日志组 `my-logs` 中命名的日志流 `20150601` 中检索日志事件：

```
aws logs get-log-events --log-group-name my-logs --log-stream-name 20150601
```

输出：

```
{
  "nextForwardToken":
  "f/31961209122447488583055879464742346735121166569214640130",
  "events": [
    {
      "ingestionTime": 1433190494190,
      "timestamp": 1433190184356,
      "message": "Example Event 1"
    },
    {
      "ingestionTime": 1433190516679,
      "timestamp": 1433190184356,
      "message": "Example Event 1"
    },
    {
      "ingestionTime": 1433190494190,
      "timestamp": 1433190184358,
      "message": "Example Event 2"
    }
  ],
  "nextBackwardToken":
  "b/31961209122358285602261756944988674324553373268216709120"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLogEvents](#)中的。

## put-log-events

以下代码示例演示了如何使用 put-log-events。

### AWS CLI

以下命令将日志事件放入日志组20150601中名为的日志流my-logs：

```
aws logs put-log-events --log-group-name my-logs --log-stream-name 20150601 --log-events file://events
```

输出：

```
{
```

```
"nextSequenceToken": "49542672486831074009579604567656788214806863282469607346"
}
```

上面的示例从当前目录中名为的文件events中读取一个 JSON 事件数组：

```
[
  {
    "timestamp": 1433190184356,
    "message": "Example Event 1"
  },
  {
    "timestamp": 1433190184358,
    "message": "Example Event 2"
  },
  {
    "timestamp": 1433190184360,
    "message": "Example Event 3"
  }
]
```

每个后续调用都需要使用序列令牌选项指定上一个调用提供的下一个序列标记：

```
aws logs put-log-events --log-group-name my-logs --log-stream-
name 20150601 --log-events file://events2 --sequence-token
"49542672486831074009579604567656788214806863282469607346"
```

输出：

```
{
  "nextSequenceToken": "49542672486831074009579604567900991230369019956308219826"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutLogEvents](#)中的。

## put-retention-policy

以下代码示例演示了如何使用 put-retention-policy。

### AWS CLI

以下命令向名为的日志组添加 5 天保留策略my-logs：

```
aws logs put-retention-policy --log-group-name my-logs --retention-in-days 5
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutRetentionPolicy](#)中的。

## CloudWatch 使用网络监控示例 AWS CLI

以下代码示例向您展示了如何使用 CloudWatch 网络监控来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### create-probe

以下代码示例演示了如何使用 create-probe。

AWS CLI

示例 1：创建使用 TCP 的探测器并将其添加到网络监视器中

以下create-probe示例创建了一个使用探测器，TCPprotocol并将该探测器添加到名为的监视器中Example\_NetworkMonitor。创建后，带有探测state器的监视器将PENDING一直持续到监视器创建为止ACTIVE。这可能需要几分钟，此时状态将更改为ACTIVE，然后您就可以开始查看CloudWatch 指标了。

```
aws networkmonitor create-probe \  
  --monitor-name Example_NetworkMonitor \  
  --probe sourceArn=arn:aws:ec2:region:111122223333:subnet/subnet-  
id,destination=10.0.0.100,destinationPort=80,protocol=TCP,packetSize=56,tags={Name=Probe1}
```

输出：

```
{
  "probeId": "probe-12345",
  "probeArn": "arn:aws:networkmonitor:region:111122223333:probe/probe-12345",
  "destination": "10.0.0.100",
  "destinationPort": 80,
  "packetSize": 56,
  "addressFamily": "IPV4",
  "vpcId": "vpc-12345",
  "state": "PENDING",
  "createdAt": "2024-03-29T12:41:57.314000-04:00",
  "modifiedAt": "2024-03-29T12:41:57.314000-04:00",
  "tags": {
    "Name": "Probe1"
  }
}
```

示例 2：使用 ICMP 创建使用探测器的探测器并将其添加到网络监视器中

以下create-probe示例创建了一个使用探测器，ICMPprotocol并将该探测器添加到名为的监视器中Example\_NetworkMonitor。创建后，带有探测state器的监视器将PENDING一直持续到监视器创建为止ACTIVE。这可能需要几分钟，此时状态将更改为ACTIVE，然后您就可以开始查看CloudWatch 指标了。

```
aws networkmonitor create-probe \
  --monitor-name Example_NetworkMonitor \
  --probe sourceArn=arn:aws:ec2:region:012345678910:subnet/subnet-
id,destination=10.0.0.100,protocol=ICMP,packetSize=56,tags={Name=Probe1}
```

输出：

```
{
  "probeId": "probe-12345",
  "probeArn": "arn:aws:networkmonitor:region:111122223333:probe/probe-12345",
  "destination": "10.0.0.100",
  "packetSize": 56,
  "addressFamily": "IPV4",
  "vpcId": "vpc-12345",
  "state": "PENDING",
  "createdAt": "2024-03-29T12:44:02.452000-04:00",
  "modifiedAt": "2024-03-29T12:44:02.452000-04:00",
}
```

```
"tags": {
  "Name": "Probe1"
}
```

有关更多信息，请参阅[亚马逊 CloudWatch 用户指南中的亚马逊 CloudWatch 网络监控器的工作原理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateProbe](#)中的。

## delete-monitor

以下代码示例演示了如何使用 delete-monitor。

### AWS CLI

#### 删除显示器

以下delete-monitor示例删除名为的监视器Example\_NetworkMonitor。

```
aws networkmonitor delete-monitor \  
  --monitor-name Example_NetworkMonitor
```

此命令不生成任何输出。

有关更多信息，请参阅[亚马逊 CloudWatch 用户指南中的亚马逊 CloudWatch 网络监控器的工作原理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMonitor](#)中的。

## delete-probe

以下代码示例演示了如何使用 delete-probe。

### AWS CLI

#### 删除探测器

以下delete-probe示例probe-12345从名为的网络监视器中删除 ID 为的探测器Example\_NetworkMonitor。

```
aws networkmonitor delete-probe \  
  --monitor-name Example_NetworkMonitor
```

```
--monitor-name Example_NetworkMonitor \  
--probe-id probe-12345
```

此命令不生成任何输出。

有关更多信息，请参阅[亚马逊 CloudWatch 用户指南中的亚马逊 CloudWatch 网络监控器的工作原理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteProbe](#)中的。

## get-probe

以下代码示例演示了如何使用 get-probe。

### AWS CLI

#### 查看探测器详情

以下get-probe示例返回有关探测器的详细信息，probeIDprobe-12345该探测器与名为的监视器相关联Example\_NetworkMonitor。

```
aws networkmonitor get-probe \  
  --monitor-name Example_NetworkMonitor \  
  --probe-id probe-12345
```

输出：

```
{  
  "probeId": "probe-12345",  
  "probeArn": "arn:aws:networkmonitor:region:012345678910:probe/probe-12345",  
  "sourceArn": "arn:aws:ec2:region:012345678910:subnet/subnet-12345",  
  "destination": "10.0.0.100",  
  "destinationPort": 80,  
  "protocol": "TCP",  
  "packetSize": 56,  
  "addressFamily": "IPV4",  
  "vpcId": "vpc-12345",  
  "state": "ACTIVE",  
  "createdAt": "2024-03-29T12:41:57.314000-04:00",  
  "modifiedAt": "2024-03-29T12:42:28.610000-04:00",  
  "tags": {  
    "Name": "Probe1"  
  }  
}
```

```
}  
}
```

有关更多信息，请参阅[亚马逊 CloudWatch 用户指南中的亚马逊 CloudWatch 网络监控器的工作原理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetProbe](#)中的。

## list-monitors

以下代码示例演示了如何使用 list-monitors。

### AWS CLI

示例 1：列出所有显示器（单个显示器）

以下list-monitors示例仅返回一个显示器的列表。显示器state是ACTIVE，它有 60 秒aggregationPeriod的时间。

```
aws networkmonitor list-monitors
```

输出：

```
{  
  "monitors": [{  
    "monitorArn": "arn:aws:networkmonitor:region:012345678910:monitor/  
Example_NetworkMonitor",  
    "monitorName": "Example_NetworkMonitor",  
    "state": "ACTIVE",  
    "aggregationPeriod": 60,  
    "tags": {  
      "Monitor": "Monitor1"  
    }  
  }  
]  
}
```

有关更多信息，请参阅[亚马逊 CloudWatch 用户指南中的亚马逊 CloudWatch 网络监控器的工作原理](#)。

示例 2：列出所有显示器（多台显示器）



以下list-monitors示例返回三台显示器的列表。state一台显示器中的一台是ACTIVE并生成CloudWatch 指标。另外两个监视器的状态为INACTIVE且未生成 CloudWatch 指标。所有三台显示器的使用时间均为 aggregationPeriod 60 秒。

```
aws networkmonitor list-monitors
```

输出：

```
{
  "monitors": [
    {
      "monitorArn": "arn:aws:networkmonitor:us-east-1:111122223333:monitor/Example_NetworkMonitor",
      "monitorName": "Example_NetworkMonitor",
      "state": "INACTIVE",
      "aggregationPeriod": 60,
      "tags": {}
    },
    {
      "monitorArn": "arn:aws:networkmonitor:us-east-1:111122223333:monitor/Example_NetworkMonitor2",
      "monitorName": "Example_NetworkMonitor2",
      "state": "ACTIVE",
      "aggregationPeriod": 60,
      "tags": {
        "Monitor": "Monitor1"
      }
    },
    {
      "monitorArn": "arn:aws:networkmonitor:us-east-1:111122223333:monitor/TestNetworkMonitor_CLI",
      "monitorName": "TestNetworkMonitor_CLI",
      "state": "INACTIVE",
      "aggregationPeriod": 60,
      "tags": {}
    }
  ]
}
```

有关更多信息，请参阅[亚马逊 CloudWatch 用户指南中的亚马逊 CloudWatch 网络监控器的工作原理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListMonitors](#)中的。

## update-monitor

以下代码示例演示了如何使用 update-monitor。

### AWS CLI

#### 更新显示器

以下update-monitor示例将显示器aggregationPeriod从60秒更改为30秒。

```
aws networkmonitor update-monitor \  
  --monitor-name Example_NetworkMonitor \  
  --aggregation-period 30
```

输出：

```
{  
  "monitorArn": "arn:aws:networkmonitor:region:012345678910:monitor/  
Example_NetworkMonitor",  
  "monitorName": "Example_NetworkMonitor",  
  "state": "PENDING",  
  "aggregationPeriod": 30,  
  "tags": {  
    "Monitor": "Monitor1"  
  }  
}
```

有关更多信息，请参阅[亚马逊 CloudWatch 用户指南中的亚马逊 CloudWatch 网络监控器的工作原理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateMonitor](#)中的。

## update-probe

以下代码示例演示了如何使用 update-probe。

### AWS CLI

#### 更新探测器

以下update-probe示例更新探测器的原始 destination IP 地址，并将更新packetSize为60。

```
aws networkmonitor update-probe \  
  --monitor-name Example_NetworkMonitor \  
  --probe-id probe-12345 \  
  --destination 10.0.0.150 \  
  --packet-size 60
```

输出：

```
{  
  "probeId": "probe-12345",  
  "probeArn": "arn:aws:networkmonitor:region:012345678910:probe/probe-12345",  
  "sourceArn": "arn:aws:ec2:region:012345678910:subnet/subnet-12345",  
  "destination": "10.0.0.150",  
  "destinationPort": 80,  
  "protocol": "TCP",  
  "packetSize": 60,  
  "addressFamily": "IPV4",  
  "vpcId": "vpc-12345",  
  "state": "PENDING",  
  "createdAt": "2024-03-29T12:41:57.314000-04:00",  
  "modifiedAt": "2024-03-29T13:52:23.115000-04:00",  
  "tags": {  
    "Name": "Probe1"  
  }  
}
```

有关更多信息，请参阅[亚马逊 CloudWatch 用户指南中的亚马逊 CloudWatch 网络监控器的工作原理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateProbe](#)中的。

## CodeArtifact 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 CodeArtifact。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### **associate-external-connection**

以下代码示例演示了如何使用 `associate-external-connection`。

#### AWS CLI

##### 向存储库添加外部连接

以下 `associate-external-connection` 示例将 `npmjs.com` 的外部连接添加到名为 `test-repo` 的存储库中。

```
aws codeartifact associate-external-connection \  
  --repository test-repo \  
  --domain test-domain \  
  --external-connection public:npmjs
```

##### 输出：

```
{  
  "repository": {  
    "name": "test-repo",  
    "administratorAccount": "111122223333",  
    "domainName": "test-domain",  
    "domainOwner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/test-domain/  
test-repo",  
    "upstreams": [],  
    "externalConnections": [  
      {  
        "externalConnectionName": "public:npmjs",  
        "packageFormat": "npm",  
        "status": "AVAILABLE"  
      }  
    ]  
  }  
}
```

```
}  
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[添加外部连接](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateExternalConnection](#) 中的。

## copy-package-versions

以下代码示例演示了如何使用 `copy-package-versions`。

### AWS CLI

将软件包版本从一个存储库复制到另一个存储库

以下内容 `copy-package-versions` 将名为 `test-package` 的软件包的 4.0.0 和 5.0.0 版本从 `my-repo` 移到了测试存储库。

```
aws codeartifact copy-package-versions \  
  --domain test-domain \  
  --source-repository my-repo \  
  --destination-repository test-repo \  
  --format npm \  
  --package test-package \  
  --versions '["4.0.0", "5.0.0"]'
```

输出：

```
{  
  "format": "npm",  
  "package": "test-package",  
  "versions": [  
    {  
      "version": "5.0.0",  
      "revision": "REVISION-1-SAMPLE-6C81EFF7DA55CC",  
      "status": "Published"  
    },  
    {  
      "version": "4.0.0",  
      "revision": "REVISION-2-SAMPLE-55C752BEE772FC",  
      "status": "Published"  
    }  
  ]  
}
```

```
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的在[仓库之间复制软件包](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CopyPackageVersions](#)中的。

## create-domain

以下代码示例演示了如何使用 create-domain。

### AWS CLI

#### 创建域名

以下create-domain示例创建了一个名为 test-domain 的域。

```
aws codeartifact create-domain \  
  --domain test-domain
```

输出：

```
{  
  "domain": {  
    "name": "test-domain",  
    "owner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:domain/test-domain",  
    "status": "Active",  
    "createdTime": "2020-10-20T13:16:48.559000-04:00",  
    "encryptionKey": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-  
cdef-EXAMPLE11111",  
    "repositoryCount": 0,  
    "assetSizeBytes": 0  
  }  
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[创建域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDomain](#)中的。

## create-repository

以下代码示例演示了如何使用 create-repository。

## AWS CLI

### 创建存储库

以下create-repository示例在名为 test-domain 的域中创建一个名为 test-repo 的存储库。

```
aws codeartifact create-repository \  
  --domain test-domain \  
  --domain-owner 111122223333 \  
  --repository test-repo \  
  --description "This is a test repository."
```

输出：

```
{  
  "repository": {  
    "name": "test-repo",  
    "administratorAccount": "111122223333",  
    "domainName": "test-domain",  
    "domainOwner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/test-domain/  
test-repo",  
    "description": "This is a test repository.",  
    "upstreams": [],  
    "externalConnections": []  
  }  
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[创建域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRepository](#)中的。

### delete-domain-permissions-policy

以下代码示例演示了如何使用 delete-domain-permissions-policy。

## AWS CLI

### 从域中删除权限策略文档

以下delete-domain-permissions-policy示例从名为 test-domain 的域中删除权限策略。

```
aws codeartifact delete-domain-permissions-policy \  
  --domain test-domain \  
  --policy-name test-policy
```

```
--domain test-domain
```

输出：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "BasicDomainPolicy",
      "Action": [
        "codeartifact:GetDomainPermissionsPolicy",
        "codeartifact:ListRepositoriesInDomain",
        "codeartifact:GetAuthorizationToken",
        "codeartifact:CreateRepository"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      }
    }
  ]
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[删除域名策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDomainPermissionsPolicy](#)中的。

## delete-domain

以下代码示例演示了如何使用 delete-domain。

### AWS CLI

#### 删除域名

以下delete-domain示例删除名为的域test-domain。

```
aws codeartifact delete-domain \
  --domain test-domain
```

输出：



```
{
  "domain": {
    "name": "test-domain",
    "owner": "417498243647",
    "arn": "arn:aws:codeartifact:us-west-2:417498243647:domain/test-domain",
    "status": "Deleted",
    "createdTime": "2020-10-20T13:16:48.559000-04:00",
    "encryptionKey": "arn:aws:kms:us-west-2:417498243647:key/c9fe2447-0795-4fda-afbe-8464574ae162",
    "repositoryCount": 0,
    "assetSizeBytes": 0
  }
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[删除域名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDomain](#)中的。

## delete-package-versions

以下代码示例演示了如何使用 delete-package-versions。

### AWS CLI

#### 删除软件包版本

以下delete-package-versions示例删除名为 test-package 的软件包的 4.0.0 版。

```
aws codeartifact delete-package-versions \
  --domain test-domain \
  --repo test-repo \
  --format npm \
  --package test-package \
  --versions 4.0.0
```

输出：

```
{
  "successfulVersions": {
    "4.0.0": {
      "revision": "Ciqe5/9yicvkJT13b5/LdLpCyE6fqA7poa9qp+FilPs=",
```

```

        "status": "Deleted"
      }
    },
    "failedVersions": {}
  }

```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的“[删除软件包版本](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeletePackageVersions](#) 中的。

## delete-repository-permissions-policy

以下代码示例演示了如何使用 delete-repository-permissions-policy。

### AWS CLI

从存储库中删除权限策略

以下 delete-repository-permissions-policy 示例从名为 test-repo 的存储库中删除权限策略。

```

aws codeartifact delete-repository-permissions-policy \
  --domain test-domain \
  --repository test-repo

```

输出：

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": [
        "codeartifact:DescribePackageVersion",
        "codeartifact:DescribeRepository",
        "codeartifact:GetPackageVersionReadme",
        "codeartifact:GetRepositoryEndpoint",
        "codeartifact:ListPackages",
        "codeartifact:ListPackageVersions",

```

```

        "codeartifact:ListPackageVersionAssets",
        "codeartifact:ListPackageVersionDependencies",
        "codeartifact:ReadFromRepository"
    ],
    "Resource": "*"
}
]
}

```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[删除策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteRepositoryPermissionsPolicy](#) 中的。

## delete-repository

以下代码示例演示了如何使用 delete-repository。

### AWS CLI

#### 删除存储库

以下 delete-repository 示例删除名为的域 test-repo 中名为的存储库 test-domain。

```

aws codeartifact delete-repository \
  --domain test-domain \
  --repository test-repo

```

输出：

```

{
  "repository": {
    "name": "test-repo",
    "administratorAccount": "111122223333",
    "domainName": "test-domain",
    "domainOwner": "111122223333",
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/test-domain/test-repo",
    "description": "This is a test repository",
    "upstreams": [],
    "externalConnections": []
  }
}

```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[删除仓库](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRepository](#)中的。

## describe-domain

以下代码示例演示了如何使用 describe-domain。

### AWS CLI

获取有关域名的信息

以下describe-domain示例返回名为 DomainDescription test-domain 的域的对象。

```
aws codeartifact describe-domain \  
  --domain test-domain
```

输出：

```
{  
  "domain": {  
    "name": "test-domain",  
    "owner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:domain/test-domain",  
    "status": "Active",  
    "createdTime": "2020-10-20T13:16:48.559000-04:00",  
    "encryptionKey": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-  
cdef-EXAMPLE11111",  
    "repositoryCount": 2,  
    "assetSizeBytes": 0,  
    "s3BucketArn": "arn:aws:s3:::assets-111122223333-us-west-2"  
  }  
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[域名概述](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDomain](#)中的。

## describe-repository

以下代码示例演示了如何使用 describe-repository。

## AWS CLI

获取有关存储库的信息

以下describe-repository示例返回名为 RepositoryDescription test-repo 的存储库的对象。

```
aws codeartifact describe-repository \  
  --domain test-domain \  
  --repository test-repo
```

输出：

```
{  
  "repository": {  
    "name": "test-repo",  
    "administratorAccount": "111122223333",  
    "domainName": "test-domain",  
    "domainOwner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/test-domain/  
test-repo",  
    "description": "This is a test repository.",  
    "upstreams": [],  
    "externalConnections": []  
  }  
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[创建域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeRepository](#)中的。

## disassociate-external-connection

以下代码示例演示了如何使用 disassociate-external-connection。

### AWS CLI

从存储库中移除外部连接

以下disassociate-external-connection示例从名为 test-repo 的存储库中删除了与 npmjs.com 的外部连接。

```
aws codeartifact disassociate-external-connection \  
  --domain test-domain \  
  --repository test-repo
```

```
--repository test-repo \  
--domain test-domain \  
--external-connection public:npmjs
```

输出：

```
{  
  "repository": {  
    "name": "test-repo",  
    "administratorAccount": "111122223333",  
    "domainName": "test-domain",  
    "domainOwner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/test-domain/  
test-repo",  
    "upstreams": [],  
    "externalConnections": []  
  }  
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[移除外部连接](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateExternalConnection](#) 中的。

## dispose-package-versions

以下代码示例演示了如何使用 `dispose-package-versions`。

AWS CLI

删除软件包版本的资产并将其状态设置为“已处置”

以下 `dispose-package-versions` 示例删除了测试包版本 4.0.0 的资源并将其状态设置为“已处置”。

```
aws codeartifact dispose-package-versions \  
--domain test-domain \  
--repo test-repo \  
--format npm \  
--package test-package \  
--versions 4.0.0
```

输出：

```
{
  "successfulVersions": {
    "4.0.0": {
      "revision": "Ciqe5/9yicvkJT13b5/LdLpCyE6fqA7poa9qp+FilPs=",
      "status": "Disposed"
    }
  },
  "failedVersions": {}
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》CodeArtifact 中的“使用[软件包](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisposePackageVersions](#) 中的。

## get-authorization-token

以下代码示例演示了如何使用 get-authorization-token。

### AWS CLI

获取授权令牌

以下 get-authorization-token 示例检索 CodeArtifact 授权令牌。

```
aws codeartifact get-authorization-token \
  --domain test-domain \
  --query authorizationToken \
  --output text
```

输出：

```
This command will return the authorization token. You can store the output in an
environment variable when calling the command.
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[不使用 login 命令配置 pip](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetAuthorizationToken](#) 中的。

## get-domain-permissions-policy

以下代码示例演示了如何使用 get-domain-permissions-policy。

## AWS CLI

### 获取域的权限策略文档

以下`get-domain-permissions-policy`示例获取了附加到名为 `test-domain` 的域的权限策略。

```
aws codeartifact get-domain-permissions-policy \  
  --domain test-domain
```

输出：

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "BasicDomainPolicy",  
      "Action": [  
        "codeartifact:GetDomainPermissionsPolicy",  
        "codeartifact:ListRepositoriesInDomain",  
        "codeartifact:GetAuthorizationToken",  
        "codeartifact:CreateRepository"  
      ],  
      "Effect": "Allow",  
      "Resource": "*",  
      "Principal": {  
        "AWS": "arn:aws:iam::111122223333:root"  
      }  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[阅读域名政策](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDomainPermissionsPolicy](#)中的。

## get-package-version-asset

以下代码示例演示了如何使用 `get-package-version-asset`。

## AWS CLI

### 从包版本中获取资产



以下`get-package-version-asset`示例检索名为 `test-package` 的 npm 软件包的 4.0.0 版 `package.tgz` 资产。

```
aws codeartifact get-package-version-asset \  
  --domain test-domain \  
  --repository test-repo \  
  --format npm \  
  --package test-package \  
  --package-version 4.0.0 \  
  --asset 'package.tgz' \  
  outfileName
```

输出：

The output for this command will also store the raw asset in the file provided in place of `outfileName`.

```
{  
  "assetName": "package.tgz",  
  "packageVersion": "4.0.0",  
  "packageVersionRevision": "Ciqe5/9yicvkJT13b5/LdLpCyE6fqA7poa9qp+FilPs="
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[列出软件包版本资产](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetPackageVersionAsset](#) 中的。

## get-package-version-readme

以下代码示例演示了如何使用 `get-package-version-readme`。

### AWS CLI

获取软件包版本的自述文件

以下`get-package-version-readme`示例检索名为 `test-package` 的 npm 软件包版本 4.0.0 的自述文件。

```
aws codeartifact get-package-version-readme \  
  --domain test-domain \  
  --repo test-repo \  
  --package test-package
```

```
--format npm \  
--package test-package \  
--package-version 4.0.0
```

输出：

```
{  
  "format": "npm",  
  "package": "test-package",  
  "version": "4.0.0",  
  "readme": "<div align=\"center\">\n  <a href=\"https://github.com/test-package/  
testpack\"> ... more content ... \n",  
  "versionRevision": "Ciqe5/9yicvkJT13b5/LdLpCyE6fqA7poa9qp+FilPs="  
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[“查看软件包版本自述文件”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetPackageVersionReadme](#)中的。

## get-repository-endpoint

以下代码示例演示了如何使用 `get-repository-endpoint`。

### AWS CLI

获取仓库的 URL 端点

以下 `get-repository-endpoint` 示例返回测试存储库的 npm 端点。

```
aws codeartifact get-repository-endpoint \  
  --domain test-domain \  
  --repository test-repo \  
  --format npm
```

输出：

```
{  
  "repositoryEndpoint": "https://test-domain-111122223333.d.codeartifact.us-  
west-2.amazonaws.com/npm/test-repo/"  
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的 [Connect 到存储库](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetRepositoryEndpoint](#) 中的。

## get-repository-permissions-policy

以下代码示例演示了如何使用 `get-repository-permissions-policy`。

### AWS CLI

获取存储库的权限策略文档

以下 `get-repository-permissions-policy` 示例将权限策略附加到名为 `test-repo` 的存储库。

```
aws codeartifact get-repository-permissions-policy \  
  --domain test-domain \  
  --repository test-repo
```

输出：

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::111122223333:root"  
      },  
      "Action": [  
        "codeartifact:DescribePackageVersion",  
        "codeartifact:DescribeRepository",  
        "codeartifact:GetPackageVersionReadme",  
        "codeartifact:GetRepositoryEndpoint",  
        "codeartifact:ListPackages",  
        "codeartifact:ListPackageVersions",  
        "codeartifact:ListPackageVersionAssets",  
        "codeartifact:ListPackageVersionDependencies",  
        "codeartifact:ReadFromRepository"  
      ],  
      "Resource": "*"   
    }  
  ]  
}
```

```
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[阅读策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRepositoryPermissionsPolicy](#)中的。

## list-domains

以下代码示例演示了如何使用 list-domains。

### AWS CLI

#### 列出域名

以下list-domains示例返回发起呼叫的 AWS 账户拥有的所有域名的摘要。

```
aws codeartifact list-domains
```

输出：

```
{
  "domains": [
    {
      "name": "my-domain",
      "owner": "111122223333",
      "status": "Active",
      "encryptionKey": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    },
    {
      "name": "test-domain",
      "owner": "111122223333",
      "status": "Active",
      "encryptionKey": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
    }
  ]
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》CodeArtifact中的“使用[域名](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDomains](#)中的。

## list-package-version-assets

以下代码示例演示了如何使用 `list-package-version-assets`。

### AWS CLI

查看软件包版本的资产

以下 `list-package-version-assets` 示例检索名为 `test-package` 的 npm 软件包的 4.0.0 版资产。

```
aws codeartifact list-package-version-assets \
  --domain test-domain \
  --repo test-repo \
  --format npm \
  --package test-package \
  --package-version 4.0.0
```

输出：

```
{
  "format": "npm",
  "package": "test-package",
  "version": "4.0.0",
  "versionRevision": "Ciqe5/9yicvkJT13b5/LdLpCyE6fqA7poa9qp+FilPs=",
  "assets": [
    {
      "name": "package.tgz",
      "size": 316680,
      "hashes": {
        "MD5": "60078ec6d9e76b89fb55c860832742b2",
        "SHA-1": "b44a9b6297bcb698f1c51a3545a2b3b368d59c52",
        "SHA-256":
          "d2aa8c6afc3c8591765785a37d1c5acae482a8eb3ab9729ed28922692454f2e2",
        "SHA-512":
          "3e585d15c8a594e20d7de57b362ea81754c011acb2641a19f1b72c8531ea39825896bab344ae616a0a5a824cb9"
      }
    }
  ]
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[列出软件包版本资产](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListPackageVersionAssets](#) 中的。

## list-package-version-dependencies

以下代码示例演示了如何使用 `list-package-version-dependencies`。

### AWS CLI

查看软件包版本的依赖关系

以下 `list-package-version-dependencies` 示例检索名为 `test-package` 的 npm 软件包版本 4.0.0 的依赖关系。

```
aws codeartifact list-package-version-dependencies \  
  --domain test-domain \  
  --repo test-repo \  
  --format npm \  
  --package test-package \  
  --package-version 4.0.0
```

输出：

```
{  
  "format": "npm",  
  "package": "test-package",  
  "version": "4.0.0",  
  "versionRevision": "Ciqe5/9yicvkJT13b5/LdLpCyE6fqA7poa9qp+FilPs=",  
  "dependencies": [  
    {  
      "namespace": "testns",  
      "package": "testdep1",  
      "dependencyType": "regular",  
      "versionRequirement": "1.8.5"  
    },  
    {  
      "namespace": "testns",  
      "package": "testdep2",  
      "dependencyType": "regular",  
      "versionRequirement": "1.8.5"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[查看和更新软件包版本详细信息和依赖关系](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPackageVersionDependencies](#)中的。

## list-package-versions

以下代码示例演示了如何使用 `list-package-versions`。

### AWS CLI

列出软件包的软件包版本

以下`list-package-versions`示例返回名为的软件包的软件包版本列表`kind-of`。

```
aws codeartifact list-package-versions \  
  --package kind-of \  
  --domain test-domain \  
  --repository test-repo \  
  --format npm
```

输出：

```
{  
  "defaultDisplayVersion": "1.0.1",  
  "format": "npm",  
  "package": "kind-of",  
  "versions": [  
    {  
      "version": "1.0.1",  
      "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",  
      "status": "Published"  
    },  
    {  
      "version": "1.0.0",  
      "revision": "REVISION-SAMPLE-2-C752BEEF6D2CFC",  
      "status": "Published"  
    },  
    {  
      "version": "0.1.2",  
      "revision": "REVISION-SAMPLE-3-654S65A5C5E1FC",  
      "status": "Published"  
    },  
    {  
      "version": "0.1.1",
```

```
    "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
    "status": "Published"
  },
  {
    "version": "0.1.0",
    "revision": "REVISION-SAMPLE-4-AF669139B772FC",
    "status": "Published"
  }
]
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[列出软件包版本](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPackageVersions](#)中的。

## list-packages

以下代码示例演示了如何使用 list-packages。

### AWS CLI

列出存储库中的软件包

以下list-packages示例列出了名为的域中名为test-repo的存储库中的软件包test-domain。

```
aws codeartifact list-packages \
  --domain test-domain \
  --repository test-repo
```

输出：

```
{
  "packages": [
    {
      "format": "npm",
      "package": "lodash"
    }
    {
      "format": "python",
      "package": "test-package"
    }
  ]
}
```



```
]
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[列出软件包名称](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListPackages](#)中的。

## list-repositories-in-domain

以下代码示例演示了如何使用 `list-repositories-in-domain`。

### AWS CLI

列出域中的仓库

以下 `list-repositories-in-domain` 示例返回测试域中所有存储库的摘要。

```
aws codeartifact list-repositories-in-domain \
  --domain test-domain
```

输出：

```
{
  "repositories": [
    {
      "name": "test-repo",
      "administratorAccount": "111122223333",
      "domainName": "test-domain",
      "domainOwner": "111122223333",
      "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/test-
domain/test-repo",
      "description": "This is a test repository."
    },
    {
      "name": "test-repo2",
      "administratorAccount": "111122223333",
      "domainName": "test-domain",
      "domainOwner": "111122223333",
      "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/test-
domain/test-repo2",
      "description": "This is a test repository."
    }
  ]
}
```

```
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[列出仓库](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRepositoriesInDomain](#)中的。

## list-repositories

以下代码示例演示了如何使用 list-repositories。

### AWS CLI

#### 列出存储库

以下list-repositories示例返回进行调用的 AWS 账户所拥有的域中所有存储库的摘要。

```
aws codeartifact list-repositories
```

输出：

```
{
  "repositories": [
    {
      "name": "npm-store",
      "administratorAccount": "111122223333",
      "domainName": "my-domain",
      "domainOwner": "111122223333",
      "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/my-domain/npm-store",
      "description": "Provides npm artifacts from npm, Inc."
    },
    {
      "name": "target-repo",
      "administratorAccount": "111122223333",
      "domainName": "my-domain",
      "domainOwner": "111122223333",
      "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/my-domain/target-repo",
      "description": "test target repo"
    },
    {
      "name": "test-repo2",
      "administratorAccount": "111122223333",
```

```
        "domainName": "test-domain",
        "domainOwner": "111122223333",
        "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/test-
domain/test-repo2",
        "description": "This is a test repository."
    }
]
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[列出仓库](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListRepositories](#)中的。

## login

以下代码示例演示了如何使用 login。

### AWS CLI

使用 login 命令配置对存储库的身份验证

以下 login 示例在名为 test-domain 的域中使用名为 test-repo 的存储库来配置 npm 软件包管理器。

```
aws codeartifact login \
  --domain test-domain \
  --repository test-repo \
  --tool npm
```

输出：

```
Successfully configured npm to use AWS CodeArtifact repository https://test-
domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/npm/test-repo/
Login expires in 12 hours at 2020-11-12 01:53:16-05:00
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[AWS CLI 入门](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的[“登录”](#)。

## put-domain-permissions-policy

以下代码示例演示了如何使用 put-domain-permissions-policy。

## AWS CLI

### 将权限策略附加到网域

以下put-domain-permissions-policy示例将在 policy.json 文件中定义的权限策略附加到名为 test-domain 的域中。

```
aws codeartifact put-domain-permissions-policy \  
  --domain test-domain \  
  --policy-document file://PATH/T0/policy.json
```

输出：

```
{  
  "policy": {  
    "resourceArn": "arn:aws:codeartifact:region-id:111122223333:domain/test-domain",  
    "document": "{ ...policy document content...}",  
    "revision": "MQ1yyTQRASRU3HB58gBtSDHXG7Q3hvxxxxxxxxx="  }  
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[设置域名策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutDomainPermissionsPolicy](#)中的。

## put-repository-permissions-policy

以下代码示例演示了如何使用 put-repository-permissions-policy。

## AWS CLI

### 将权限策略附加到存储库

以下put-repository-permissions-policy示例将在 policy.json 文件中定义的权限策略附加到名为 test-repo 的存储库。

```
aws codeartifact put-repository-permissions-policy \  
  --domain test-domain \  
  --repository test-repo \  
  --policy-document file://PATH/T0/policy.json
```

输出：

```
{
  "policy": {
    "resourceArn": "arn:aws:codeartifact:region-id:111122223333:repository/test-domain/test-repo",
    "document": "{ ...policy document content...}",
    "revision": "MQlYyTQRASRU3HB58gBtSDHXG7Q3hvxxxxxxxx="
  }
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[设置策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[PutRepositoryPermissionsPolicy](#)中的。

## update-package-versions-status

以下代码示例演示了如何使用 `update-package-versions-status`。

### AWS CLI

#### 更新软件包版本状态

以下 `update-package-versions-status` 示例将测试包版本 4.0.0 的状态更新为已存档。

```
aws codeartifact update-package-versions-status \
  --domain test-domain \
  --repo test-repo \
  --format npm \
  --package test-package \
  --versions 4.0.0 \
  --target-status Archived
```

输出：

```
{
  "successfulVersions": {
    "4.0.0": {
      "revision": "Ciqe5/9yicvkJT13b5/LdLpCyE6fqA7poa9qp+FilPs=",
      "status": "Archived"
    }
  },
}
```

```
"failedVersions": {}  
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[“更新软件包版本状态”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePackageVersionsStatus](#)中的。

## update-repository

以下代码示例演示了如何使用 update-repository。

### AWS CLI

#### 更新存储库

以下update-repository示例将名为 test-domain 的域中名为 test-repo 的存储库的描述更新为“这是更新的描述”。

```
aws codeartifact update-repository \  
  --domain test-domain \  
  --repository test-repo \  
  --description "this is an updated description"
```

输出：

```
{  
  "repository": {  
    "name": "test-repo",  
    "administratorAccount": "111122223333",  
    "domainName": "test-domain",  
    "domainOwner": "111122223333",  
    "arn": "arn:aws:codeartifact:us-west-2:111122223333:repository/test-domain/  
test-repo",  
    "description": "this is an updated description",  
    "upstreams": [],  
    "externalConnections": []  
  }  
}
```

有关更多信息，请参阅《AWS CodeArtifact 用户指南》中的[查看或修改存储库配置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRepository](#)中的。

## CodeBuild 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 CodeBuild。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### batch-delete-builds

以下代码示例演示了如何使用 batch-delete-builds。

AWS CLI

要删除中的内部版本 AWS CodeBuild。

以下batch-delete-builds示例删除 CodeBuild 具有指定 ID 的内部版本。

```
aws codebuild batch-delete-builds --ids my-build-project-one:a1b2c3d4-5678-9012-abcd-11111EXAMPLE my-build-project-two:a1b2c3d4-5678-9012-abcd-22222EXAMPLE
```

输出：

```
{
  "buildsNotDeleted": [
    {
      "id": "arn:aws:codebuild:us-west-2:123456789012:build/my-build-project-one:a1b2c3d4-5678-9012-abcd-11111EXAMPLE",
      "statusCode": "BUILD_IN_PROGRESS"
    }
  ],
  "buildsDeleted": [
```

```
    "arn:aws:codebuild:us-west-2:123456789012:build/my-build-project-  
two:a1b2c3d4-5678-9012-abcd-22222EXAMPLE"  
  ]  
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[删除构建 \(AWS CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchDeleteBuilds](#) 中的。

## batch-get-build-batches

以下代码示例演示了如何使用 batch-get-build-batches。

### AWS CLI

查看内部版本的详细信息 AWS CodeBuild。

以下 batch-get-build-batches 示例获取有关 CodeBuild 使用指定 ID 编译批次的信息。

```
aws codebuild batch-get-build-batches \  
  --ids codebuild-demo-project:e9c4f4df-3f43-41d2-ab3a-60fe2EXAMPLE
```

输出：

```
{  
  "buildBatches": [  
    {  
      "id": "codebuild-demo-project:e9c4f4df-3f43-41d2-ab3a-60fe2EXAMPLE",  
      "arn": "arn:aws:codebuild:us-west-2:123456789012:build-batch/codebuild-  
demo-project:e9c4f4df-3f43-41d2-ab3a-60fe2EXAMPLE",  
      "startTime": "2020-11-03T21:52:20.775000+00:00",  
      "endTime": "2020-11-03T21:56:59.784000+00:00",  
      "currentPhase": "SUCCEEDED",  
      "buildBatchStatus": "SUCCEEDED",  
      "resolvedSourceVersion": "0a6546f68309560d08a310daac92314c4d378f6b",  
      "projectName": "codebuild-demo-project",  
      "phases": [  
        {  
          "phaseType": "SUBMITTED",  
          "phaseStatus": "SUCCEEDED",  
          "startTime": "2020-11-03T21:52:20.775000+00:00",  
          "endTime": "2020-11-03T21:52:20.976000+00:00",  
          "durationInSeconds": 0  
        }  
      ]  
    }  
  ]  
}
```



```
    },
    {
      "phaseType": "DOWNLOAD_BATCHSPEC",
      "phaseStatus": "SUCCEEDED",
      "startTime": "2020-11-03T21:52:20.976000+00:00",
      "endTime": "2020-11-03T21:52:57.401000+00:00",
      "durationInSeconds": 36
    },
    {
      "phaseType": "IN_PROGRESS",
      "phaseStatus": "SUCCEEDED",
      "startTime": "2020-11-03T21:52:57.401000+00:00",
      "endTime": "2020-11-03T21:56:59.751000+00:00",
      "durationInSeconds": 242
    },
    {
      "phaseType": "COMBINE_ARTIFACTS",
      "phaseStatus": "SUCCEEDED",
      "startTime": "2020-11-03T21:56:59.751000+00:00",
      "endTime": "2020-11-03T21:56:59.784000+00:00",
      "durationInSeconds": 0
    },
    {
      "phaseType": "SUCCEEDED",
      "startTime": "2020-11-03T21:56:59.784000+00:00"
    }
  ],
  "source": {
    "type": "GITHUB",
    "location": "https://github.com/my-repo/codebuild-demo-project.git",
    "gitCloneDepth": 1,
    "gitSubmodulesConfig": {
      "fetchSubmodules": false
    },
    "reportBuildStatus": false,
    "insecureSsl": false
  },
  "secondarySources": [],
  "secondarySourceVersions": [],
  "artifacts": {
    "location": ""
  },
  "secondaryArtifacts": [],
  "cache": {
```

```
    "type": "NO_CACHE"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/amazonlinux2-x86_64-standard:3.0",
    "computeType": "BUILD_GENERAL1_SMALL",
    "environmentVariables": [],
    "privilegedMode": false,
    "imagePullCredentialsType": "CODEBUILD"
  },
  "logConfig": {
    "cloudWatchLogs": {
      "status": "ENABLED"
    },
    "s3Logs": {
      "status": "DISABLED",
      "encryptionDisabled": false
    }
  },
  "buildTimeoutInMinutes": 60,
  "queuedTimeoutInMinutes": 480,
  "complete": true,
  "initiator": "Strohm",
  "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
  "buildBatchNumber": 6,
  "buildBatchConfig": {
    "serviceRole": "arn:aws:iam::123456789012:role/service-role/codebuild-demo-project",
    "restrictions": {
      "maximumBuildsAllowed": 100
    },
    "timeoutInMins": 480
  },
  "buildGroups": [
    {
      "identifier": "DOWNLOAD_SOURCE",
      "ignoreFailure": false,
      "currentBuildSummary": {
        "arn": "arn:aws:codebuild:us-west-2:123456789012:build/codebuild-demo-project:379737d8-bc35-48ec-97fd-776d27545315",
        "requestedOn": "2020-11-03T21:52:21.394000+00:00",
        "buildStatus": "SUCCEEDED",
        "primaryArtifact": {
          "type": "no_artifacts",
```

```
        "identifier": "DOWNLOAD_SOURCE"
      },
      "secondaryArtifacts": []
    }
  },
  {
    "identifier": "linux_small",
    "dependsOn": [],
    "ignoreFailure": false,
    "currentBuildSummary": {
      "arn": "arn:aws:codebuild:us-west-2:123456789012:build/
codebuild-demo-project:dd785171-ed84-4bb6-8ede-ceeb86e54bdb",
      "requestedOn": "2020-11-03T21:52:57.604000+00:00",
      "buildStatus": "SUCCEEDED",
      "primaryArtifact": {
        "type": "no_artifacts",
        "identifier": "linux_small"
      },
      "secondaryArtifacts": []
    }
  },
  {
    "identifier": "linux_medium",
    "dependsOn": [
      "linux_small"
    ],
    "ignoreFailure": false,
    "currentBuildSummary": {
      "arn": "arn:aws:codebuild:us-west-2:123456789012:build/
codebuild-demo-project:97cf7bd4-5313-4786-8243-4aef350a1267",
      "requestedOn": "2020-11-03T21:54:18.474000+00:00",
      "buildStatus": "SUCCEEDED",
      "primaryArtifact": {
        "type": "no_artifacts",
        "identifier": "linux_medium"
      },
      "secondaryArtifacts": []
    }
  },
  {
    "identifier": "linux_large",
    "dependsOn": [
      "linux_medium"
    ],
```

```

        "ignoreFailure": false,
        "currentBuildSummary": {
          "arn": "arn:aws:codebuild:us-west-2:123456789012:build/
codebuild-demo-project:60a194cd-0d03-4337-9db1-d41476a17d27",
          "requestedOn": "2020-11-03T21:55:39.203000+00:00",
          "buildStatus": "SUCCEEDED",
          "primaryArtifact": {
            "type": "no_artifacts",
            "identifier": "linux_large"
          },
          "secondaryArtifacts": []
        }
      ]
    ],
    "buildBatchesNotFound": []
  }
}

```

有关更多信息，请参阅《用户AWS CodeBuild 指南》AWS CodeBuild 中的 Batch building in。 <  
<https://docs.aws.amazon.com/codebuild/latest/userguide/batch-build.html>>

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetBuildBatches](#)中的。

## batch-get-builds

以下代码示例演示了如何使用 batch-get-builds。

### AWS CLI

查看内部版本的详细信息 AWS CodeBuild。

以下batch-get-builds示例获取有关 CodeBuild 具有指定 ID 的内部版本的信息。

```
aws codebuild batch-get-builds --ids codebuild-demo-project:e9c4f4df-3f43-41d2-
ab3a-60fe2EXAMPLE codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE
```

输出：

```

{
  "buildsNotFound": [],
  "builds": [
    {

```

```
    "artifacts": {
      "md5sum": "0e95edf915048a0c22efe6d139fff837",
      "location": "arn:aws:s3:::codepipeline-us-west-2-820783811474/CodeBuild-Python-Pip/BuildArtif/6DJsqQa",
      "encryptionDisabled": false,
      "sha256sum":
"cfaf0df33a090966a737f64ae4fe498969fdc842a0c9aec540bf93c37ac0d05a2"
    },
    "logs": {
      "cloudWatchLogs": {
        "status": "ENABLED"
      },
      "s3Logs": {
        "status": "DISABLED"
      },
      "streamName": "46472baf-8f6b-43c2-9255-b3b963af2732",
      "groupName": "/aws/codebuild/codebuild-demo-project",
      "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-west-2#logEvent:group=/aws/codebuild/codebuild-demo-project;stream=46472baf-8f6b-43c2-9255-b3b963af2732"
    },
    "timeoutInMinutes": 60,
    "environment": {
      "privilegedMode": false,
      "computeType": "BUILD_GENERAL1_MEDIUM",
      "image": "aws/codebuild/windows-base:1.0",
      "environmentVariables": [],
      "type": "WINDOWS_CONTAINER"
    },
    "projectName": "codebuild-demo-project",
    "buildComplete": true,
    "source": {
      "gitCloneDepth": 1,
      "insecureSsl": false,
      "type": "CODEPIPELINE"
    },
    "buildStatus": "SUCCEEDED",
    "secondaryArtifacts": [],
    "phases": [
      {
        "durationInSeconds": 0,
        "startTime": 1548717462.122,
        "phaseType": "SUBMITTED",
        "endTime": 1548717462.484,
```

```
    "phaseStatus": "SUCCEEDED"
  },
  {
    "durationInSeconds": 0,
    "startTime": 1548717462.484,
    "phaseType": "QUEUED",
    "endTime": 1548717462.775,
    "phaseStatus": "SUCCEEDED"
  },
  {
    "durationInSeconds": 34,
    "endTime": 1548717496.909,
    "contexts": [
      {
        "statusCode": "",
        "message": ""
      }
    ],
    "startTime": 1548717462.775,
    "phaseType": "PROVISIONING",
    "phaseStatus": "SUCCEEDED"
  },
  {
    "durationInSeconds": 15,
    "endTime": 1548717512.555,
    "contexts": [
      {
        "statusCode": "",
        "message": ""
      }
    ],
    "startTime": 1548717496.909,
    "phaseType": "DOWNLOAD_SOURCE",
    "phaseStatus": "SUCCEEDED"
  },
  {
    "durationInSeconds": 0,
    "endTime": 1548717512.734,
    "contexts": [
      {
        "statusCode": "",
        "message": ""
      }
    ]
  },
],
```

```
    "startTime": 1548717512.555,  
    "phaseType": "INSTALL",  
    "phaseStatus": "SUCCEEDED"  
  },  
  {  
    "durationInSeconds": 0,  
    "endTime": 1548717512.924,  
    "contexts": [  
      {  
        "statusCode": "",  
        "message": ""  
      }  
    ],  
    "startTime": 1548717512.734,  
    "phaseType": "PRE_BUILD",  
    "phaseStatus": "SUCCEEDED"  
  },  
  {  
    "durationInSeconds": 9,  
    "endTime": 1548717522.254,  
    "contexts": [  
      {  
        "statusCode": "",  
        "message": ""  
      }  
    ],  
    "startTime": 1548717512.924,  
    "phaseType": "BUILD",  
    "phaseStatus": "SUCCEEDED"  
  },  
  {  
    "durationInSeconds": 3,  
    "endTime": 1548717525.498,  
    "contexts": [  
      {  
        "statusCode": "",  
        "message": ""  
      }  
    ],  
    "startTime": 1548717522.254,  
    "phaseType": "POST_BUILD",  
    "phaseStatus": "SUCCEEDED"  
  },  
  {
```

```
        "durationInSeconds": 9,
        "endTime": 1548717534.646,
        "contexts": [
            {
                "statusCode": "",
                "message": ""
            }
        ],
        "startTime": 1548717525.498,
        "phaseType": "UPLOAD_ARTIFACTS",
        "phaseStatus": "SUCCEEDED"
    },
    {
        "durationInSeconds": 2,
        "endTime": 1548717536.846,
        "contexts": [
            {
                "statusCode": "",
                "message": ""
            }
        ],
        "startTime": 1548717534.646,
        "phaseType": "FINALIZING",
        "phaseStatus": "SUCCEEDED"
    },
    {
        "startTime": 1548717536.846,
        "phaseType": "COMPLETED"
    }
],
"startTime": 1548717462.122,
"encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
"initiator": "codepipeline/CodeBuild-Pipeline",
"secondarySources": [],
"serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-service-role",
"currentPhase": "COMPLETED",
"id": "codebuild-demo-project:e9c4f4df-3f43-41d2-ab3a-60fe2EXAMPLE",
"cache": {
    "type": "NO_CACHE"
},
"sourceVersion": "arn:aws:s3:::codepipeline-us-west-2-820783811474/CodeBuild-Python-Pip/SourceArti/1TspnN3.zip",
"endTime": 1548717536.846,
```



```
    "arn": "arn:aws:codebuild:us-west-2:123456789012:build/codebuild-demo-
project:e9c4f4df-3f43-41d2-ab3a-60fe2EXAMPLE",
    "queuedTimeoutInMinutes": 480,
    "resolvedSourceVersion": "f2194c1757bbdcb0f8f229254a4b3c8b27d43e0b"
  },
  {
    "artifacts": {
      "md5sum": "",
      "overrideArtifactName": false,
      "location": "arn:aws:s3:::my-artifacts/codebuild-demo-project",
      "encryptionDisabled": false,
      "sha256sum": ""
    },
    "logs": {
      "cloudWatchLogs": {
        "status": "ENABLED"
      },
      "s3Logs": {
        "status": "DISABLED"
      },
      "streamName": "4dea3ca4-20ec-4898-b22a-a9eb9292775d",
      "groupName": "/aws/codebuild/codebuild-demo-project",
      "deepLink": "https://console.aws.amazon.com/cloudwatch/
home?region=us-west-2#logEvent:group=/aws/codebuild/codebuild-demo-
project;stream=4dea3ca4-20ec-4898-b22a-a9eb9292775d"
    },
    "timeoutInMinutes": 60,
    "environment": {
      "privilegedMode": false,
      "computeType": "BUILD_GENERAL1_MEDIUM",
      "image": "aws/codebuild/windows-base:1.0",
      "environmentVariables": [],
      "type": "WINDOWS_CONTAINER"
    },
    "projectName": "codebuild-demo-project",
    "buildComplete": true,
    "source": {
      "gitCloneDepth": 1,
      "location": "https://github.com/my-repo/codebuild-demo-project.git",
      "insecureSsl": false,
      "reportBuildStatus": false,
      "type": "GITHUB"
    },
    "buildStatus": "SUCCEEDED",
```

```
"secondaryArtifacts": [],
"phases": [
  {
    "durationInSeconds": 0,
    "startTime": 1548716241.89,
    "phaseType": "SUBMITTED",
    "endTime": 1548716242.241,
    "phaseStatus": "SUCCEEDED"
  },
  {
    "durationInSeconds": 0,
    "startTime": 1548716242.241,
    "phaseType": "QUEUED",
    "endTime": 1548716242.536,
    "phaseStatus": "SUCCEEDED"
  },
  {
    "durationInSeconds": 33,
    "endTime": 1548716276.171,
    "contexts": [
      {
        "statusCode": "",
        "message": ""
      }
    ],
    "startTime": 1548716242.536,
    "phaseType": "PROVISIONING",
    "phaseStatus": "SUCCEEDED"
  },
  {
    "durationInSeconds": 15,
    "endTime": 1548716291.809,
    "contexts": [
      {
        "statusCode": "",
        "message": ""
      }
    ],
    "startTime": 1548716276.171,
    "phaseType": "DOWNLOAD_SOURCE",
    "phaseStatus": "SUCCEEDED"
  },
  {
    "durationInSeconds": 0,
```

```
    "endTime": 1548716291.993,
    "contexts": [
      {
        "statusCode": "",
        "message": ""
      }
    ],
    "startTime": 1548716291.809,
    "phaseType": "INSTALL",
    "phaseStatus": "SUCCEEDED"
  },
  {
    "durationInSeconds": 0,
    "endTime": 1548716292.191,
    "contexts": [
      {
        "statusCode": "",
        "message": ""
      }
    ],
    "startTime": 1548716291.993,
    "phaseType": "PRE_BUILD",
    "phaseStatus": "SUCCEEDED"
  },
  {
    "durationInSeconds": 9,
    "endTime": 1548716301.622,
    "contexts": [
      {
        "statusCode": "",
        "message": ""
      }
    ],
    "startTime": 1548716292.191,
    "phaseType": "BUILD",
    "phaseStatus": "SUCCEEDED"
  },
  {
    "durationInSeconds": 3,
    "endTime": 1548716304.783,
    "contexts": [
      {
        "statusCode": "",
        "message": ""
      }
    ]
  }
]
```

```
    }
  ],
  "startTime": 1548716301.622,
  "phaseType": "POST_BUILD",
  "phaseStatus": "SUCCEEDED"
},
{
  "durationInSeconds": 8,
  "endTime": 1548716313.775,
  "contexts": [
    {
      "statusCode": "",
      "message": ""
    }
  ],
  "startTime": 1548716304.783,
  "phaseType": "UPLOAD_ARTIFACTS",
  "phaseStatus": "SUCCEEDED"
},
{
  "durationInSeconds": 2,
  "endTime": 1548716315.935,
  "contexts": [
    {
      "statusCode": "",
      "message": ""
    }
  ],
  "startTime": 1548716313.775,
  "phaseType": "FINALIZING",
  "phaseStatus": "SUCCEEDED"
},
{
  "startTime": 1548716315.935,
  "phaseType": "COMPLETED"
}
],
"startTime": 1548716241.89,
"secondarySourceVersions": [],
"initiator": "my-codebuild-project",
"arn": "arn:aws:codebuild:us-west-2:123456789012:build/codebuild-demo-
project:815e755f-bade-4a7e-80f0-efe51EXAMPLE",
"encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
```

```

        "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-
codebuild-service-role",
        "currentPhase": "COMPLETED",
        "id": "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE",
        "cache": {
            "type": "NO_CACHE"
        },
        "endTime": 1548716315.935,
        "secondarySources": [],
        "queuedTimeoutInMinutes": 480,
        "resolvedSourceVersion": "f2194c1757bbdcb0f8f229254a4b3c8b27d43e0b"
    }
]
}

```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[查看版本详情 \(AWS CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[BatchGetBuilds](#)中的。

## batch-get-projects

以下代码示例演示了如何使用 batch-get-projects。

### AWS CLI

获取 AWS CodeBuild 构建项目名称列表。

以下 batch-get-projects 示例获取按名称指定的 CodeBuild 构建项目列表。

```
aws codebuild batch-get-projects --names codebuild-demo-project codebuild-demo-
project2 my-other-demo-project
```

在以下输出中，projectsNotFound 数组列出了所有已指定但未找到的构建项目名称。projects 数组列出了可找到相关信息的所有构建项目的详细信息。

```

{
  "projectsNotFound": [],
  "projects": [
    {
      "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
      "name": "codebuild-demo-project2",
      "queuedTimeoutInMinutes": 480,
      "timeoutInMinutes": 60,
    }
  ]
}

```

```

    "source": {
        "buildspec": "version: 0.2\n\n#env:\n #variables:\n # key:
\"value\"\n # key: \"value\"\n #parameter-store:\n # key: \"value\"\n
# key: \"value\"\n\n#phases:\n #install:\n #commands:\n # - command\n
# - command\n #pre_build:\n #commands:\n # - command\n # - command
\n build:\n #commands:\n # - command\n # - command\n #post_build:\n
#commands:\n # - command\n # - command\n#artifacts:\n #files:\n #
- location\n # - location\n #name: $(date +%Y-%m-%d)\n #discard-paths: yes\n
#base-directory: location\n#cache:\n #paths:\n # - paths",
        "type": "NO_SOURCE",
        "insecureSsl": false,
        "gitCloneDepth": 1
    },
    "artifacts": {
        "type": "NO_ARTIFACTS"
    },
    "badge": {
        "badgeEnabled": false
    },
    "lastModified": 1540588091.108,
    "created": 1540588091.108,
    "arn": "arn:aws:codebuild:us-west-2:123456789012:project/test-for-
sample",
    "secondarySources": [],
    "secondaryArtifacts": [],
    "cache": {
        "type": "NO_CACHE"
    },
    "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-test-
role",
    "environment": {
        "image": "aws/codebuild/java:openjdk-8",
        "privilegedMode": true,
        "type": "LINUX_CONTAINER",
        "computeType": "BUILD_GENERAL1_SMALL",
        "environmentVariables": []
    },
    "tags": []
},
{
    "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
    "name": "my-other-demo-project",
    "queuedTimeoutInMinutes": 480,
    "timeoutInMinutes": 60,

```

```
    "source": {
      "location": "https://github.com/iversonic/codedeploy-sample.git",
      "reportBuildStatus": false,
      "buildspec": "buildspec.yml",
      "insecureSsl": false,
      "gitCloneDepth": 1,
      "type": "GITHUB",
      "auth": {
        "type": "OAUTH"
      }
    },
    "artifacts": {
      "type": "NO_ARTIFACTS"
    },
    "badge": {
      "badgeEnabled": false
    },
    "lastModified": 1523401711.73,
    "created": 1523401711.73,
    "arn": "arn:aws:codebuild:us-west-2:123456789012:project/Project2",
    "cache": {
      "type": "NO_CACHE"
    },
    "serviceRole": "arn:aws:iam::123456789012:role/service-role/codebuild-Project2-service-role",
    "environment": {
      "image": "aws/codebuild/nodejs:4.4.7",
      "privilegedMode": false,
      "type": "LINUX_CONTAINER",
      "computeType": "BUILD_GENERAL1_SMALL",
      "environmentVariables": []
    },
    "tags": []
  }
]
```

有关更多信息，请参阅 [《AWS CodeBuild 用户指南》](#) 中的 [查看构建项目的详细信息 \(AWS CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchGetProjects](#) 中的。

## batch-get-report-groups

以下代码示例演示了如何使用 batch-get-report-groups。

### AWS CLI

要在中获取有关一个或多个报告组的信息 AWS CodeBuild。

以下batch-get-report-groups示例检索有关具有指定 ARN 的报告组的信息。

```
aws codebuild batch-get-report-groups \
  --report-group-arns arn:aws:codebuild:<region-ID>:<user-ID>:report-group/
  <report-group-name>
```

输出：

```
{
  "reportGroups": [
    {
      "arn": "arn:aws:codebuild:<region-ID>:<user-ID>:report-group/<report-
group-name>",
      "name": "report-group-name",
      "type": "TEST",
      "exportConfig": {
        "exportConfigType": "NO_EXPORT"
      },
      "created": "2020-10-01T18:04:08.466000+00:00",
      "lastModified": "2020-10-01T18:04:08.466000+00:00",
      "tags": []
    }
  ],
  "reportGroupsNotFound": []
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的使用[报告组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetReportGroups](#)中的。

## batch-get-reports

以下代码示例演示了如何使用 batch-get-reports。



## AWS CLI

要在中获取有关一个或多个报告的信息 AWS CodeBuild。

以下batch-get-reports示例检索有关具有指定 ARN 的报告的信息。

```
aws codebuild batch-get-reports \  
  --report-arns arn:aws:codebuild:<region-ID>:<user-ID>:report/<report-group-  
name>:<report 1 ID> arn:aws:codebuild:<region-ID>:<user-ID>:report/<report-group-  
name>:<report 2 ID>
```

输出：

```
{  
  "reports": [  
    {  
      "arn": "arn:aws:codebuild:<region-ID>:<user-ID>:report/<report-group-  
name>:<report 1 ID>",  
      "type": "TEST",  
      "name": "<report-group-name>",  
      "reportGroupArn": "arn:aws:codebuild:<region-ID>:<user-ID>:report-group/  
<report-group-name>",  
      "executionId": "arn:aws:codebuild:<region-ID>:<user-ID>:build/test-  
reports:<ID>",  
      "status": "FAILED",  
      "created": "2020-10-01T11:25:22.531000-07:00",  
      "expired": "2020-10-31T11:25:22-07:00",  
      "exportConfig": {  
        "exportConfigType": "NO_EXPORT"  
      },  
      "truncated": false,  
      "testSummary": {  
        "total": 28,  
        "statusCounts": {  
          "ERROR": 5,  
          "FAILED": 1,  
          "SKIPPED": 4,  
          "SUCCEEDED": 18,  
          "UNKNOWN": 0  
        },  
        "durationInNanoSeconds": 94000000  
      },  
    },  
  ],  
}
```

```
{
  "arn": "arn:aws:codebuild:<region-ID>:<user-ID>:report/<report-group-
name>:<report 2 ID>",
  "type": "TEST",
  "name": "<report-group-name>",
  "reportGroupArn": "arn:aws:codebuild:<region-ID>:<user-ID>:report-group/
<report-group-name>",
  "executionId": "arn:aws:codebuild:<region-ID>:<user-ID>:build/test-
reports:<ID>",
  "status": "FAILED",
  "created": "2020-10-01T11:13:05.816000-07:00",
  "expired": "2020-10-31T11:13:05-07:00",
  "exportConfig": {
    "exportConfigType": "NO_EXPORT"
  },
  "truncated": false,
  "testSummary": {
    "total": 28,
    "statusCounts": {
      "ERROR": 5,
      "FAILED": 1,
      "SKIPPED": 4,
      "SUCCEEDED": 18,
      "UNKNOWN": 0
    }
  },
  "durationInNanoSeconds": 94000000
}
],
"reportsNotFound": []
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[使用报告](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[BatchGetReports](#)中的。

## create-project

以下代码示例演示了如何使用 create-project。

### AWS CLI

#### 示例 1：创建 AWS CodeBuild 构建项目

以下create-project示例使用 S3 存储桶中的源文件创建 CodeBuild 构建项目

```
aws codebuild create-project \  
  --name "my-demo-project" \  
  --source "{\"type\": \"S3\", \"location\": \"codebuild-us-west-2-123456789012-  
input-bucket/my-source.zip\"}" \  
  --artifacts [{"type": \"S3\", \"location\": \"codebuild-us-west-2-123456789012-  
output-bucket\"}] \  
  --environment "{\"type\": \"LINUX_CONTAINER\", \"image\": \"aws/codebuild/  
standard:1.0\", \"computeType\": \"BUILD_GENERAL1_SMALL\"}" \  
  --service-role "arn:aws:iam::123456789012:role/service-role/my-codebuild-  
service-role"
```

输出：

```
{  
  "project": {  
    "arn": "arn:aws:codebuild:us-west-2:123456789012:project/my-demo-project",  
    "name": "my-cli-demo-project",  
    "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",  
    "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-  
service-role",  
    "lastModified": 1556839783.274,  
    "badge": {  
      "badgeEnabled": false  
    },  
    "queuedTimeoutInMinutes": 480,  
    "environment": {  
      "image": "aws/codebuild/standard:1.0",  
      "computeType": "BUILD_GENERAL1_SMALL",  
      "type": "LINUX_CONTAINER",  
      "imagePullCredentialsType": "CODEBUILD",  
      "privilegedMode": false,  
      "environmentVariables": []  
    },  
    "artifacts": {  
      "location": "codebuild-us-west-2-123456789012-output-bucket",  
      "name": "my-cli-demo-project",  
      "namespaceType": "NONE",  
      "type": "S3",  
      "packaging": "NONE",  
      "encryptionDisabled": false  
    },  
  },  
}
```

```
    "source": {
      "type": "S3",
      "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source.zip",
      "insecureSsl": false
    },
    "timeoutInMinutes": 60,
    "cache": {
      "type": "NO_CACHE"
    },
    "created": 1556839783.274
  }
}
```

## 示例 2：使用 JSON 输入文件作为参数创建 AWS CodeBuild 构建项目

以下 `create-project` 示例通过在 JSON 输入文件中传递所有必需的参数来创建 CodeBuild 构建项目。通过仅使用 `--generate-cli-skeleton parameter` 运行命令来创建输入文件模板。

```
aws codebuild create-project --cli-input-json file://create-project.json
```

输入 JSON 文件 `create-project.json` 包含以下内容：

```
{
  "name": "codebuild-demo-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:1.0",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "serviceIAMRole"
}
```

输出：

```
{
  "project": {
    "name": "codebuild-demo-project",
    "serviceRole": "serviceIAMRole",
    "tags": [],
    "artifacts": {
      "packaging": "NONE",
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-output-bucket",
      "name": "message-util.zip"
    },
    "lastModified": 1472661575.244,
    "timeoutInMinutes": 60,
    "created": 1472661575.244,
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "aws/codebuild/standard:1.0",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-input-bucket/
MessageUtil.zip"
    },
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:alias/aws/s3",
    "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-
project"
  }
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[创建构建项目 \(AWS CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateProject](#)中的。

## create-report-group

以下代码示例演示了如何使用 create-report-group。

### AWS CLI

在中创建报告组 AWS CodeBuild。

以下`create-report-group`示例创建了一个新的报告组。

```
aws codebuild create-report-group \  
  --cli-input-json file://create-report-group-source.json
```

`create-report-group-source.json` 的内容：

```
{  
  "name": "cli-created-report-group",  
  "type": "TEST",  
  "exportConfig": {  
    "exportConfigType": "S3",  
    "s3Destination": {  
      "bucket": "my-s3-bucket",  
      "path": "",  
      "packaging": "ZIP",  
      "encryptionDisabled": true  
    }  
  }  
}
```

输出：

```
{  
  "reportGroup": {  
    "arn": "arn:aws:codebuild:<region-ID>:<user-ID>:report-group/cli-created-report-group",  
    "name": "cli-created-report-group",  
    "type": "TEST",  
    "exportConfig": {  
      "exportConfigType": "S3",  
      "s3Destination": {  
        "bucket": "my-s3-bucket",  
        "path": "",  
        "packaging": "ZIP",  
        "encryptionDisabled": true  
      }  
    },  
    "created": 1602020026.775,  
    "lastModified": 1602020026.775  
  }  
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的使用[报告组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateReportGroup](#)中的。

## create-webhook

以下代码示例演示了如何使用 create-webhook。

### AWS CLI

为项目创建 webhook 过滤器 AWS CodeBuild

以下create-webhook示例为具有两个筛选器组的名my-project为的 CodeBuild 项目创建一个 webhook。第一个筛选条件组使用与正则表达式 `^refs/heads/master$` 匹配的 Git 引用名称以及 `^refs/heads/myBranch$` 匹配的头部引用，指定在分支上创建、更新或重新打开的拉取请求。第二个筛选器组在 Git 引用名称与正则表达式不匹配的分支上指定推送请求`^refs/heads/myBranch$`。

```
aws codebuild create-webhook \
  --project-name my-project \
  --filter-groups "[[{"type": "EVENT", "pattern": "PULL_REQUEST_CREATED,
  PULL_REQUEST_UPDATED, PULL_REQUEST_REOPENED"}, {"type": "HEAD_REF", "pattern
  \": "^refs/heads/myBranch$", "excludeMatchedPattern": true}, {"type": "BASE_REF
  \": "^refs/heads/master$", "excludeMatchedPattern": true}], [{"type": "
  EVENT", "pattern": "PUSH"}, {"type": "HEAD_REF", "pattern": "^refs/heads/
  myBranch$", "excludeMatchedPattern": true}]]"
```

输出：

```
{
  "webhook": {
    "payloadUrl": "https://codebuild.us-west-2.amazonaws.com/webhooks?
    t=eyJlbnNyeXB0ZWREYXRhIjoiVVV1SMGtoeGRwSzZFRXl12Wnh4bld1Z0tKZ291TVpQNEtFamQ3RD1DYWpRaGIreVFrdm
    "url": "https://api.github.com/repos/iversonic/codedeploy-sample/
    hooks/105190656",
    "lastModifiedSecret": 1556311319.069,
    "filterGroups": [
      [
        {
          "type": "EVENT",
          "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
          PULL_REQUEST_REOPENED",
```

```

        "excludeMatchedPattern": false
    },
    {
        "type": "HEAD_REF",
        "pattern": "refs/heads/myBranch$",
        "excludeMatchedPattern": true
    },
    {
        "type": "BASE_REF",
        "pattern": "refs/heads/master$",
        "excludeMatchedPattern": true
    }
],
[
    {
        "type": "EVENT",
        "pattern": "PUSH",
        "excludeMatchedPattern": false
    },
    {
        "type": "HEAD_REF",
        "pattern": "refs/heads/myBranch$",
        "excludeMatchedPattern": true
    }
]
]
}

```

有关更多信息，请参阅AWS CodeBuild 用户指南中的[过滤 GitHub Webhook 事件 \(SDK\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateWebhook](#)中的。

## delete-build-batch

以下代码示例演示了如何使用 delete-build-batch。

### AWS CLI

删除批量构建 AWS CodeBuild。

以下delete-build-batch示例删除了指定的批量构建。

```
aws codebuild delete-build-batch \
```



```
--id <project-name>:<batch-ID>
```

输出：

```
{
  "statusCode": "BATCH_DELETED",
  "buildsDeleted": [
    "arn:aws:codebuild:<region-ID>:<account-ID>:build/<project-name>:<build-ID>",
    "arn:aws:codebuild:<region-ID>:<account-ID>:build/<project-name>:<build-ID>",
    "arn:aws:codebuild:<region-ID>:<account-ID>:build/<project-name>:<build-ID>",
    "arn:aws:codebuild:<region-ID>:<account-ID>:build/<project-name>:<build-ID>"
  ],
  "buildsNotDeleted": []
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》AWS CodeBuild [中的 Batch 构建](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBuildBatch](#)中的。

## delete-project

以下代码示例演示了如何使用 delete-project。

AWS CLI

删除 AWS CodeBuild 构建项目

以下delete-project示例删除了指定的 CodeBuild 构建项目。

```
aws codebuild delete-project --name my-project
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[删除构建项目 \(AWS CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteProject](#)中的。

## delete-report-group

以下代码示例演示了如何使用 delete-report-group。

## AWS CLI

要删除中的报告组 AWS CodeBuild。

以下delete-report-group示例删除具有指定 ARN 的报告组。

```
aws codebuild delete-report-group \  
  --arn arn:aws:codebuild:<region-ID>:<user-ID>:report-group/<report-group-name>
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的使用[报告组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteReportGroup](#)中的。

## delete-report

以下代码示例演示了如何使用 delete-report。

## AWS CLI

要在中删除报告 AWS CodeBuild。

以下delete-report示例删除了指定的报告。

```
aws codebuild delete-report \  
  --arn arn:aws:codebuild:<region-ID>:<account-ID>:report/<report-group-  
name>:<report-ID>
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[使用报告](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteReport](#)中的。

## delete-source-credentials

以下代码示例演示了如何使用 delete-source-credentials。

## AWS CLI

断开与源提供商的连接并移除其访问令牌。

以下 `delete-source-credentials` 示例断开与源提供商的连接并删除其令牌。用于连接源提供商的源凭证的 ARN 决定了哪些源凭证。

```
aws codebuild delete-source-credentials --arn arn-of-your-credentials
```

输出：

```
{
  "arn": "arn:aws:codebuild:your-region:your-account-id:token/your-server-type"
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[使用访问令牌连接源提供商 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteSourceCredentials](#) 中的。

## delete-webhook

以下代码示例演示了如何使用 `delete-webhook`。

### AWS CLI

从项目中删除 webhook 过滤器 AWS CodeBuild

以下 `delete-webhook` 示例从指定 CodeBuild 项目中删除一个 webhook。

```
aws codebuild delete-webhook --project-name my-project
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[自动停止运行构建 \(AWS CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteWebhook](#) 中的。

## describe-code-coverages

以下代码示例演示了如何使用 `describe-code-coverages`。

### AWS CLI

要获取有关代码覆盖率测试结果的详细信息，请参阅 AWS CodeBuild。

以下describe-code-coverages示例在指定报告中获取有关代码覆盖率测试结果的信息。

```
aws codebuild describe-code-coverages \  
  --report-arn arn:aws:codebuild:<region-ID>:<account-ID>:report/<report-group-  
name>:<report-ID>
```

输出：

```
{  
  "codeCoverages": [  
    {  
      "id": "20a0adcc-db13-4b66-804b-ecaf9f852855",  
      "reportARN": "arn:aws:codebuild:<region-ID>:972506530580:report/<report-  
group-name>:<report-ID>",  
      "filePath": "<source-file-1-path>",  
      "lineCoveragePercentage": 83.33,  
      "linesCovered": 5,  
      "linesMissed": 1,  
      "branchCoveragePercentage": 50.0,  
      "branchesCovered": 1,  
      "branchesMissed": 1,  
      "expired": "2020-11-20T21:22:45+00:00"  
    },  
    {  
      "id": "0887162d-bf57-4cf1-a164-e432373d1a83",  
      "reportARN": "arn:aws:codebuild:<region-ID>:972506530580:report/<report-  
group-name>:<report-ID>",  
      "filePath": "<source-file-2-path>",  
      "lineCoveragePercentage": 90.9,  
      "linesCovered": 10,  
      "linesMissed": 1,  
      "branchCoveragePercentage": 50.0,  
      "branchesCovered": 1,  
      "branchesMissed": 1,  
      "expired": "2020-11-20T21:22:45+00:00"  
    }  
  ]  
}
```

有关更多信息，请参阅 [《AWS CodeBuild 用户指南》中的代码覆盖率报告](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCodeCoverages](#)中的。

## describe-test-cases

以下代码示例演示了如何使用 describe-test-cases。

### AWS CLI

要获取有关测试用例的详细信息，请访问 AWS CodeBuild。

以下 describe-test-cases 示例获取有关指定报告中测试用例的信息。

```
aws codebuild describe-test-cases \  
  --report-arn arn:aws:codebuild:<region-ID>:<account-ID>:report/<report-group-  
name>:<report-ID>
```

输出：

```
{  
  "testCases": [  
    {  
      "reportArn": "arn:aws:codebuild:<region-ID>:<account-ID>:report/<report-  
group-name>:<report-ID>",  
      "testRawDataPath": "<test-report-path>",  
      "prefix": "NUnit.Tests.Assemblies.MockTestFixture",  
      "name": "NUnit.Tests.Assemblies.MockTestFixture.NotRunnableTest",  
      "status": "ERROR",  
      "durationInNanoSeconds": 0,  
      "message": "No arguments were provided\n",  
      "expired": "2020-11-20T17:52:10+00:00"  
    },  
    {  
      "reportArn": "arn:aws:codebuild:<region-ID>:<account-ID>:report/<report-  
group-name>:<report-ID>",  
      "testRawDataPath": "<test-report-path>",  
      "prefix": "NUnit.Tests.Assemblies.MockTestFixture",  
      "name": "NUnit.Tests.Assemblies.MockTestFixture.TestWithException",  
      "status": "ERROR",  
      "durationInNanoSeconds": 0,  
      "message": "System.ApplicationException : Intentional Exception  
\nat NUnit.Tests.Assemblies.MockTestFixture.MethodThrowsException()\nat  
NUnit.Tests.Assemblies.MockTestFixture.TestWithException()\n\n",  
      "expired": "2020-11-20T17:52:10+00:00"  
    }  
  ]  
}
```

```
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》AWS CodeBuild [中的使用测试报告](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTestCases](#)中的。

## import-source-credentials

以下代码示例演示了如何使用 import-source-credentials。

### AWS CLI

通过导入源提供者的凭据，将 AWS CodeBuild 用户连接到源提供者。

以下import-source-credentials示例为使用 BASIC\_AUTH 作为其身份验证类型的 Bitbucket 存储库导入令牌。

```
aws codebuild import-source-credentials --server-type BITBUCKET --auth-type
BASIC_AUTH --token my-Bitbucket-password --username my-Bitbucket-username
```

输出：

```
{
  "arn": "arn:aws:codebuild:us-west-2:123456789012:token/bitbucket"
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[使用访问令牌连接源提供商 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ImportSourceCredentials](#)中的。

## invalidate-project-cache

以下代码示例演示了如何使用 invalidate-project-cache。

### AWS CLI

重置 AWS CodeBuild 构建项目的缓存。

以下invalidate-project-cache示例重置指定 CodeBuild 项目的缓存。

```
aws codebuild invalidate-project-cache --project-name my-project
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodeBuild 用户指南》CodeBuild [中的构建缓存](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[InvalidateProjectCache](#)中的。

## list-build-batches-for-project

以下代码示例演示了如何使用 list-build-batches-for-project。

### AWS CLI

要在中列出特定构建项目的批量构建 AWS CodeBuild。

以下list-build-batches-for-project示例列出了指定项目的 CodeBuild 批量构建。

```
aws codebuild list-build-batches-for-project \  
  --project-name "<project-name>"
```

输出：

```
{  
  "ids": [  
    "<project-name>:<batch-ID>",  
    "<project-name>:<batch-ID>"  
  ]  
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》AWS CodeBuild [中的 Batch 构建](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBuildBatchesForProject](#)中的。

## list-build-batches

以下代码示例演示了如何使用 list-build-batches。

### AWS CLI

在中列出批量构建 AWS CodeBuild。

以下list-build-batches示例列出了当前账户的 CodeBuild 批量构建。

```
aws codebuild list-build-batches
```

输出：

```
{
  "ids": [
    "<project-name>:<batch-ID>",
    "<project-name>:<batch-ID>"
  ]
}
```

有关更多信息，请参阅《用户AWS CodeBuild 指南》AWS CodeBuild 中的 Batch building in。 <<https://docs.aws.amazon.com/codebuild/latest/userguide/batch-build.html>>

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBuildBatches](#)中的。

## list-builds-for-project

以下代码示例演示了如何使用 list-builds-for-project。

AWS CLI

查看 AWS CodeBuild 构建项目的版本列表。

以下list-builds-for-project示例按降序列出了指定 CodeBuild 构建项目的构建 ID。

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project --sort-order DESCENDING
```

输出：

```
{
  "ids": [
    "codebuild-demo-project:1a2b3c4d-5678-90ab-cdef-11111example",
    "codebuild-demo-project:1a2b3c4d-5678-90ab-cdef-22222example",
    "codebuild-demo-project:1a2b3c4d-5678-90ab-cdef-33333example",
    "codebuild-demo-project:1a2b3c4d-5678-90ab-cdef-44444example",
    "codebuild-demo-project:1a2b3c4d-5678-90ab-cdef-55555example"
  ]
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的“[查看构建项目的构建 ID 列表 \(AWS CLI\)](#)”

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBuildsForProject](#)中的。



## list-builds

以下代码示例演示了如何使用 list-builds。

### AWS CLI

获取构 AWS CodeBuild 建 ID 列表。

以下list-builds示例获取按升序排序的 CodeBuild ID 列表。

```
aws codebuild list-builds --sort-order ASCENDING
```

输出包含一个nextToken值，该值表示还有更多可用的输出。

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for
brevity...MzY20A==",
  "ids": [
    "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE"
    "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
  ]
}
```

再次运行此命令并提供上一个响应中的nextToken值作为参数，以获取输出的下一部分。重复此操作，直到您在响应中没有收到任何nextToken值。

```
aws codebuild list-builds --sort-order ASCENDING --next-token 4AEA6u7J...The full
token has been omitted for brevity...MzY20A==
```

输出的下一部分：

```
{
  "ids": [
    "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
    "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
  ]
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[查看构建 ID 列表 \(AWS CLI\)](#)

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBuilds](#)中的。

## list-curated-environment-images

以下代码示例演示了如何使用 `list-curated-environment-images`。

### AWS CLI

要获取由其管理的 Docker 映像列表 AWS CodeBuild ，您可以将其用于构建。

以下`list-curated-environment-images`示例列出了由管理的、可用于构建 CodeBuild 的 Docker 镜像。：

```
aws codebuild list-curated-environment-images
```

输出：

```
{
  "platforms": [
    {
      "platform": "AMAZON_LINUX",
      "languages": [
        {
          "language": "JAVA",
          "images": [
            {
              "description": "AWS ElasticBeanstalk - Java 7 Running on
Amazon Linux 64bit v2.1.3",
              "name": "aws/codebuild/eb-java-7-amazonlinux-64:2.1.3",
              "versions": [
                "aws/codebuild/eb-java-7-amazonlinux-64:2.1.3-1.0.0"
              ]
            },
            {
              "description": "AWS ElasticBeanstalk - Java 8 Running on
Amazon Linux 64bit v2.1.3",
              "name": "aws/codebuild/eb-java-8-amazonlinux-64:2.1.3",
              "versions": [
                "aws/codebuild/eb-java-8-amazonlinux-64:2.1.3-1.0.0"
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

```

    },
    ... LIST TRUNCATED FOR BREVITY ...
  ]
}
]
}
]
}
}

```

有关更多信息，请参阅《AWS CodeBuild 用户指南》CodeBuild中[提供的 Docker 镜像](#)

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCuratedEnvironmentImages](#)中的。

## list-projects

以下代码示例演示了如何使用 list-projects。

### AWS CLI

获取 AWS CodeBuild 构建项目名称列表。

以下list-projects示例获取按名称升序排序的 CodeBuild 构建项目列表。

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING
```

输出包含一个nextToken值，该值表示还有更多可用的输出。

```

{
  "nextToken": "Ci33ACF6...The full token has been omitted for brevity...U
+AkMx8=",
  "projects": [
    "codebuild-demo-project",
    "codebuild-demo-project2",
    ... The full list of build project names has been omitted for
brevity ...
    "codebuild-demo-project99"
  ]
}

```

再次运行此命令并提供上一个响应中的nextToken值作为参数，以获取输出的下一部分。重复此操作，直到您在响应中没有收到任何nextToken值。

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING --next-token
Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=

{
  "projects": [
    "codebuild-demo-project100",
    "codebuild-demo-project101",
    ... The full list of build project names has been omitted for
    brevity ...
    "codebuild-demo-project122"
  ]
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[查看构建项目名称列表 \(AWS CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListProjects](#)中的。

## list-report-groups

以下代码示例演示了如何使用 `list-report-groups`。

### AWS CLI

获取报告组 ARN 的 AWS CodeBuild 列表。

以下 `list-report-groups` 示例检索该地区账户的报告组 ARN。

```
aws codebuild list-report-groups
```

输出：

```
{
  "reportGroups": [
    "arn:aws:codebuild:<region-ID>:<user-ID>:report-group/report-group-1",
    "arn:aws:codebuild:<region-ID>:<user-ID>:report-group/report-group-2",
    "arn:aws:codebuild:<region-ID>:<user-ID>:report-group/report-group-3"
  ]
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的使用[报告组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListReportGroups](#)中的。

## list-reports-for-report-group

以下代码示例演示了如何使用 `list-reports-for-report-group`。

### AWS CLI

要在中获取报告组中的报告列表 AWS CodeBuild。

以下`list-report-for-report-groups`示例检索该地区账户在指定报告组中的报告。

```
aws codebuild list-reports-for-report-group \  
  --report-group-arn arn:aws:codebuild:<region-ID>:<user-ID>:report-group/<report-  
group-name>
```

输出：

```
{  
  "reports": [  
    "arn:aws:codebuild:<region-ID>:<user-ID>:report/report-1",  
    "arn:aws:codebuild:<region-ID>:<user-ID>:report/report-2",  
    "arn:aws:codebuild:<region-ID>:<user-ID>:report/report-3"  
  ]  
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的使用[报告组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListReportsForReportGroup](#)中的。

## list-reports

以下代码示例演示了如何使用 `list-reports`。

### AWS CLI

要获取当前账户的报告清单，请访问 AWS CodeBuild。

以下`list-reports`示例检索当前账户的报告的 ARN。

```
aws codebuild list-reports
```

输出：

```
{
```

```
"reports": [  
  "arn:aws:codebuild:<region-ID>:<user-ID>:report/<report-group-name>:<report  
ID>",  
  "arn:aws:codebuild:<region-ID>:<user-ID>:report/<report-group-name>:<report  
ID>",  
  "arn:aws:codebuild:<region-ID>:<user-ID>:report/<report-group-name>:<report  
ID>"  
]  
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[使用报告](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListReports](#)中的。

## list-shared-projects

以下代码示例演示了如何使用 list-shared-projects。

### AWS CLI

在中列出共享项目 AWS CodeBuild。

以下list-shared-projects示例列出了当前账户可用的 CodeBuild 共享项目。

```
aws codebuild list-shared-projects
```

输出：

```
{  
  "projects": [  
    "arn:aws:codebuild:<region-ID>:<account-ID>:project/<shared-project-  
name-1>",  
    "arn:aws:codebuild:<region-ID>:<account-ID>:project/<shared-project-name-2>"  
  ]  
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[使用共享项目](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSharedProjects](#)中的。

## list-shared-report-groups

以下代码示例演示了如何使用 list-shared-report-groups。

## AWS CLI

获取共享报告组 ARN 的 AWS CodeBuild 列表。

以下 `list-shared-report-groups` 示例检索该地区账户的报告组 ARN。

```
aws codebuild list-shared-report-groups
```

输出：

```
{
  "reportGroups": [
    "arn:aws:codebuild:<region-ID>:<user-ID>:report-group/report-group-1",
    "arn:aws:codebuild:<region-ID>:<user-ID>:report-group/report-group-2",
    "arn:aws:codebuild:<region-ID>:<user-ID>:report-group/report-group-3"
  ]
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的使用[报告组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListSharedReportGroups](#)中的。

## list-source-credentials

以下代码示例演示了如何使用 `list-source-credentials`。

### AWS CLI

要查看清单 `sourceCredentialsObjects`

以下 `list-source-credentials` 示例列出了与一个 Bitbucket AWS 账户和一个 GitHub 账户关联的账户的代币。响应中的每个 `sourceCredentialsInfos` 对象都包含连接的源凭证信息。

```
aws codebuild list-source-credentials
```

输出：

```
{
  "sourceCredentialsInfos": [
    {
      "serverType": "BITBUCKET",
      "arn": "arn:aws:codebuild:us-west-2:123456789012:token/bitbucket",

```

```

        "authType": "BASIC_AUTH"
    },
    {
        "serverType": "GITHUB",
        "arn": "arn:aws:codebuild:us-west-2:123456789012:token/github",
        "authType": "OAUTH"
    }
]
}

```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[使用访问令牌连接源提供商 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListSourceCredentials](#)中的。

## retry-build-batch

以下代码示例演示了如何使用 `retry-build-batch`。

### AWS CLI

重试失败的批量构建。AWS CodeBuild

以下 `retry-build-batch` 示例重新启动指定的批量构建。

```

aws codebuild retry-build-batch \
  --id <project-name>:<batch-ID>

```

输出：

```

{
  "buildBatch": {
    "id": "<project-name>:<batch-ID>",
    "arn": "arn:aws:codebuild:<region-ID>:<account-ID>:build-batch/<project-name>:<batch-ID>",
    "startTime": "2020-10-21T17:26:23.099000+00:00",
    "currentPhase": "SUBMITTED",
    "buildBatchStatus": "IN_PROGRESS",
    "resolvedSourceVersion": "3a9e11cb419e8fff14b03883dc4e64f6155aaa7e",
    "projectName": "<project-name>",
    "phases": [
      {
        "phaseType": "SUBMITTED",
        "phaseStatus": "SUCCEEDED",

```



```
    "startTime": "2020-10-21T17:26:23.099000+00:00",
    "endTime": "2020-10-21T17:26:23.457000+00:00",
    "durationInSeconds": 0
  },
  {
    "phaseType": "DOWNLOAD_BATCHSPEC",
    "phaseStatus": "SUCCEEDED",
    "startTime": "2020-10-21T17:26:23.457000+00:00",
    "endTime": "2020-10-21T17:26:54.902000+00:00",
    "durationInSeconds": 31
  },
  {
    "phaseType": "IN_PROGRESS",
    "phaseStatus": "CLIENT_ERROR",
    "startTime": "2020-10-21T17:26:54.902000+00:00",
    "endTime": "2020-10-21T17:28:16.060000+00:00",
    "durationInSeconds": 81
  },
  {
    "phaseType": "FAILED",
    "phaseStatus": "RETRY",
    "startTime": "2020-10-21T17:28:16.060000+00:00",
    "endTime": "2020-10-21T17:29:39.709000+00:00",
    "durationInSeconds": 83
  },
  {
    "phaseType": "SUBMITTED",
    "startTime": "2020-10-21T17:29:39.709000+00:00"
  }
],
"source": {
  "type": "GITHUB",
  "location": "https://github.com/strohm-a/<project-name>-graph.git",
  "gitCloneDepth": 1,
  "gitSubmodulesConfig": {
    "fetchSubmodules": false
  },
  "reportBuildStatus": false,
  "insecureSsl": false
},
"secondarySources": [],
"secondarySourceVersions": [],
"artifacts": {
  "location": ""
}
```

```
    },
    "secondaryArtifacts": [],
    "cache": {
      "type": "NO_CACHE"
    },
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/amazonlinux2-x86_64-standard:3.0",
    "computeType": "BUILD_GENERAL1_SMALL",
    "environmentVariables": [],
    "privilegedMode": false,
    "imagePullCredentialsType": "CODEBUILD"
  },
  "logConfig": {
    "cloudWatchLogs": {
      "status": "ENABLED"
    },
    "s3Logs": {
      "status": "DISABLED",
      "encryptionDisabled": false
    }
  },
  "buildTimeoutInMinutes": 60,
  "queuedTimeoutInMinutes": 480,
  "complete": false,
  "initiator": "<username>",
  "encryptionKey": "arn:aws:kms:<region-ID>:<account-ID>:alias/aws/s3",
  "buildBatchNumber": 4,
  "buildBatchConfig": {
    "serviceRole": "arn:aws:iam::<account-ID>:role/service-role/<project-
name>",
    "restrictions": {
      "maximumBuildsAllowed": 100
    },
    "timeoutInMins": 480
  },
  "buildGroups": [
    {
      "identifier": "DOWNLOAD_SOURCE",
      "ignoreFailure": false,
      "currentBuildSummary": {
        "arn": "arn:aws:codebuild:<region-ID>:<account-ID>:build/
<project-name>:<build-ID>",
        "requestedOn": "2020-10-21T17:26:23.889000+00:00",
```

```
        "buildStatus": "SUCCEEDED",
        "primaryArtifact": {
            "type": "no_artifacts",
            "identifier": "DOWNLOAD_SOURCE"
        },
        "secondaryArtifacts": []
    }
},
{
    "identifier": "linux_small",
    "dependsOn": [],
    "ignoreFailure": false,
    "currentBuildSummary": {
        "arn": "arn:aws:codebuild:<region-ID>:<account-ID>:build/
<project-name>:<build-ID>",
        "requestedOn": "2020-10-21T17:26:55.115000+00:00",
        "buildStatus": "FAILED",
        "primaryArtifact": {
            "type": "no_artifacts",
            "identifier": "linux_small"
        },
        "secondaryArtifacts": []
    }
},
{
    "identifier": "linux_medium",
    "dependsOn": [
        "linux_small"
    ],
    "ignoreFailure": false,
    "currentBuildSummary": {
        "arn": "arn:aws:codebuild:<region-ID>:<account-ID>:build/
<project-name>:<build-ID>",
        "requestedOn": "2020-10-21T17:26:54.594000+00:00",
        "buildStatus": "STOPPED"
    }
},
{
    "identifier": "linux_large",
    "dependsOn": [
        "linux_medium"
    ],
    "ignoreFailure": false,
    "currentBuildSummary": {
```

```

        "arn": "arn:aws:codebuild:<region-ID>:<account-ID>:build/
<project-name>:<build-ID>",
        "requestedOn": "2020-10-21T17:26:54.701000+00:00",
        "buildStatus": "STOPPED"
    }
}
]
}
}

```

有关更多信息，请参阅《AWS CodeBuild 用户指南》AWS CodeBuild 中的 [Batch 构建](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RetryBuildBatch](#) 中的。

## retry-build

以下代码示例演示了如何使用 `retry-build`。

### AWS CLI

重试失败的内置。AWS CodeBuild

以下 `retry-build` 示例重新启动指定的构建。

```

aws codebuild retry-build \
  --id <project-name>:<build-ID>

```

输出：

```

{
  "build": {
    "id": "<project-name>:<build-ID>",
    "arn": "arn:aws:codebuild:<region-ID>:<account-ID>:build/<project-
name>:<build-ID>",
    "buildNumber": 9,
    "startTime": "2020-10-21T17:51:38.161000+00:00",
    "currentPhase": "QUEUED",
    "buildStatus": "IN_PROGRESS",
    "projectName": "<project-name>",
    "phases": [
      {
        "phaseType": "SUBMITTED",
        "phaseStatus": "SUCCEEDED",

```

```

        "startTime": "2020-10-21T17:51:38.161000+00:00",
        "endTime": "2020-10-21T17:51:38.210000+00:00",
        "durationInSeconds": 0
    },
    {
        "phaseType": "QUEUED",
        "startTime": "2020-10-21T17:51:38.210000+00:00"
    }
],
"source": {
    "type": "GITHUB",
    "location": "<GitHub-repo-URL>",
    "gitCloneDepth": 1,
    "gitSubmodulesConfig": {
        "fetchSubmodules": false
    },
    "reportBuildStatus": false,
    "insecureSsl": false
},
"secondarySources": [],
"secondarySourceVersions": [],
"artifacts": {
    "location": ""
},
"secondaryArtifacts": [],
"cache": {
    "type": "NO_CACHE"
},
"environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/amazonlinux2-x86_64-standard:3.0",
    "computeType": "BUILD_GENERAL1_SMALL",
    "environmentVariables": [],
    "privilegedMode": false,
    "imagePullCredentialsType": "CODEBUILD"
},
"serviceRole": "arn:aws:iam::<account-ID>:role/service-role/<service-role-name>",
"logs": {
    "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=<region-ID>#logEvent:group=null;stream=null",
    "cloudWatchLogsArn": "arn:aws:logs:<region-ID>:<account-ID>:log-group:null:log-stream:null",
    "cloudWatchLogs": {

```

```

        "status": "ENABLED"
    },
    "s3Logs": {
        "status": "DISABLED",
        "encryptionDisabled": false
    }
},
"timeoutInMinutes": 60,
"queuedTimeoutInMinutes": 480,
"buildComplete": false,
"initiator": "<username>",
"encryptionKey": "arn:aws:kms:<region-ID>:<account-ID>:alias/aws/s3"
}
}

```

有关更多信息，请参阅《AWS CodeBuild 用户指南》AWS CodeBuild [中的 Batch 构建](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RetryBuild](#) 中的。

## start-build-batch

以下代码示例演示了如何使用 start-build-batch。

### AWS CLI

开始批量构建 AWS CodeBuild。

以下 start-build-batch 示例启动指定项目的批量构建。

```
aws codebuild start-build-batch \
  --project-name <project-name>
```

输出：

```
{
  "buildBatch": {
    "id": "<project-name>:<batch-ID>",
    "arn": "arn:aws:codebuild:<region-ID>:<account-ID>:build-batch/<project-name>:<batch-ID>",
    "startTime": "2020-10-21T16:54:24.740000+00:00",
    "currentPhase": "SUBMITTED",
    "buildBatchStatus": "IN_PROGRESS",
    "projectName": "<project-name>",

```

```
"source": {
  "type": "GITHUB",
  "location": "<GitHub-repo-URL>",
  "gitCloneDepth": 1,
  "gitSubmodulesConfig": {
    "fetchSubmodules": false
  },
  "reportBuildStatus": false,
  "insecureSsl": false
},
"secondarySources": [],
"secondarySourceVersions": [],
"artifacts": {
  "location": ""
},
"secondaryArtifacts": [],
"cache": {
  "type": "NO_CACHE"
},
"environment": {
  "type": "LINUX_CONTAINER",
  "image": "aws/codebuild/amazonlinux2-x86_64-standard:3.0",
  "computeType": "BUILD_GENERAL1_SMALL",
  "environmentVariables": [],
  "privilegedMode": false,
  "imagePullCredentialsType": "CODEBUILD"
},
"logConfig": {
  "cloudWatchLogs": {
    "status": "ENABLED"
  },
  "s3Logs": {
    "status": "DISABLED",
    "encryptionDisabled": false
  }
},
"buildTimeoutInMinutes": 60,
"queuedTimeoutInMinutes": 480,
"complete": false,
"initiator": "<username>",
"encryptionKey": "arn:aws:kms:<region-ID>:<account-ID>:alias/aws/s3",
"buildBatchNumber": 3,
"buildBatchConfig": {
```

```

        "serviceRole": "arn:aws:iam::<account-ID>:role/service-role/<service-
role-name>",
        "restrictions": {
            "maximumBuildsAllowed": 100
        },
        "timeoutInMins": 480
    }
}
}

```

有关更多信息，请参阅《AWS CodeBuild 用户指南》AWS CodeBuild 中的 [Batch 构建](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartBuildBatch](#) 中的。

## start-build

以下代码示例演示了如何使用 start-build。

### AWS CLI

开始运行 AWS CodeBuild 构建项目的构建。

以下 start-build 示例为指定 CodeBuild 项目启动构建。该版本会覆盖项目的设置（允许在生成超时之前排队的分钟数）和项目的构件设置。

```

aws codebuild start-build \
  --project-name "my-demo-project" \
  --queued-timeout-in-minutes-override 5 \
  --artifacts-override {"\type\": \"S3\", \"location\": \"arn:aws:s3:::artifacts-
override\", \"overrideArtifactName\": true"}

```

输出：

```

{
  "build": {
    "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role",
    "buildStatus": "IN_PROGRESS",
    "buildComplete": false,
    "projectName": "my-demo-project",
    "timeoutInMinutes": 60,
    "source": {
      "insecureSsl": false,

```



```
    "type": "S3",
    "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source.zip"
  },
  "queuedTimeoutInMinutes": 5,
  "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
  "currentPhase": "QUEUED",
  "startTime": 1556905683.568,
  "environment": {
    "computeType": "BUILD_GENERAL1_MEDIUM",
    "environmentVariables": [],
    "type": "LINUX_CONTAINER",
    "privilegedMode": false,
    "image": "aws/codebuild/standard:1.0",
    "imagePullCredentialsType": "CODEBUILD"
  },
  "phases": [
    {
      "phaseStatus": "SUCCEEDED",
      "startTime": 1556905683.568,
      "phaseType": "SUBMITTED",
      "durationInSeconds": 0,
      "endTime": 1556905684.524
    },
    {
      "startTime": 1556905684.524,
      "phaseType": "QUEUED"
    }
  ],
  "logs": {
    "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logEvent:group=null;stream=null"
  },
  "artifacts": {
    "encryptionDisabled": false,
    "location": "arn:aws:s3:::artifacts-override/my-demo-project",
    "overrideArtifactName": true
  },
  "cache": {
    "type": "NO_CACHE"
  },
  "id": "my-demo-project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE",
  "initiator": "my-aws-account-name",
```

```
    "arn": "arn:aws:codebuild:us-west-2:123456789012:build/my-demo-  
project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE"  
  }  
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[运行构建 \(AWS CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartBuild](#) 中的。

## stop-build-batch

以下代码示例演示了如何使用 stop-build-batch。

### AWS CLI

停止正在进行的批量构建。AWS CodeBuild

以下 stop-build-batch 示例停止指定的批量构建。

```
aws codebuild stop-build-batch \  
  --id <project-name>:<batch-ID>
```

输出：

```
{  
  "buildBatch": {  
    "id": "<project-name>:<batch-ID>",  
    "arn": "arn:aws:codebuild:<region-ID>:<account-ID>:build-batch/<project-  
name>:<batch-ID>",  
    "startTime": "2020-10-21T16:54:24.740000+00:00",  
    "endTime": "2020-10-21T16:56:05.152000+00:00",  
    "currentPhase": "STOPPED",  
    "buildBatchStatus": "STOPPED",  
    "resolvedSourceVersion": "aef7744ed069c51098e15c360f4102cd2cd1ad64",  
    "projectName": "<project-name>",  
    "phases": [  
      {  
        "phaseType": "SUBMITTED",  
        "phaseStatus": "SUCCEEDED",  
        "startTime": "2020-10-21T16:54:24.740000+00:00",  
        "endTime": "2020-10-21T16:54:25.039000+00:00",  
        "durationInSeconds": 0  
      }  
    ],  
  },  
}
```

```
{
  "phaseType": "DOWNLOAD_BATCHSPEC",
  "phaseStatus": "SUCCEEDED",
  "startTime": "2020-10-21T16:54:25.039000+00:00",
  "endTime": "2020-10-21T16:54:56.583000+00:00",
  "durationInSeconds": 31
},
{
  "phaseType": "IN_PROGRESS",
  "phaseStatus": "STOPPED",
  "startTime": "2020-10-21T16:54:56.583000+00:00",
  "endTime": "2020-10-21T16:56:05.152000+00:00",
  "durationInSeconds": 68
},
{
  "phaseType": "STOPPED",
  "startTime": "2020-10-21T16:56:05.152000+00:00"
}
],
"source": {
  "type": "GITHUB",
  "location": "<GitHub-repo-URL>",
  "gitCloneDepth": 1,
  "gitSubmodulesConfig": {
    "fetchSubmodules": false
  },
  "reportBuildStatus": false,
  "insecureSsl": false
},
"secondarySources": [],
"secondarySourceVersions": [],
"artifacts": {
  "location": ""
},
"secondaryArtifacts": [],
"cache": {
  "type": "NO_CACHE"
},
"environment": {
  "type": "LINUX_CONTAINER",
  "image": "aws/codebuild/amazonlinux2-x86_64-standard:3.0",
  "computeType": "BUILD_GENERAL1_SMALL",
  "environmentVariables": [],
  "privilegedMode": false,
```

```

    "imagePullCredentialsType": "CODEBUILD"
  },
  "logConfig": {
    "cloudWatchLogs": {
      "status": "ENABLED"
    },
    "s3Logs": {
      "status": "DISABLED",
      "encryptionDisabled": false
    }
  },
  "buildTimeoutInMinutes": 60,
  "queuedTimeoutInMinutes": 480,
  "complete": true,
  "initiator": "Strohm",
  "encryptionKey": "arn:aws:kms:<region-ID>:<account-ID>:alias/aws/s3",
  "buildBatchNumber": 3,
  "buildBatchConfig": {
    "serviceRole": "arn:aws:iam::<account-ID>:role/service-role/<project-
name>",
    "restrictions": {
      "maximumBuildsAllowed": 100
    },
    "timeoutInMins": 480
  },
  "buildGroups": [
    {
      "identifier": "DOWNLOAD_SOURCE",
      "ignoreFailure": false,
      "currentBuildSummary": {
        "arn": "arn:aws:codebuild:<region-ID>:<account-ID>:build/
<project-name>:<build-ID>",
        "requestedOn": "2020-10-21T16:54:25.468000+00:00",
        "buildStatus": "SUCCEEDED",
        "primaryArtifact": {
          "type": "no_artifacts",
          "identifier": "DOWNLOAD_SOURCE"
        },
        "secondaryArtifacts": []
      }
    },
    {
      "identifier": "linux_small",
      "dependsOn": [],

```

```

        "ignoreFailure": false,
        "currentBuildSummary": {
            "arn": "arn:aws:codebuild:<region-ID>:<account-ID>:build/
<project-name>:<build-ID>",
            "requestedOn": "2020-10-21T16:54:56.833000+00:00",
            "buildStatus": "IN_PROGRESS"
        }
    },
    {
        "identifier": "linux_medium",
        "dependsOn": [
            "linux_small"
        ],
        "ignoreFailure": false,
        "currentBuildSummary": {
            "arn": "arn:aws:codebuild:<region-ID>:<account-ID>:build/
<project-name>:<build-ID>",
            "requestedOn": "2020-10-21T16:54:56.211000+00:00",
            "buildStatus": "PENDING"
        }
    },
    {
        "identifier": "linux_large",
        "dependsOn": [
            "linux_medium"
        ],
        "ignoreFailure": false,
        "currentBuildSummary": {
            "arn": "arn:aws:codebuild:<region-ID>:<account-ID>:build/
<project-name>:<build-ID>",
            "requestedOn": "2020-10-21T16:54:56.330000+00:00",
            "buildStatus": "PENDING"
        }
    }
]
}
}

```

有关更多信息，请参阅《AWS CodeBuild 用户指南》AWS CodeBuild [中的 Batch 构建](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StopBuildBatch](#) 中的。

## stop-build

以下代码示例演示了如何使用 stop-build。

### AWS CLI

停止构建项目的构 AWS CodeBuild 建。

以下stop-build示例停止指定的 CodeBuild 构建。

```
aws codebuild stop-build --id my-demo-project:12345678-a1b2-c3d4-e5f6-11111EXAMPLE
```

输出：

```
{
  "build": {
    "startTime": 1556906956.318,
    "initiator": "my-aws-account-name",
    "projectName": "my-demo-project",
    "currentPhase": "COMPLETED",
    "cache": {
      "type": "NO_CACHE"
    },
    "source": {
      "insecureSsl": false,
      "location": "codebuild-us-west-2-123456789012-input-bucket/my-source.zip",
      "type": "S3"
    },
    "id": "my-demo-project:1a2b3c4d-5678-90ab-cdef-11111EXAMPLE",
    "endTime": 1556906974.781,
    "phases": [
      {
        "durationInSeconds": 0,
        "phaseType": "SUBMITTED",
        "endTime": 1556906956.935,
        "phaseStatus": "SUCCEEDED",
        "startTime": 1556906956.318
      },
      {
        "durationInSeconds": 1,
        "phaseType": "QUEUED",
        "endTime": 1556906958.272,
        "phaseStatus": "SUCCEEDED",

```

```
    "startTime": 1556906956.935
  },
  {
    "phaseType": "PROVISIONING",
    "phaseStatus": "SUCCEEDED",
    "durationInSeconds": 14,
    "contexts": [
      {
        "message": "",
        "statusCode": ""
      }
    ],
    "endTime": 1556906972.847,
    "startTime": 1556906958.272
  },
  {
    "phaseType": "DOWNLOAD_SOURCE",
    "phaseStatus": "SUCCEEDED",
    "durationInSeconds": 0,
    "contexts": [
      {
        "message": "",
        "statusCode": ""
      }
    ],
    "endTime": 1556906973.552,
    "startTime": 1556906972.847
  },
  {
    "phaseType": "INSTALL",
    "phaseStatus": "SUCCEEDED",
    "durationInSeconds": 0,
    "contexts": [
      {
        "message": "",
        "statusCode": ""
      }
    ],
    "endTime": 1556906973.75,
    "startTime": 1556906973.552
  },
  {
    "phaseType": "PRE_BUILD",
    "phaseStatus": "SUCCEEDED",
```

```
    "durationInSeconds": 0,
    "contexts": [
      {
        "message": "",
        "statusCode": ""
      }
    ],
    "endTime": 1556906973.937,
    "startTime": 1556906973.75
  },
  {
    "durationInSeconds": 0,
    "phaseType": "BUILD",
    "endTime": 1556906974.781,
    "phaseStatus": "STOPPED",
    "startTime": 1556906973.937
  },
  {
    "phaseType": "COMPLETED",
    "startTime": 1556906974.781
  }
],
"artifacts": {
  "location": "arn:aws:s3:::artifacts-override/my-demo-project",
  "encryptionDisabled": false,
  "overrideArtifactName": true
},
"buildComplete": true,
"buildStatus": "STOPPED",
"encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
"serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role",
"queuedTimeoutInMinutes": 5,
"timeoutInMinutes": 60,
"environment": {
  "type": "LINUX_CONTAINER",
  "environmentVariables": [],
  "computeType": "BUILD_GENERAL1_MEDIUM",
  "privilegedMode": false,
  "image": "aws/codebuild/standard:1.0",
  "imagePullCredentialsType": "CODEBUILD"
},
"logs": {
  "streamName": "1a2b3c4d-5678-90ab-cdef-11111EXAMPLE",
```



```

        "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logEvent:group=/aws/codebuild/my-demo-project;stream=1a2b3c4d-5678-90ab-
cdef-11111EXAMPLE",
        "groupName": "/aws/codebuild/my-demo-project"
    },
    "arn": "arn:aws:codebuild:us-west-2:123456789012:build/my-demo-
project:1a2b3c4d-5678-90ab-cdef-11111EXAMPLE"
}
}

```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的[停止构建 \(AWS CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[StopBuild](#)中的。

## update-project

以下代码示例演示了如何使用 update-project。

### AWS CLI

更改 AWS CodeBuild 构建项目的设置。

以下 update-project 示例更改了名为的指定 CodeBuild 构建项目的设置 my-demo-project。

```

aws codebuild update-project --name "my-demo-project" \
  --description "This project is updated" \
  --source "{\"type\": \"S3\", \"location\": \"codebuild-us-west-2-123456789012-
input-bucket/my-source-2.zip\"}" \
  --artifacts "{\"type\": \"S3\", \"location\": \"codebuild-us-west-2-123456789012-
output-bucket-2\"}" \
  --environment "{\"type\": \"LINUX_CONTAINER\", \"image\": \"aws/codebuild/
standard:1.0\", \"computeType\": \"BUILD_GENERAL1_MEDIUM\"}" \
  --service-role "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role"

```

输出显示更新的设置。

```

{
  "project": {
    "arn": "arn:aws:codebuild:us-west-2:123456789012:project/my-demo-project",
    "environment": {
      "privilegedMode": false,

```

```
    "environmentVariables": [],
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:1.0",
    "computeType": "BUILD_GENERAL1_MEDIUM",
    "imagePullCredentialsType": "CODEBUILD"
  },
  "queuedTimeoutInMinutes": 480,
  "description": "This project is updated",
  "artifacts": {
    "packaging": "NONE",
    "name": "my-demo-project",
    "type": "S3",
    "namespaceType": "NONE",
    "encryptionDisabled": false,
    "location": "codebuild-us-west-2-123456789012-output-bucket-2"
  },
  "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
  "badge": {
    "badgeEnabled": false
  },
  "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role",
  "lastModified": 1556840545.967,
  "tags": [],
  "timeoutInMinutes": 60,
  "created": 1556839783.274,
  "name": "my-demo-project",
  "cache": {
    "type": "NO_CACHE"
  },
  "source": {
    "type": "S3",
    "insecureSsl": false,
    "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source-2.zip"
  }
}
```

有关更多信息，请参阅 [《AWS CodeBuild 用户指南》](#) 中的“更改构建项目的设置 (AWS CLI)”

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateProject](#) 中的。

## update-report-group

以下代码示例演示了如何使用 update-report-group。

### AWS CLI

更新中的报告组 AWS CodeBuild。

以下update-report-group示例将报告组的导出类型更改为“NO\_EXPORT”。

```
aws codebuild update-report-group \
  --arn arn:aws:codebuild:<region-ID>:<user-ID>:report-group/cli-created-report-
  group \
  --export-config="exportConfigType=NO_EXPORT"
```

输出：

```
{
  "reportGroup": {
    "arn": "arn:aws:codebuild:<region-ID>:<user-ID>:report-group/cli-created-
    report-group",
    "name": "cli-created-report-group",
    "type": "TEST",
    "exportConfig": {
      "exportConfigType": "NO_EXPORT"
    },
    "created": 1602020686.009,
    "lastModified": 1602021033.454,
    "tags": []
  }
}
```

有关更多信息，请参阅《AWS CodeBuild 用户指南》中的使用[报告组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateReportGroup](#)中的。

## update-webhook

以下代码示例演示了如何使用 update-webhook。

### AWS CLI

更新项目的 webhook AWS CodeBuild

以下update-webhook示例使用两个筛选器组更新指定 CodeBuild 项目的 webhook。该--rotate-secret参数指定每次代码更改触发生成时 GitHub 轮换项目的密钥。第一个筛选条件组使用与正则表达式 ^refs/heads/master\$ 匹配的 Git 引用名称以及与 ^refs/heads/myBranch\$ 匹配的头部引用，指定在分支上创建、更新或重新打开的拉取请求。第二个筛选器组在 Git 引用名称与正则表达式不匹配的分支上指定推送请求 ^refs/heads/myBranch\$。

```
aws codebuild update-webhook \
  --project-name Project2 \
  --rotate-secret \
  --filter-groups "[[{\\"type\\":\\"EVENT\\",\\"pattern\\":\\"PULL_REQUEST_CREATED,
  PULL_REQUEST_UPDATED, PULL_REQUEST_REOPENED\\"},{\\"type\\":\\"HEAD_REF\\",\\"pattern
  \":\\"^refs/heads/myBranch$\\",\\"excludeMatchedPattern\\":true},{\\"type\\":\\"BASE_REF
  \\",\\"pattern\\":\\"^refs/heads/master$\\",\\"excludeMatchedPattern\\":true}],[{\\"type\\":
  \\"EVENT\\",\\"pattern\\":\\"PUSH\\"},{\\"type\\":\\"HEAD_REF\\",\\"pattern\\":\\"^refs/heads/
  myBranch$\\",\\"excludeMatchedPattern\\":true}]]]"
```

输出：

```
{
  "webhook": {
    "filterGroups": [
      [
        {
          "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
          PULL_REQUEST_REOPENED",
          "type": "EVENT"
        },
        {
          "excludeMatchedPattern": true,
          "pattern": "refs/heads/myBranch$",
          "type": "HEAD_REF"
        },
        {
          "excludeMatchedPattern": true,
          "pattern": "refs/heads/master$",
          "type": "BASE_REF"
        }
      ],
      [
        {
          "pattern": "PUSH",
          "type": "EVENT"
        }
      ]
    ]
  }
}
```

```
    },
    {
      "excludeMatchedPattern": true,
      "pattern": "refs/heads/myBranch$",
      "type": "HEAD_REF"
    }
  ],
  "lastModifiedSecret": 1556312220.133
}
```

有关更多信息，请参阅 [《AWS CodeBuild 用户指南》](#) 中的“更改构建项目的设置 (AWS CLI)”

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateWebhook](#) 中的。

## CodeCommit 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 CodeCommit。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-approval-rule-template-with-repository**

以下代码示例演示了如何使用 `associate-approval-rule-template-with-repository`。

AWS CLI

将批准规则模板与存储库关联

以下 `associate-approval-rule-template-with-repository` 示例将指定的批准规则模板与名为 `MyDemoRepo` 的存储库相关联。

```
aws codecommit associate-approval-rule-template-with-repository \  
  --repository-name MyDemoRepo \  
  --approval-rule-template-name 2-approver-rule-for-main
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的将[批准规则模板与存储库关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateApprovalRuleTemplateWithRepository](#) 中的。

## **batch-associate-approval-rule-template-with-repositories**

以下代码示例演示了如何使用 `batch-associate-approval-rule-template-with-repositories`。

### AWS CLI

在单个操作中将批准规则模板与多个存储库关联

以下 `batch-associate-approval-rule-template-with-repositories` 示例将指定的批准规则模板与名为 `MyDemoRepo` 和 `MyOtherDemoRepo` 的存储库相关联。

注意：批准规则模板特定于创建这些模板的 AWS 区域。它们只能与该 AWS 区域的存储库关联。

```
aws codecommit batch-associate-approval-rule-template-with-repositories \  
  --repository-names MyDemoRepo, MyOtherDemoRepo \  
  --approval-rule-template-name 2-approver-rule-for-main
```

输出：

```
{  
  "associatedRepositoryNames": [  
    "MyDemoRepo",  
    "MyOtherDemoRepo"  
  ],  
  "errors": []  
}
```

```
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的将[批准规则模板与存储库关联](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchAssociateApprovalRuleTemplateWithRepositories](#)中的。

## batch-describe-merge-conflicts

以下代码示例演示了如何使用 batch-describe-merge-conflicts。

### AWS CLI

获取有关两个提交说明符之间合并时所有文件或文件子集中的合并冲突的信息

以下batch-describe-merge-conflicts示例确定了将名为的源分支feature-randomizationfeature与名为的存储库中main使用THREE\_WAY\_MERGE策略命名的目标分支合并时的合并冲突MyDemoRepo。

```
aws codecommit batch-describe-merge-conflicts \  
  --source-commit-specifier feature-randomizationfeature \  
  --destination-commit-specifier main \  
  --merge-option THREE_WAY_MERGE \  
  --repository-name MyDemoRepo
```

输出：

```
{  
  "conflicts": [  
    {  
      "conflictMetadata": {  
        "filePath": "readme.md",  
        "fileSizes": {  
          "source": 139,  
          "destination": 230,  
          "base": 85  
        },  
        "fileModes": {  
          "source": "NORMAL",  
          "destination": "NORMAL",  
          "base": "NORMAL"  
        },  
        "objectTypes": {
```

```

        "source": "FILE",
        "destination": "FILE",
        "base": "FILE"
    },
    "numberOfConflicts": 1,
    "isBinaryFile": {
        "source": false,
        "destination": false,
        "base": false
    },
    "contentConflict": true,
    "fileModeConflict": false,
    "objectTypeConflict": false,
    "mergeOperations": {
        "source": "M",
        "destination": "M"
    }
},
"mergeHunks": [
    {
        "isConflict": true,
        "source": {
            "startLine": 0,
            "endLine": 3,
            "hunkContent": "VGhpcyBpEXAMPLE=="
        },
        "destination": {
            "startLine": 0,
            "endLine": 1,
            "hunkContent": "VXNlIHRoEXAMPLE=="
        }
    }
]
}
],
"errors": [],
"destinationCommitId": "86958e0aEXAMPLE",
"sourceCommitId": "6ccd57fdEXAMPLE",
"baseCommitId": "767b6958EXAMPLE"
}

```

有关更多信息，请参阅AWS CodeCommit 用户指南中的[解决拉取请求中的冲突](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchDescribeMergeConflicts](#)中的。



## batch-disassociate-approval-rule-template-from-repositories

以下代码示例演示了如何使用 `batch-disassociate-approval-rule-template-from-repositories`。

### AWS CLI

在一次操作中取消批准规则模板与多个存储库的关联

以下 `batch-disassociate-approval-rule-template-from-repositories` 示例取消指定批准规则模板与名为 `MyDemoRepo` 和 `MyOtherDemoRepo` 的存储库的关联。

```
aws codecommit batch-disassociate-approval-rule-template-from-repositories \
  --repository-names MyDemoRepo, MyOtherDemoRepo \
  --approval-rule-template-name 1-approval-rule-for-all pull requests
```

输出：

```
{
  "disassociatedRepositoryNames": [
    "MyDemoRepo",
    "MyOtherDemoRepo"
  ],
  "errors": []
}
```

有关更多信息，请参阅AWS CodeCommit 用户指南中的[取消关联批准规则模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchDisassociateApprovalRuleTemplateFromRepositories](#)中的。

## batch-get-commits

以下代码示例演示了如何使用 `batch-get-commits`。

### AWS CLI

查看有关多次提交的信息

以下 `batch-get-commits` 示例显示有关指定提交的详细信息。

```
aws codecommit batch-get-commits \
```

```
--repository-name MyDemoRepo \  
--commit-ids 317f8570EXAMPLE 4c925148EXAMPLE
```

输出：

```
{  
  "commits": [  
    {  
      "additionalData": "",  
      "committer": {  
        "date": "1508280564 -0800",  
        "name": "Mary Major",  
        "email": "mary_major@example.com"  
      },  
      "author": {  
        "date": "1508280564 -0800",  
        "name": "Mary Major",  
        "email": "mary_major@example.com"  
      },  
      "commitId": "317f8570EXAMPLE",  
      "treeId": "1f330709EXAMPLE",  
      "parents": [  
        "6e147360EXAMPLE"  
      ],  
      "message": "Change variable name and add new response element"  
    },  
    {  
      "additionalData": "",  
      "committer": {  
        "date": "1508280542 -0800",  
        "name": "Li Juan",  
        "email": "li_juan@example.com"  
      },  
      "author": {  
        "date": "1508280542 -0800",  
        "name": "Li Juan",  
        "email": "li_juan@example.com"  
      },  
      "commitId": "4c925148EXAMPLE",  
      "treeId": "1f330709EXAMPLE",  
      "parents": [  
        "317f8570EXAMPLE"  
      ],  
    }  
  ]  
}
```

```
    "message": "Added new class"
  }
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[查看提交详情](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchGetCommits](#) 中的。

## batch-get-repositories

以下代码示例演示了如何使用 batch-get-repositories。

### AWS CLI

查看有关多个存储库的详细信息

此示例显示有关多个 AWS CodeCommit 存储库的详细信息。

```
aws codecommit batch-get-repositories \
  --repository-names MyDemoRepo MyOtherDemoRepo
```

输出：

```
{
  "repositoriesNotFound": [],
  "repositories": [
    {
      "creationDate": 1429203623.625,
      "defaultBranch": "main",
      "repositoryName": "MyDemoRepo",
      "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
      "lastModifiedDate": 1430783812.0869999,
      "repositoryDescription": "My demonstration repository",
      "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
      "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
      "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
      "accountId": "111111111111"
    },
    {
      "creationDate": 1429203623.627,
```

```

        "defaultBranch": "main",
        "repositoryName": "MyOtherDemoRepo",
        "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyOtherDemoRepo",
        "lastModifiedDate": 1430783812.0889999,
        "repositoryDescription": "My other demonstration repository",
        "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/repos/
MyOtherDemoRepo",
        "repositoryId": "cfc29ac4-b0cb-44dc-9990-f6f51EXAMPLE",
        "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo"
        "accountId": "111111111111"
    }
],
"repositoriesNotFound": []
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetRepositories](#)中的。

## create-approval-rule-template

以下代码示例演示了如何使用 create-approval-rule-template。

### AWS CLI

#### 创建批准规则模板

以下create-approval-rule-template示例创建了一个名为的批准规则模板2-approver-rule-for-main ``。The template requires two users who assume the role of ``CodeCommitReview，用于批准任何拉取请求，然后才能将其合并到main分支。

```

aws codecommit create-approval-rule-template \
  --approval-rule-template-name 2-approver-rule-for-main \
  --approval-rule-template-description "Requires two developers from the team to
approve the pull request if the destination branch is main" \
  --approval-rule-template-content "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\", \"NumberOfApprovalsNeeded\": 2, \"ApprovalPoolMembers\":
[\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}"

```

输出：

```
{
```

```
"approvalRuleTemplate": {
  "approvalRuleTemplateName": "2-approver-rule-for-main",
  "creationDate": 1571356106.936,
  "approvalRuleTemplateId": "dd8b17fe-EXAMPLE",
  "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",
  \"DestinationReferences\": [\"refs/heads/main\"], \"Statements\": [{\"Type
  \": \"Approvers\", \"NumberOfApprovalsNeeded\": 2, \"ApprovalPoolMembers\":
  [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
  "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
  "approvalRuleTemplateDescription": "Requires two developers from the team to
  approve the pull request if the destination branch is main",
  "lastModifiedDate": 1571356106.936,
  "ruleContentSha256": "4711b576EXAMPLE"
}
```

有关更多信息，请参阅AWS CodeCommit 用户指南中的[创建批准规则模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateApprovalRuleTemplate](#)中的。

## create-branch

以下代码示例演示了如何使用 create-branch。

### AWS CLI

#### 创建分支

此示例在 AWS CodeCommit 存储库中创建分支。该命令只在出现错误时生成输出。

命令:

```
aws codecommit create-branch --repository-name MyDemoRepo --branch-name MyNewBranch
--commit-id 317f8570EXAMPLE
```

输出:

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateBranch](#)中的。

## create-commit

以下代码示例演示了如何使用 `create-commit`。

### AWS CLI

#### 创建提交

以下 `create-commit` 示例演示如何为存储库创建初始提交，该存储库将 `readme.md` 文件添加到 `main` 分支 `MyDemoRepo` 中名为 `MyDemoRepo` 的存储库中。

```
aws codecommit create-commit \  
  --repository-name MyDemoRepo \  
  --branch-name main \  
  --put-files "filePath=readme.md,fileContent='Welcome to our team repository.'"
```

输出：

```
{  
  "filesAdded": [  
    {  
      "blobId": "5e1c309d-EXAMPLE",  
      "absolutePath": "readme.md",  
      "fileMode": "NORMAL"  
    }  
  ],  
  "commitId": "4df8b524-EXAMPLE",  
  "treeId": "55b57003-EXAMPLE",  
  "filesDeleted": [],  
  "filesUpdated": []  
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》AWS CodeCommit [中的创建提交](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateCommit](#) 中的。

## create-pull-request-approval-rule

以下代码示例演示了如何使用 `create-pull-request-approval-rule`。

### AWS CLI

#### 为拉取请求创建批准规则

以下create-pull-request-approval-rule示例为指定的拉取请求创建名Require two approved approvers为的批准规则。该规则规定，一个批准池需要两次批准。该池包括CodeCommit 通过在123456789012 AWS 账户CodeCommitReview中扮演角色进行访问的所有用户。它还包括Nikhil\_Jayashankar来自同一 AWS 账户的 IAM 用户或联合用户。

```
aws codecommit create-pull-request-approval-rule \
  --approval-rule-name "Require two approved approvers" \
  --approval-rule-content "{\"Version\": \"2018-11-08\", \"Statements\":
  [{\"Type\": \"Approvers\", \"NumberOfApprovalsNeeded\": 2, \"ApprovalPoolMembers
  \": [\"CodeCommitApprovers:123456789012:Nikhil_Jayashankar\",
  \"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}"
```

输出：

```
{
  "approvalRule": {
    "approvalRuleName": "Require two approved approvers",
    "lastModifiedDate": 1570752871.932,
    "ruleContentSha256": "7c44e6ebEXAMPLE",
    "creationDate": 1570752871.932,
    "approvalRuleId": "aac33506-EXAMPLE",
    "approvalRuleContent": "{\"Version\": \"2018-11-08\", \"Statements\":
    [{\"Type\": \"Approvers\", \"NumberOfApprovalsNeeded\": 2, \"ApprovalPoolMembers
    \": [\"CodeCommitApprovers:123456789012:Nikhil_Jayashankar\",
    \"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
    "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major"
  }
}
```

有关更多信息，请参阅AWS CodeCommit 用户指南中的[创建批准规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePullRequestApprovalRule](#)中的。

## create-pull-request

以下代码示例演示了如何使用 create-pull-request。

### AWS CLI

#### 创建拉取请求

以下 `create-pull-request` 示例创建了一个名为“发音难度分析器”的拉取请求，其描述为“请在周二之前查看这些更改”，该请求的目标是“jane-branch”源分支，并将合并到名为“”的存储库中的默认分支“main”中。AWS CodeCommit MyDemoRepo

```
aws codecommit create-pull-request \  
  --title "My Pull Request" \  
  --description "Please review these changes by Tuesday" \  
  --client-request-token 123Example \  
  --targets repositoryName=MyDemoRepo,sourceReference=MyNewBranch
```

输出：

```
{  
  "pullRequest": {  
    "approvalRules": [  
      {  
        "approvalRuleContent": "{\"Version\": \"2018-11-08\",  
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type  
\": \"Approvers\", \"NumberOfApprovalsNeeded\": 2, \"ApprovalPoolMembers\":  
[\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",  
        "approvalRuleId": "dd8b17fe-EXAMPLE",  
        "approvalRuleName": "2-approver-rule-for-main",  
        "creationDate": 1571356106.936,  
        "lastModifiedDate": 571356106.936,  
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",  
        "originApprovalRuleTemplate": {  
          "approvalRuleTemplateId": "dd3d22fe-EXAMPLE",  
          "approvalRuleTemplateName": "2-approver-rule-for-main"  
        },  
        "ruleContentSha256": "4711b576EXAMPLE"  
      },  
    ],  
    "authorArn": "arn:aws:iam::111111111111:user/Jane_Doe",  
    "description": "Please review these changes by Tuesday",  
    "title": "Pronunciation difficulty analyzer",  
    "pullRequestTargets": [  
      {  
        "destinationCommit": "5d036259EXAMPLE",  
        "destinationReference": "refs/heads/main",  
        "repositoryName": "MyDemoRepo",  
        "sourceCommit": "317f8570EXAMPLE",  
        "sourceReference": "refs/heads/jane-branch",  
        "mergeMetadata": {
```



```
        "isMerged": false
      }
    ]
  },
  "lastActivityDate": 1508962823.285,
  "pullRequestId": "42",
  "clientRequestToken": "123Example",
  "pullRequestStatus": "OPEN",
  "creationDate": 1508962823.285
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePullRequest](#)中的。

## create-repository

以下代码示例演示了如何使用 create-repository。

### AWS CLI

#### 创建存储库

此示例创建了一个存储库并将其与用户的 AWS 帐户关联。

命令:

```
aws codecommit create-repository --repository-name MyDemoRepo --repository-
description "My demonstration repository"
```

输出:

```
{
  "repositoryMetadata": {
    "repositoryName": "MyDemoRepo",
    "cloneUrlSsh": "ssh://git-codecommit.us-east-1.amazonaws.com/v1/
repos/MyDemoRepo",
    "lastModifiedDate": 1444766838.027,
    "repositoryDescription": "My demonstration repository",
    "cloneUrlHttp": "https://git-codecommit.us-east-1.amazonaws.com/v1/
repos/MyDemoRepo",
    "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
    "Arn": "arn:aws:codecommit:us-
east-1:111111111111EXAMPLE:MyDemoRepo",
```

```

    "accountId": "111111111111"
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRepository](#)中的。

## create-unreferenced-merge-commit

以下代码示例演示了如何使用 `create-unreferenced-merge-commit`。

### AWS CLI

创建表示合并两个提交说明符后结果的未引用提交

以下`create-unreferenced-merge-commit`示例创建了一个提交，该提交表示名为的源分支`bugfix-1234`与名为的存储库中`main`使用 `THREE_WAY_MERGE` 策略命名的目标分支之间的合并结果。MyDemoRepo

```

aws codecommit create-unreferenced-merge-commit \
  --source-commit-specifier bugfix-1234 \
  --destination-commit-specifier main \
  --merge-option THREE_WAY_MERGE \
  --repository-name MyDemoRepo \
  --name "Maria Garcia" \
  --email "maria_garcia@example.com" \
  --commit-message "Testing the results of this merge."

```

输出：

```

{
  "commitId": "4f178133EXAMPLE",
  "treeId": "389765daEXAMPLE"
}

```

有关更多信息，请参阅AWS CodeCommit 用户指南[中的解决拉取请求中的冲突](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUnreferencedMergeCommit](#)中的。

## credential-helper

以下代码示例演示了如何使用 `credential-helper`。

## AWS CLI

使用以下命令设置 C AWS LI 中包含的凭证助手 AWS CodeCommit

该credential-helper实用程序不是为直接从 AWS CLI 调用而设计的。相反，它旨在用作设置本地计算机的git config命令的参数。它允许 Git 在需要进行身份验证以与 CodeCommit 存储库交互时使用 HTTPS 和您的 IAM 用户证书或 Amazon EC2 实例角色的 AWS 加密签名版本。

```
git config --global credential.helper '!aws codecommit credential-helper $@'
git config --global credential.UseHttpPath true
```

输出：

```
[credential]
  helper = !aws codecommit credential-helper $@
  UseHttpPath = true
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的设置 AWS CodeCommit 使用其他方法。仔细阅读内容，然后按照以下主题之一中的步骤进行操作：《AWS CodeCommit 用户指南》中的“适用于 Linux、macOS 或 Unix 上的 HTTPS 连接”或“适用于 Windows 上的 HTTPS 连接”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CredentialHelper](#)中的。

## delete-approval-rule-template

以下代码示例演示了如何使用 delete-approval-rule-template。

### AWS CLI

删除批准规则模板

以下delete-approval-rule-template示例删除了指定的批准规则模板。

```
aws codecommit delete-approval-rule-template \
  --approval-rule-template-name 1-approver-for-all-pull-requests
```

输出：

```
{
  "approvalRuleTemplateId": "41de97b7-EXAMPLE"
}
```

有关更多信息，请参阅AWS CodeCommit 用户指南中的[删除批准规则模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteApprovalRuleTemplate](#)中的。

## delete-branch

以下代码示例演示了如何使用 delete-branch。

### AWS CLI

#### 删除分支

此示例说明如何删除 AWS CodeCommit 存储库中的分支。

命令：

```
aws codecommit delete-branch --repository-name MyDemoRepo --branch-name MyNewBranch
```

输出：

```
{
  "branch": {
    "commitId": "317f8570EXAMPLE",
    "branchName": "MyNewBranch"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBranch](#)中的。

## delete-comment-content

以下代码示例演示了如何使用 delete-comment-content。

### AWS CLI

#### 删除评论的内容

如果您创建了评论，则只能删除评论的内容。此示例演示如何删除系统生成的 ID 为的评论的 ff30b348EXAMPLEb9aa670f 内容。

```
aws codecommit delete-comment-content \
  --comment-id ff30b348EXAMPLEb9aa670f
```

输出：

```
{
  "comment": {
    "creationDate": 1508369768.142,
    "deleted": true,
    "lastModifiedDate": 1508369842.278,
    "clientRequestToken": "123Example",
    "commentId": "ff30b348EXAMPLEb9aa670f",
    "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
    "callerReactions": [],
    "reactionCounts":
      {
        "CLAP" : 1
      }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCommentContent](#)中的。

## delete-file

以下代码示例演示了如何使用 delete-file。

### AWS CLI

#### 删除文件

以下delete-file示例演示如何从名为main的存储库中名为main、最新提交 ID 为的分支c5709475EXAMPLE中删除名为MyDemoRepo的文件README.md。

```
aws codecommit delete-file \
  --repository-name MyDemoRepo \
  --branch-name main \
  --file-path README.md \
  --parent-commit-id c5709475EXAMPLE
```

输出：

```
{
  "blobId": "559b44fEXAMPLE",
  "commitId": "353cf655EXAMPLE",
```

```
"filePath": "README.md",
"treeId": "6bc824cEXAMPLE"
}
```

有关更多信息，请参阅 AWS CodeCommit API 参考指南 [AWS CodeCommit 中的编辑或删除文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteFile](#) 中的。

## delete-pull-request-approval-rule

以下代码示例演示了如何使用 delete-pull-request-approval-rule。

### AWS CLI

删除拉取请求的批准规则

以下 delete-pull-request-approval-rule 示例删除了 My Approval Rule 为指定拉取请求命名的批准规则。

```
aws codecommit delete-pull-request-approval-rule \
  --approval-rule-name "My Approval Rule" \
  --pull-request-id 15
```

输出：

```
{
  "approvalRuleId": "077d8e8a8-EXAMPLE"
}
```

有关更多信息，请参阅 AWS CodeCommit 用户指南中的 [编辑或删除批准规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeletePullRequestApprovalRule](#) 中的。

## delete-repository

以下代码示例演示了如何使用 delete-repository。

### AWS CLI

删除存储库

此示例说明如何删除 AWS CodeCommit 存储库。

**命令:**

```
aws codecommit delete-repository --repository-name MyDemoRepo
```

**输出:**

```
{
  "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRepository](#)中的。

**describe-merge-conflicts**

以下代码示例演示了如何使用 `describe-merge-conflicts`。

**AWS CLI**

获取有关合并冲突的详细信息

以下`describe-merge-conflicts`示例使用 `THREE_WAY_MERGE` 策略确定`readme.md`在指定源分支和目标分支中命名的文件的合并冲突。

```
aws codecommit describe-merge-conflicts \
  --source-commit-specifier feature-randomizationfeature \
  --destination-commit-specifier main \
  --merge-option THREE_WAY_MERGE \
  --file-path readme.md \
  --repository-name MyDemoRepo
```

**输出:**

```
{
  "conflictMetadata": {
    "filePath": "readme.md",
    "fileSizes": {
      "source": 139,
      "destination": 230,
      "base": 85
    },
    "fileModes": {
      "source": "NORMAL",
```

```
        "destination": "NORMAL",
        "base": "NORMAL"
    },
    "objectTypes": {
        "source": "FILE",
        "destination": "FILE",
        "base": "FILE"
    },
    "numberOfConflicts": 1,
    "isBinaryFile": {
        "source": false,
        "destination": false,
        "base": false
    },
    "contentConflict": true,
    "fileModeConflict": false,
    "objectTypeConflict": false,
    "mergeOperations": {
        "source": "M",
        "destination": "M"
    }
},
"mergeHunks": [
    {
        "isConflict": true,
        "source": {
            "startLine": 0,
            "endLine": 3,
            "hunkContent": "VGhpcyBpEXAMPLE="
        },
        "destination": {
            "startLine": 0,
            "endLine": 1,
            "hunkContent": "VXNlIHRoEXAMPLE="
        }
    }
],
"destinationCommitId": "86958e0aEXAMPLE",
"sourceCommitId": "6ccd57fdEXAMPLE",
"baseCommitId": "767b69580EXAMPLE"
}
```

有关更多信息，请参阅AWS CodeCommit 用户指南中的[解决拉取请求中的冲突](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeMergeConflicts](#)中的。

## describe-pull-request-events

以下代码示例演示了如何使用 `describe-pull-request-events`。

### AWS CLI

查看拉取请求中的事件

以下`describe-pull-request-events`示例检索 ID 为“8”的拉取请求的事件。

```
aws codecommit describe-pull-request-events --pull-request-id 8
```

输出：

```
{
  "pullRequestEvents": [
    {
      "pullRequestId": "8",
      "pullRequestEventType": "PULL_REQUEST_CREATED",
      "eventDate": 1510341779.53,
      "actor": "arn:aws:iam::111111111111:user/Zhang_Wei"
    },
    {
      "pullRequestStatusChangedEventMetadata": {
        "pullRequestStatus": "CLOSED"
      },
      "pullRequestId": "8",
      "pullRequestEventType": "PULL_REQUEST_STATUS_CHANGED",
      "eventDate": 1510341930.72,
      "actor": "arn:aws:iam::111111111111:user/Jane_Doe"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePullRequestEvents](#)中的。

## disassociate-approval-rule-template-from-repository

以下代码示例演示了如何使用 `disassociate-approval-rule-template-from-repository`。

## AWS CLI

### 取消批准规则模板与存储库的关联

以下`disassociate-approval-rule-template-from-repository`示例取消指定批准规则模板与名为MyDemoRepo的存储库的关联。

```
aws codecommit disassociate-approval-rule-template-from-repository \  
  --repository-name MyDemoRepo \  
  --approval-rule-template-name 1-approver-rule-for-all-pull-requests
```

此命令不生成任何输出。

有关更多信息，请参阅AWS CodeCommit 用户指南中的[取消关联批准规则模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateApprovalRuleTemplateFromRepository](#)中的。

## evaluate-pull-request-approval-rules

以下代码示例演示了如何使用 `evaluate-pull-request-approval-rules`。

### AWS CLI

#### 评估拉取请求是否满足其所有批准规则

以下`evaluate-pull-request-approval-rules`示例评估了指定拉取请求的批准规则的状态。在此示例中，拉取请求的批准规则未得到满足，因此命令的输出显示`approved`值为`false`。

```
aws codecommit evaluate-pull-request-approval-rules \  
  --pull-request-id 27 \  
  --revision-id 9f29d167EXAMPLE
```

输出：

```
{  
  "evaluation": {  
    "approved": false,  
    "approvalRulesNotSatisfied": [  
      "Require two approved approvers"  
    ],  
    "overridden": false,
```

```

    "approvalRulesSatisfied": []
  }
}

```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的[合并拉取请求](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EvaluatePullRequestApprovalRules](#)中的。

## get-approval-rule-template

以下代码示例演示了如何使用 get-approval-rule-template。

### AWS CLI

获取批准规则模板的内容

以下get-approval-rule-template示例获取名为的批准规则模板的内容1-approver-rule-for-all-pull-requests。

```

aws codecommit get-approval-rule-template \
  --approval-rule-template-name 1-approver-rule-for-all-pull-requests

```

输出：

```

{
  "approvalRuleTemplate": {
    "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\", \"Statements\": [
    [{\"Type\": \"Approvers\", \"NumberOfApprovalsNeeded\": 1, \"ApprovalPoolMembers\": [
    [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]]}]}\",
    \"ruleContentSha256\": \"621181bbEXAMPLE\",
    \"lastModifiedDate\": 1571356106.936,
    \"creationDate\": 1571356106.936,
    \"approvalRuleTemplateName\": \"1-approver-rule-for-all-pull-requests\",
    \"lastModifiedUser\": \"arn:aws:iam::123456789012:user/Li_Juan\",
    \"approvalRuleTemplateId\": \"a29abb15-EXAMPLE\",
    \"approvalRuleTemplateDescription\": \"All pull requests must be approved by
    one developer on the team.\"
  }
}

```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[管理批准规则模板](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetApprovalRuleTemplate](#)中的。

## get-blob

以下代码示例演示了如何使用 `get-blob`。

### AWS CLI

查看有关 Git blob 对象的信息

以下 `get-blob` 示例在名为 “” 的存储库中检索有关 ID 为 “2eb4af3bexAmple” 的 Git blob 的信息。  
AWS CodeCommit MyDemoRepo

```
aws codecommit get-blob --repository-name MyDemoRepo --blob-id 2eb4af3bEXAMPLE
```

输出：

```
{
  "content": "QSBcCaW5hcnkgTGFyToEXAMPLE="
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetBlob](#) 中的。

## get-branch

以下代码示例演示了如何使用 `get-branch`。

### AWS CLI

获取有关分支的信息

此示例获取有关 AWS CodeCommit 存储库中分支的信息。

命令：

```
aws codecommit get-branch --repository-name MyDemoRepo --branch-name MyNewBranch
```

输出：

```
{
  "BranchInfo": {
    "commitID": "317f8570EXAMPLE",
    "branchName": "MyNewBranch"
  }
}
```

```
}  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBranch](#)中的。

## get-comment-reactions

以下代码示例演示了如何使用 `get-comment-reactions`。

### AWS CLI

#### 查看表情符号对评论的反应

以下`get-comment-reactions`示例列出了对 ID 为的评论的所有表情符号反应`abcd1234EXAMPLEb5678efgh`。如果您的 shell 的字体支持显示表情符号版本 1.0，则会在`emoji`输出中显示表情符号。

```
aws codecommit get-comment-reactions \  
  --comment-id abcd1234EXAMPLEb5678efgh
```

输出：

```
{  
  "reactionsForComment": {  
    [  
      {  
        "reaction": {  
          "emoji": "??",  
          "shortCode": "thumbsup",  
          "unicode": "U+1F44D"  
        },  
        "users": [  
          "arn:aws:iam::123456789012:user/Li_Juan",  
          "arn:aws:iam::123456789012:user/Mary_Major",  
          "arn:aws:iam::123456789012:user/Jorge_Souza"  
        ]  
      },  
      {  
        "reaction": {  
          "emoji": "??",  
          "shortCode": "thumbsdown",  
          "unicode": "U+1F44E"  
        }  
      }  
    ]  
  }  
}
```

```

    },
    "users": [
      "arn:aws:iam::123456789012:user/Nikhil_Jayashankar"
    ]
  },
  {
    "reaction": {
      "emoji": "??",
      "shortCode": "confused",
      "unicode": "U+1F615"
    },
    "users": [
      "arn:aws:iam::123456789012:user/Saanvi_Sarkar"
    ]
  }
]
}
}
}

```

有关更多信息，请参阅 [《AWS CodeCommit 用户指南》AWS CodeCommit 中的对提交进行评论](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetCommentReactions](#) 中的。

## get-comment

以下代码示例演示了如何使用 get-comment。

### AWS CLI

#### 查看评论的详细信息

此示例演示如何查看系统生成的评论 ID 为的评论的 ff30b348EXAMPLEb9aa670f 详细信息。

```
aws codecommit get-comment \
  --comment-id ff30b348EXAMPLEb9aa670f
```

输出：

```
{
  "comment": {
    "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
    "clientRequestToken": "123Example",
    "commentId": "ff30b348EXAMPLEb9aa670f",
```

```

    "content": "Whoops - I meant to add this comment to the line, but I don't
see how to delete it.",
    "creationDate": 1508369768.142,
    "deleted": false,
    "commentId": "",
    "lastModifiedDate": 1508369842.278,
    "callerReactions": [],
    "reactionCounts":
    {
        "SMILE" : 6,
        "THUMBSUP" : 1
    }
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetComment](#)中的。

## get-comments-for-compared-commit

以下代码示例演示了如何使用 `get-comments-for-compared-commit`。

### AWS CLI

#### 查看对提交的评论

此示例演示如何查看对名为的存储库中两个提交之间的比较所做的视图评论MyDemoRepo。

```

aws codecommit get-comments-for-compared-commit \
  --repository-name MyDemoRepo \
  --before-commit-ID 6e147360EXAMPLE \
  --after-commit-id 317f8570EXAMPLE

```

#### 输出：

```

{
  "commentsForComparedCommitData": [
    {
      "afterBlobId": "1f330709EXAMPLE",
      "afterCommitId": "317f8570EXAMPLE",
      "beforeBlobId": "80906a4cEXAMPLE",
      "beforeCommitId": "6e147360EXAMPLE",
      "comments": [
        {

```

```

        "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
        "clientRequestToken": "123Example",
        "commentId": "ff30b348EXAMPLEb9aa670f",
        "content": "Whoops - I meant to add this comment to the line,
not the file, but I don't see how to delete it.",
        "creationDate": 1508369768.142,
        "deleted": false,
        "CommentId": "123abc-EXAMPLE",
        "lastModifiedDate": 1508369842.278,
        "callerReactions": [],
        "reactionCounts":
        {
            "SMILE" : 6,
            "THUMBSUP" : 1
        }
    },
    {
        "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
        "clientRequestToken": "123Example",
        "commentId": "553b509bEXAMPLE56198325",
        "content": "Can you add a test case for this?",
        "creationDate": 1508369612.240,
        "deleted": false,
        "commentId": "456def-EXAMPLE",
        "lastModifiedDate": 1508369612.240,
        "callerReactions": [],
        "reactionCounts":
        {
            "THUMBSUP" : 2
        }
    }
],
"location": {
    "filePath": "cl_sample.js",
    "filePosition": 1232,
    "relativeFileVersion": "after"
},
"repositoryName": "MyDemoRepo"
}
],
"nextToken": "exampleToken"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCommentsForComparedCommit](#)中的。



## get-comments-for-pull-request

以下代码示例演示了如何使用 `get-comments-for-pull-request`。

### AWS CLI

#### 查看拉取请求的评论

此示例演示如何在名为的仓库中查看拉取请求的评论MyDemoRepo。

```
aws codecommit get-comments-for-pull-request \  
  --repository-name MyDemoRepo \  
  --before-commit-ID 317f8570EXAMPLE \  
  --after-commit-id 5d036259EXAMPLE
```

#### 输出：

```
{  
  "commentsForPullRequestData": [  
    {  
      "afterBlobId": "1f330709EXAMPLE",  
      "afterCommitId": "5d036259EXAMPLE",  
      "beforeBlobId": "80906a4cEXAMPLE",  
      "beforeCommitId": "317f8570EXAMPLE",  
      "comments": [  
        {  
          "authorArn": "arn:aws:iam::111111111111:user/Saanvi_Sarkar",  
          "clientRequestToken": "",  
          "commentId": "abcd1234EXAMPLEb5678efgh",  
          "content": "These don't appear to be used anywhere. Can we  
remove them?",  
          "creationDate": 1508369622.123,  
          "deleted": false,  
          "lastModifiedDate": 1508369622.123,  
          "callerReactions": [],  
          "reactionCounts":  
            {  
              "THUMBSUP" : 6,  
              "CONFUSED" : 1  
            }  
        },  
        {  
          "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",  
          "clientRequestToken": "",
```

```

        "commentId": "442b498bEXAMPLE5756813",
        "content": "Good catch. I'll remove them.",
        "creationDate": 1508369829.104,
        "deleted": false,
        "lastModifiedDate": 150836912.273,
        "callerReactions": ["THUMBSUP"]
        "reactionCounts":
        {
            "THUMBSUP" : 14
        }
    },
    "location": {
        "filePath": "ahs_count.py",
        "filePosition": 367,
        "relativeFileVersion": "AFTER"
    },
    "repositoryName": "MyDemoRepo",
    "pullRequestId": "42"
}
],
"nextToken": "exampleToken"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCommentsForPullRequest](#)中的。

## get-commit

以下代码示例演示了如何使用 get-commit。

### AWS CLI

查看有关仓库中提交的信息

此示例显示了有关在名为 "" 的存储库中系统生成的 ID 为 "7e9fd3091thisisanexamplethisisanexample1" 的提交的详细信息。AWS CodeCommit MyDemoRepo

命令:

```
aws codecommit get-commit --repository-name MyDemoRepo --commit-id
7e9fd3091thisisanexamplethisisanexample1
```

输出：

```
{
  "commit": {
    "additionalData": "",
    "committer": {
      "date": "1484167798 -0800",
      "name": "Mary Major",
      "email": "mary_major@example.com"
    },
    "author": {
      "date": "1484167798 -0800",
      "name": "Mary Major",
      "email": "mary_major@example.com"
    },
    "treeId": "347a3408thisisanexampletreeidexample",
    "parents": [
      "7aa87a031thisisanexamplethisisanexample1"
    ],
    "message": "Fix incorrect variable name"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCommit](#)中的。

## get-differences

以下代码示例演示了如何使用 get-differences。

### AWS CLI

获取有关仓库中提交说明符差异的信息

此示例显示了在名为的 AWS CodeCommit 存储库中重命名的 MyDemoRepo 文件夹中有关两个提交说明符（分支、标记、HEAD 或其他完全限定引用，例如提交 ID）之间更改的视图元数据信息。该示例包括几个不需要的选项，包括--before-commit-specifier、--before-path 和--after-path，以便更全面地说明如何使用这些选项来限制结果。响应包括文件模式权限。

命令：

```
aws codecommit get-differences --repository-name MyDemoRepo --before-commit-specifier 955bba12thisisanexamplethisisanexample --after-commit-specifier
```

```
14a95463thisisanexamplethisisanexample --before-path tmp/example-folder --after-  
path tmp/renamed-folder
```

输出：

```
{  
  "differences": [  
    {  
      "afterBlob": {  
        "path": "blob.txt",  
        "blobId": "2eb4af3b1thisisanexamplethisisanexample1",  
        "mode": "100644"  
      },  
      "changeType": "M",  
      "beforeBlob": {  
        "path": "blob.txt",  
        "blobId": "bf7fcf281thisisanexamplethisisanexample1",  
        "mode": "100644"  
      }  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDifferences](#)中的。

## get-file

以下代码示例演示了如何使用 get-file。

### AWS CLI

获取存储库中文件的 base-64 编码内容 AWS CodeCommit

以下get-file示例演示如何README.md从名为的存储库中名为的分支main中获取名为 base-64 的编码内容。MyDemoRepo

```
aws codecommit get-file \  
  --repository-name MyDemoRepo \  
  --commit-specifier main \  
  --file-path README.md
```

输出：

```
{
  "blobId": "559b44fEXAMPLE",
  "commitId": "c5709475EXAMPLE",
  "fileContent": "IyBQaHVzEXAMPLE",
  "filePath": "README.md",
  "fileMode": "NORMAL",
  "fileSize": 1563
}
```

有关更多信息，请参阅 AWS CodeCommit API 参考指南[GetFile](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetFile](#)中的。

## get-folder

以下代码示例演示了如何使用 get-folder。

### AWS CLI

获取 AWS CodeCommit 存储库中文件夹的内容

以下 get-folder 示例演示如何从名为的存储库中获取顶级文件夹的内容 MyDemoRepo。

```
aws codecommit get-folder --repository-name MyDemoRepo --folder-path ""
```

输出：

```
{
  "commitId": "c5709475EXAMPLE",
  "files": [
    {
      "absolutePath": ".gitignore",
      "blobId": "74094e8bEXAMPLE",
      "fileMode": "NORMAL",
      "relativePath": ".gitignore"
    },
    {
      "absolutePath": "Gemfile",
      "blobId": "9ceb72f6EXAMPLE",
      "fileMode": "NORMAL",
      "relativePath": "Gemfile"
    },
    {
```

```

        "absolutePath": "Gemfile.lock",
        "blobId": "795c4a2aEXAMPLE",
        "fileMode": "NORMAL",
        "relativePath": "Gemfile.lock"
    },
    {
        "absolutePath": "LICENSE.txt",
        "blobId": "0c7932c8EXAMPLE",
        "fileMode": "NORMAL",
        "relativePath": "LICENSE.txt"
    },
    {
        "absolutePath": "README.md",
        "blobId": "559b44feEXAMPLE",
        "fileMode": "NORMAL",
        "relativePath": "README.md"
    }
],
"folderPath": "",
"subFolders": [
    {
        "absolutePath": "public",
        "relativePath": "public",
        "treeId": "d5e92ae3aEXAMPLE"
    },
    {
        "absolutePath": "tmp",
        "relativePath": "tmp",
        "treeId": "d564d0bcEXAMPLE"
    }
],
"subModules": [],
"symbolicLinks": [],
"treeId": "7b3c4dadEXAMPLE"
}

```

有关更多信息，请参阅 AWS CodeCommit API 参考指南 `GetFolder` 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetFolder](#) 中的。

## get-merge-commit

以下代码示例演示了如何使用 `get-merge-commit`。

## AWS CLI

获取有关合并提交的详细信息

以下`get-merge-commit`示例显示了在名为的存储库中`main`使用 `THREE_WAY_MERGE` 策略命名的`bugfix-bug1234`源分支的合并提交的详细信息，目标分支名为 `THREE_WAY_MERGE`。MyDemoRepo

```
aws codecommit get-merge-commit \  
  --source-commit-specifier bugfix-bug1234 \  
  --destination-commit-specifier main \  
  --merge-option THREE_WAY_MERGE \  
  --repository-name MyDemoRepo
```

输出：

```
{  
  "sourceCommitId": "c5709475EXAMPLE",  
  "destinationCommitId": "317f8570EXAMPLE",  
  "baseCommitId": "fb12a539EXAMPLE",  
  "mergeCommitId": "ffc4d608eEXAMPLE"  
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[查看提交详情](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMergeCommit](#)中的。

## `get-merge-conflicts`

以下代码示例演示了如何使用 `get-merge-conflicts`。

## AWS CLI

查看拉取请求是否存在合并冲突

以下`get-merge-conflicts`示例显示名为“`main`”的源分支的尖端与名为“`main`”的存储库中名为 `feature-randomizationfeature` “`main`”的目标分支之间是否存在任何合并冲突MyDemoRepo。

```
aws codecommit get-merge-conflicts \  
  --repository-name MyDemoRepo \  
  --pull-request-id 1234
```

```
--source-commit-specifier feature-randomizationfeature \  
--destination-commit-specifier main \  
--merge-option THREE_WAY_MERGE
```

输出：

```
{  
  "mergeable": false,  
  "destinationCommitId": "86958e0aEXAMPLE",  
  "sourceCommitId": "6ccd57fdEXAMPLE",  
  "baseCommitId": "767b6958EXAMPLE",  
  "conflictMetadataList": [  
    {  
      "filePath": "readme.md",  
      "fileSizes": {  
        "source": 139,  
        "destination": 230,  
        "base": 85  
      },  
      "fileModes": {  
        "source": "NORMAL",  
        "destination": "NORMAL",  
        "base": "NORMAL"  
      },  
      "objectTypes": {  
        "source": "FILE",  
        "destination": "FILE",  
        "base": "FILE"  
      },  
      "numberOfConflicts": 1,  
      "isBinaryFile": {  
        "source": false,  
        "destination": false,  
        "base": false  
      },  
      "contentConflict": true,  
      "fileModeConflict": false,  
      "objectTypeConflict": false,  
      "mergeOperations": {  
        "source": "M",  
        "destination": "M"  
      }  
    }  
  ]  
}
```



```
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMergeConflicts](#)中的。

## get-merge-options

以下代码示例演示了如何使用 `get-merge-options`。

### AWS CLI

获取有关可用于合并两个指定分支的合并选项的信息

以下`get-merge-options`示例确定了可用于合并名为的源分支和名为`bugfix-bug1234`的存储库`main`中名为的目标分支的合并选项`MyDemoRepo`。

```
aws codecommit get-merge-options \
  --source-commit-specifier bugfix-bug1234 \
  --destination-commit-specifier main \
  --repository-name MyDemoRepo
```

输出：

```
{
  "mergeOptions": [
    "FAST_FORWARD_MERGE",
    "SQUASH_MERGE",
    "THREE_WAY_MERGE"
  ],
  "sourceCommitId": "18059494EXAMPLE",
  "destinationCommitId": "ffd3311dEXAMPLE",
  "baseCommitId": "ffd3311dEXAMPLE"
}
```

有关更多信息，请参阅AWS CodeCommit 用户指南中的[解决拉取请求中的冲突](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMergeOptions](#)中的。

## get-pull-request-approval-states

以下代码示例演示了如何使用 `get-pull-request-approval-states`。

## AWS CLI

### 查看拉取请求的批准情况

以下`get-pull-request-approval-states`示例返回对指定拉取请求的批准。

```
aws codecommit get-pull-request-approval-states \  
  --pull-request-id 8 \  
  --revision-id 9f29d167EXAMPLE
```

输出：

```
{  
  "approvals": [  
    {  
      "userArn": "arn:aws:iam::123456789012:user/Mary_Major",  
      "approvalState": "APPROVE"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[查看拉取请求](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPullRequestApprovalStates](#)中的。

## `get-pull-request-override-state`

以下代码示例演示了如何使用 `get-pull-request-override-state`。

## AWS CLI

### 获取有关拉取请求的覆盖状态的信息

以下`get-pull-request-override-state`示例返回指定拉取请求的覆盖状态。在此示例中，名为 Mary Major 的用户覆盖了拉取请求的批准规则，因此输出返回的值为 `true`：

```
aws codecommit get-pull-request-override-state \  
  --pull-request-id 34 \  
  --revision-id 9f29d167EXAMPLE
```

输出：

```
{
  "overridden": true,
  "overrider": "arn:aws:iam::123456789012:user/Mary_Major"
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的[改写拉取请求的批准规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetPullRequestOverrideState](#)中的。

## get-pull-request

以下代码示例演示了如何使用 get-pull-request。

### AWS CLI

查看拉取请求的详细信息

此示例演示如何查看 ID 为的拉取请求的相关信息27。

```
aws codecommit get-pull-request \
  --pull-request-id 27
```

输出：

```
{
  "pullRequest": {
    "approvalRules": [
      {
        "approvalRuleContent": "{\"Version\": \"2018-11-08\", \"Statements\": [{\"Type\": \"Approvers\", \"NumberOfApprovalsNeeded\": 2, \"ApprovalPoolMembers\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "approvalRuleId": "dd8b17fe-EXAMPLE",
        "approvalRuleName": "2-approver-rule-for-main",
        "creationDate": 1571356106.936,
        "lastModifiedDate": 571356106.936,
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
        "ruleContentSha256": "4711b576EXAMPLE"
      }
    ],
    "lastActivityDate": 1562619583.565,
    "pullRequestTargets": [
      {
```

```

        "sourceCommit": "ca45e279EXAMPLE",
        "sourceReference": "refs/heads/bugfix-1234",
        "mergeBase": "a99f5ddbEXAMPLE",
        "destinationReference": "refs/heads/main",
        "mergeMetadata": {
            "isMerged": false
        },
        "destinationCommit": "2abfc6beEXAMPLE",
        "repositoryName": "MyDemoRepo"
    }
],
"revisionId": "e47def21EXAMPLE",
"title": "Quick fix for bug 1234",
"authorArn": "arn:aws:iam::123456789012:user/Nikhil_Jayashankar",
"clientRequestToken": "d8d7612e-EXAMPLE",
"creationDate": 1562619583.565,
"pullRequestId": "27",
"pullRequestStatus": "OPEN"
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPullRequest](#)中的。

## get-repository-triggers

以下代码示例演示了如何使用 get-repository-triggers。

### AWS CLI

获取有关存储库中触发器的信息

此示例显示有关为名为的 AWS CodeCommit 存储库配置的触发器的详细信息MyDemoRepo。

```
aws codecommit get-repository-triggers \
  --repository-name MyDemoRepo
```

输出：

```
{
  "configurationId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
  "triggers": [
    {

```

```
    "destinationArn": "arn:aws:sns:us-
east-1:111111111111:MyCodeCommitTopic",
    "branches": [
        "main",
        "preprod"
    ],
    "name": "MyFirstTrigger",
    "customData": "",
    "events": [
        "all"
    ]
},
{
    "destinationArn": "arn:aws:lambda:us-
east-1:111111111111:function:MyCodeCommitPythonFunction",
    "branches": [],
    "name": "MySecondTrigger",
    "customData": "EXAMPLE",
    "events": [
        "all"
    ]
}
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRepositoryTriggers](#)中的。

## get-repository

以下代码示例演示了如何使用 `get-repository`。

### AWS CLI

获取有关存储库的信息

此示例显示有关 AWS CodeCommit 存储库的详细信息。

```
aws codecommit get-repository \
  --repository-name MyDemoRepo
```

输出：

```
{
```

```
"repositoryMetadata": {
  "creationDate": 1429203623.625,
  "defaultBranch": "main",
  "repositoryName": "MyDemoRepo",
  "cloneUrlSsh": "ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/v1/
repos/MyDemoRepo",
  "lastModifiedDate": 1430783812.0869999,
  "repositoryDescription": "My demonstration repository",
  "cloneUrlHttp": "https://codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo",
  "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
  "Arn": "arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
  "accountId": "111111111111"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRepository](#)中的。

## list-approval-rule-templates

以下代码示例演示了如何使用 `list-approval-rule-templates`。

### AWS CLI

列出某个 AWS 区域中的所有批准规则模板

以下 `list-approval-rule-templates` 示例列出了指定区域中的所有批准规则模板。如果未将 AWS 区域指定为参数，则该命令将返回用于运行该命令的 AWS CLI 配置文件中指定的区域的批准规则模板。

```
aws codecommit list-approval-rule-templates \
  --region us-east-2
```

输出：

```
{
  "approvalRuleTemplateName": [
    "2-approver-rule-for-main",
    "1-approver-rule-for-all-pull-requests"
  ]
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[管理批准规则模板](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListApprovalRuleTemplates](#)中的。

## **list-associated-approval-rule-templates-for-repository**

以下代码示例演示了如何使用 `list-associated-approval-rule-templates-for-repository`。

### AWS CLI

列出与存储库关联的所有模板

以下`list-associated-approval-rule-templates-for-repository`示例列出了与名为的存储库关联的所有批准规则模板MyDemoRepo。

```
aws codecommit list-associated-approval-rule-templates-for-repository \  
  --repository-name MyDemoRepo
```

输出：

```
{  
  "approvalRuleTemplateName": [  
    "2-approver-rule-for-main",  
    "1-approver-rule-for-all-pull-requests"  
  ]  
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[管理批准规则模板](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAssociatedApprovalRuleTemplatesForRepository](#)中的。

## **list-branches**

以下代码示例演示了如何使用 `list-branches`。

### AWS CLI

查看分支名称列表

此示例列出了 AWS CodeCommit 存储库中的所有分支名称。

```
aws codecommit list-branches \  
  --repository-name MyDemoRepo
```

输出：

```
{  
  "branches": [  
    "MyNewBranch",  
    "main"  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBranches](#)中的。

## list-pull-requests

以下代码示例演示了如何使用 `list-pull-requests`。

### AWS CLI

查看仓库中的拉取请求列表

此示例演示如何在名为 “” 的仓库中列出 ARN 为 'arn:aws:iam::111111111111:user/li\_juan' 且状态为 “已关闭” 的 IAM 用户创建的拉取请求：AWS CodeCommit MyDemoRepo

```
aws codecommit list-pull-requests --author-arn arn:aws:iam::111111111111:user/  
Li_Juan --pull-request-status CLOSED --repository-name MyDemoRepo
```

输出：

```
{  
  "nextToken": "",  
  "pullRequestIds": ["2", "12", "16", "22", "23", "35", "30", "39", "47"]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPullRequests](#)中的。

## list-repositories-for-approval-rule-template

以下代码示例演示了如何使用 `list-repositories-for-approval-rule-template`。



## AWS CLI

列出与模板关联的所有存储库

以下 `list-repositories-for-approval-rule-template` 示例列出了与指定批准规则模板关联的所有存储库。

```
aws codecommit list-repositories-for-approval-rule-template \  
  --approval-rule-template-name 2-approver-rule-for-main
```

输出：

```
{  
  "repositoryNames": [  
    "MyDemoRepo",  
    "MyClonedRepo"  
  ]  
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[管理批准规则模板](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListRepositoriesForApprovalRuleTemplate](#) 中的。

## `list-repositories`

以下代码示例演示了如何使用 `list-repositories`。

## AWS CLI

查看存储库列表

此示例列出了与用户 AWS 账户关联的所有 AWS CodeCommit 仓库。

命令：

```
aws codecommit list-repositories
```

输出：

```
{  
  "repositories": [  

```

```
{
  "repositoryName": "MyDemoRepo"
  "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
},
{
  "repositoryName": "MyOtherDemoRepo"
  "repositoryId": "cfc29ac4-b0cb-44dc-9990-f6f51EXAMPLE"
}
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRepositories](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

#### 查看存储库的 AWS 标签

以下`list-tags-for-resource`示例列出了指定存储库的标签键和标签值。

```
aws codecommit list-tags-for-resource \
  --resource-arn arn:aws:codecommit:us-west-2:111111111111:MyDemoRepo
```

输出：

```
{
  "tags": {
    "Status": "Secret",
    "Team": "Saanvi"
  }
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[查看存储库的标签](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## merge-branches-by-fast-forward

以下代码示例演示了如何使用 `merge-branches-by-fast-forward`。

## AWS CLI

### 使用快进合并策略合并两个分支

以下merge-branches-by-fast-forward示例将指定的源分支与名MyDemoRepo为的存储库中的指定目标分支合并。

```
aws codecommit merge-branches-by-fast-forward \  
  --source-commit-specifier bugfix-bug1234 \  
  --destination-commit-specifier bugfix-bug1233 \  
  --repository-name MyDemoRepo
```

输出：

```
{  
  "commitId": "4f178133EXAMPLE",  
  "treeId": "389765daEXAMPLE"  
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[比较和合并分支](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[MergeBranchesByFastForward](#)中的。

## merge-branches-by-squash

以下代码示例演示了如何使用 merge-branches-by-squash。

## AWS CLI

### 使用 squash 合并策略合并两个分支

以下merge-branches-by-squash示例将指定的源分支与名MyDemoRepo为的存储库中的指定目标分支合并。

```
aws codecommit merge-branches-by-squash \  
  --source-commit-specifier bugfix-bug1234 \  
  --destination-commit-specifier bugfix-bug1233 \  
  --author-name "Maria Garcia" \  
  --email "maria_garcia@example.com" \  
  --commit-message "Merging two fix branches to prepare for a general patch." \  
  --repository-name MyDemoRepo
```

输出：

```
{
  "commitId": "4f178133EXAMPLE",
  "treeId": "389765daEXAMPLE"
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[比较和合并分支](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[MergeBranchesBySquash](#)中的。

## merge-branches-by-three-way

以下代码示例演示了如何使用 merge-branches-by-three-way。

### AWS CLI

使用三向合并策略合并两个分支

以下merge-branches-by-three-way示例将指定的源分支与名MyDemoRepo为的存储库中的指定目标分支合并。

```
aws codecommit merge-branches-by-three-way \
  --source-commit-specifier main \
  --destination-commit-specifier bugfix-bug1234 \
  --author-name "Jorge Souza" --email "jorge_souza@example.com" \
  --commit-message "Merging changes from main to bugfix branch before additional
testing." \
  --repository-name MyDemoRepo
```

输出：

```
{
  "commitId": "4f178133EXAMPLE",
  "treeId": "389765daEXAMPLE"
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[比较和合并分支](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[MergeBranchesByThreeWay](#)中的。

## merge-pull-request-by-fast-forward

以下代码示例演示了如何使用 merge-pull-request-by-fast-forward。

### AWS CLI

#### 合并并关闭拉取请求

此示例演示如何在名为的存储库中合并和关闭 ID 为 '47'、源提交 ID 为 '99132ab0Example' 的拉取请求。MyDemoRepo

```
aws codecommit merge-pull-request-by-fast-forward \  
  --pull-request-id 47 \  
  --source-commit-id 99132ab0EXAMPLE \  
  --repository-name MyDemoRepo
```

输出：

```
{  
  "pullRequest": {  
    "approvalRules": [  
      {  
        "approvalRuleContent": "{\"Version\": \"2018-11-08\", \"Statements\":  
[{\n\"Type\": \"Approvers\", \"NumberOfApprovalsNeeded\": 1, \"ApprovalPoolMembers\":  
[\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",  
        "approvalRuleId": "dd8b17fe-EXAMPLE",  
        "approvalRuleName": "I want one approver for this pull request",  
        "creationDate": 1571356106.936,  
        "lastModifiedDate": 571356106.936,  
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",  
        "ruleContentSha256": "4711b576EXAMPLE"  
      }  
    ],  
    "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",  
    "clientRequestToken": "",  
    "creationDate": 1508530823.142,  
    "description": "Review the latest changes and updates to the global  
variables",  
    "lastActivityDate": 1508887223.155,  
    "pullRequestId": "47",  
    "pullRequestStatus": "CLOSED",  
    "pullRequestTargets": [  
      {
```

```

        "destinationCommit": "9f31c968EXAMPLE",
        "destinationReference": "refs/heads/main",
        "mergeMetadata": {
            "isMerged": true,
            "mergedBy": "arn:aws:iam::123456789012:user/Mary_Major"
        },
        "repositoryName": "MyDemoRepo",
        "sourceCommit": "99132ab0EXAMPLE",
        "sourceReference": "refs/heads/variables-branch"
    }
],
"title": "Consolidation of global variables"
}
}

```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的[合并拉取请求](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[MergePullRequestByFastForward](#)中的。

## merge-pull-request-by-squash

以下代码示例演示了如何使用 merge-pull-request-by-squash。

### AWS CLI

使用 squash 合并策略合并拉取请求

以下 merge-pull-request-by-squash 示例在名为的存储库中使用 ACCEPT\_SOURCE 的冲突解决策略合并并关闭指定的拉取请求。MyDemoRepo

```

aws codecommit merge-pull-request-by-squash \
  --pull-request-id 47 \
  --source-commit-id 99132ab0EXAMPLE \
  --repository-name MyDemoRepo \
  --conflict-detail-level LINE_LEVEL \
  --conflict-resolution-strategy ACCEPT_SOURCE \
  --name "Jorge Souza" --email "jorge_souza@example.com" \
  --commit-message "Merging pull request 47 by squash and accepting source in
merge conflicts"

```

输出：

```
{
```

```

"pullRequest": {
  "approvalRules": [
    {
      "approvalRuleContent": "{\"Version\": \"2018-11-08\",
\\\"DestinationReferences\\\": [\\\"refs/heads/main\\\"],\\\"Statements\\\": [{\\\"Type
\\\": \\\"Approvers\\\",\\\"NumberOfApprovalsNeeded\\\": 2,\\\"ApprovalPoolMembers\\\":
[\\\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\\\"]}}]",
      "approvalRuleId": "dd8b17fe-EXAMPLE",
      "approvalRuleName": "2-approver-rule-for-main",
      "creationDate": 1571356106.936,
      "lastModifiedDate": 571356106.936,
      "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
      "originApprovalRuleTemplate": {
        "approvalRuleTemplateId": "dd8b17fe-EXAMPLE",
        "approvalRuleTemplateName": "2-approver-rule-for-main"
      },
      "ruleContentSha256": "4711b576EXAMPLE"
    }
  ],
  "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
  "clientRequestToken": "",
  "creationDate": 1508530823.142,
  "description": "Review the latest changes and updates to the global
variables",
  "lastActivityDate": 1508887223.155,
  "pullRequestId": "47",
  "pullRequestStatus": "CLOSED",
  "pullRequestTargets": [
    {
      "destinationCommit": "9f31c968EXAMPLE",
      "destinationReference": "refs/heads/main",
      "mergeMetadata": {
        "isMerged": true,
        "mergedBy": "arn:aws:iam::123456789012:user/Mary_Major"
      },
      "repositoryName": "MyDemoRepo",
      "sourceCommit": "99132ab0EXAMPLE",
      "sourceReference": "refs/heads/variables-branch"
    }
  ],
  "title": "Consolidation of global variables"
}
}

```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的[合并拉取请求](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[MergePullRequestBySquash](#)中的。

## merge-pull-request-by-three-way

以下代码示例演示了如何使用 merge-pull-request-by-three-way。

### AWS CLI

使用三向合并策略合并拉取请求

以下merge-pull-request-by-three-way示例使用名MyDemoRepo为的存储库中冲突详细信息和冲突解决策略的默认选项合并并关闭指定的拉取请求。

```
aws codecommit merge-pull-request-by-three-way \  
  --pull-request-id 47 \  
  --source-commit-id 99132ab0EXAMPLE \  
  --repository-name MyDemoRepo \  
  --name "Maria Garcia" \  
  --email "maria_garcia@example.com" \  
  --commit-message "Merging pull request 47 by three-way with default options"
```

输出：

```
{  
  "pullRequest": {  
    "approvalRules": [  
      {  
        "approvalRuleContent": "{\"Version\": \"2018-11-08\",  
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type  
\": \"Approvers\", \"NumberOfApprovalsNeeded\": 2, \"ApprovalPoolMembers\":  
[\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",  
        "approvalRuleId": "dd8b17fe-EXAMPLE",  
        "approvalRuleName": "2-approver-rule-for-main",  
        "creationDate": 1571356106.936,  
        "lastModifiedDate": 571356106.936,  
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",  
        "originApprovalRuleTemplate": {  
          "approvalRuleTemplateId": "dd8b17fe-EXAMPLE",  
          "approvalRuleTemplateName": "2-approver-rule-for-main"  
        },  
        "ruleContentSha256": "4711b576EXAMPLE"  
      }  
    ]  
  }  
}
```



```

    }
  ],
  "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
  "clientRequestToken": "",
  "creationDate": 1508530823.142,
  "description": "Review the latest changes and updates to the global
variables",
  "lastActivityDate": 1508887223.155,
  "pullRequestId": "47",
  "pullRequestStatus": "CLOSED",
  "pullRequestTargets": [
    {
      "destinationCommit": "9f31c968EXAMPLE",
      "destinationReference": "refs/heads/main",
      "mergeMetadata": {
        "isMerged": true,
        "mergedBy": "arn:aws:iam::123456789012:user/Mary_Major"
      },
      "repositoryName": "MyDemoRepo",
      "sourceCommit": "99132ab0EXAMPLE",
      "sourceReference": "refs/heads/variables-branch"
    }
  ],
  "title": "Consolidation of global variables"
}
}

```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的[合并拉取请求](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[MergePullRequestByThreeWay](#)中的。

## override-pull-request-approval-rules

以下代码示例演示了如何使用 `override-pull-request-approval-rules`。

### AWS CLI

改写拉取请求的批准规则要求

以下 `override-pull-request-approval-rules` 示例覆盖了指定拉取请求的批准规则。要改为撤消覆盖，请将 `--override-status` 参数值设置为 `REVOKE`

```
aws codecommit override-pull-request-approval-rules \
```

```
--pull-request-id 34 \  
--revision-id 927df8d8EXAMPLE \  
--override-status OVERRIDE
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的[改写拉取请求的批准规则](#)。

• 有关 API 的详细信息，请参阅AWS CLI 命令参考[OverridePullRequestApprovalRules](#)中的。

## post-comment-for-compared-commit

以下代码示例演示了如何使用 `post-comment-for-compared-commit`。

### AWS CLI

对提交创建评论

此示例演示了如何在比较名为 "Can you add a test case for this?" 的存储库中的两个提交时向 `cl_sample.js` 文件添加对变更的注释 `MyDemoRepo`。

```
aws codecommit post-comment-for-compared-commit \  
  --repository-name MyDemoRepo \  
  --before-commit-id 317f8570EXAMPLE \  
  --after-commit-id 5d036259EXAMPLE \  
  --client-request-token 123Example \  
  --content "Can you add a test case for this?" \  
  --location filePath=cl_sample.js,filePosition=1232,relativeFileVersion=AFTER
```

输出：

```
{  
  "afterBlobId": "1f330709EXAMPLE",  
  "afterCommitId": "317f8570EXAMPLE",  
  "beforeBlobId": "80906a4cEXAMPLE",  
  "beforeCommitId": "6e147360EXAMPLE",  
  "comment": {  
    "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",  
    "clientRequestToken": "",  
    "commentId": "553b509bEXAMPLE56198325",  
    "content": "Can you add a test case for this?",  
    "creationDate": 1508369612.203,  
    "deleted": false,
```

```
    "commentId": "abc123-EXAMPLE",
    "lastModifiedDate": 1508369612.203,
    "callerReactions": [],
    "reactionCounts": []
  },
  "location": {
    "filePath": "cl_sample.js",
    "filePosition": 1232,
    "relativeFileVersion": "AFTER"
  },
  "repositoryName": "MyDemoRepo"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PostCommentForComparedCommit](#)中的。

## post-comment-for-pull-request

以下代码示例演示了如何使用 `post-comment-for-pull-request`。

### AWS CLI

#### 向拉取请求添加评论

以下 `post-comment-for-pull-request` 示例添加了注释“这些似乎没有在任何地方使用。Can we remove them?”。关于在名为的存储库中 ID 为的拉取请求47中对 `ahs_count.py` 文件的更改 `MyDemoRepo`。

```
aws codecommit post-comment-for-pull-request \
  --pull-request-id "47" \
  --repository-name MyDemoRepo \
  --before-commit-id 317f8570EXAMPLE \
  --after-commit-id 5d036259EXAMPLE \
  --client-request-token 123Example \
  --content "These don't appear to be used anywhere. Can we remove them?" \
  --location filePath=ahs_count.py,filePosition=367,relativeFileVersion=AFTER
```

输出：

```
{
  "afterBlobId": "1f330709EXAMPLE",
  "afterCommitId": "5d036259EXAMPLE",
```

```

    "beforeBlobId": "80906a4cEXAMPLE",
    "beforeCommitId": "317f8570EXAMPLE",
    "comment": {
      "authorArn": "arn:aws:iam::111111111111:user/Saanvi_Sarkar",
      "clientRequestToken": "123Example",
      "commentId": "abcd1234EXAMPLEb5678efgh",
      "content": "These don't appear to be used anywhere. Can we remove
them?",
      "creationDate": 1508369622.123,
      "deleted": false,
      "CommentId": "",
      "lastModifiedDate": 1508369622.123,
      "callerReactions": [],
      "reactionCounts": []
    },
    "location": {
      "filePath": "ahs_count.py",
      "filePosition": 367,
      "relativeFileVersion": "AFTER"
    },
    "repositoryName": "MyDemoRepo",
    "pullRequestId": "47"
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PostCommentForPullRequest](#)中的。

## post-comment-reply

以下代码示例演示了如何使用 post-comment-reply。

### AWS CLI

回复提交或拉取请求中的评论

此示例演示如何将系统生成的 ID "Good catch. I'll remove them." 为的评论添加回复。abcd1234EXAMPLEb5678efgh

```

aws codecommit post-comment-reply \
  --in-reply-to abcd1234EXAMPLEb5678efgh \
  --content "Good catch. I'll remove them." \
  --client-request-token 123Example

```

输出：

```
{
  "comment": {
    "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
    "clientRequestToken": "123Example",
    "commentId": "442b498bEXAMPLE5756813",
    "content": "Good catch. I'll remove them.",
    "creationDate": 1508369829.136,
    "deleted": false,
    "CommentId": "abcd1234EXAMPLEb5678efgh",
    "lastModifiedDate": 150836912.221,
    "callerReactions": [],
    "reactionCounts": []
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PostCommentReply](#)中的。

## put-comment-reaction

以下代码示例演示了如何使用 put-comment-reaction。

### AWS CLI

使用表情符号回复对提交的评论

以下put-comment-reaction示例回复了标识为、表情符号反应值为的评论:thumbsup:。abcd1234EXAMPLEb5678efgh

```
aws codecommit put-comment-reaction \
  --comment-id abcd1234EXAMPLEb5678efgh \
  --reaction-value :thumbsup:
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS CodeCommit 用户指南》AWS CodeCommit中的对提交进行评论](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutCommentReaction](#)中的。

## put-file

以下代码示例演示了如何使用 put-file。

## AWS CLI

### 向存储库添加文件

以下 `put-file` 示例将名为 “ExampleSolution.py” 的文件添加到名为 “” 的存储库中，添加到名为 “feature-randomizationfeationfeationfeation” 的分支中，该分支的最新提交 ID 为 “4c925148Example”。MyDemoRepo

```
aws codecommit put-file \  
  --repository-name MyDemoRepo \  
  --branch-name feature-randomizationfeature \  
  --file-content file://MyDirectory/ExampleSolution.py \  
  --file-path /solutions/ExampleSolution.py \  
  --parent-commit-id 4c925148EXAMPLE \  
  --name "Maria Garcia" \  
  --email "maria_garcia@example.com" \  
  --commit-message "I added a third randomization routine."
```

输出：

```
{  
  "blobId": "2eb4af3bEXAMPLE",  
  "commitId": "317f8570EXAMPLE",  
  "treeId": "347a3408EXAMPLE"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutFile](#)中的。

## put-repository-triggers

以下代码示例演示了如何使用 `put-repository-triggers`。

### AWS CLI

#### 在存储库中添加或更新触发器

此示例演示如何使用已创建的 JSON 文件（此处名为 `MyTriggers.json`）更新名为 `MySecondTrigger` “`MyFirstTrigger`” 和 “” 的触发器，该文件包含名为的存储库的所有触发器的结构。MyDemoRepo要了解如何获取现有触发器的 JSON，请参阅 `get-repository-triggers` 命令。

```
aws codecommit put-repository-triggers \  
  --repository-name MyDemoRepo \  
  --trigger-name MySecondTrigger \  
  --trigger-structure MyTriggers.json
```

```
--repository-name MyDemoRepo file://MyTriggers.json
```

MyTriggers.json 的内容：

```
{
  "repositoryName": "MyDemoRepo",
  "triggers": [
    {
      "destinationArn": "arn:aws:sns:us-
east-1:80398EXAMPLE:MyCodeCommitTopic",
      "branches": [
        "main",
        "preprod"
      ],
      "name": "MyFirstTrigger",
      "customData": "",
      "events": [
        "all"
      ]
    },
    {
      "destinationArn": "arn:aws:lambda:us-
east-1:111111111111:function:MyCodeCommitPythonFunction",
      "branches": [],
      "name": "MySecondTrigger",
      "customData": "EXAMPLE",
      "events": [
        "all"
      ]
    }
  ]
}
```

输出：

```
{
  "configurationId": "6fa51cd8-35c1-EXAMPLE"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutRepositoryTriggers](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

向现有存储库添加 AWS 标签

以下tag-resource示例使用两个标签对指定的存储库进行标记。

```
aws codecommit tag-resource \  
  --resource-arn arn:aws:codecommit:us-west-2:111111111111:MyDemoRepo \  
  --tags Status=Secret,Team=Saanvi
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的[向存储库添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## test-repository-triggers

以下代码示例演示了如何使用 test-repository-triggers。

### AWS CLI

测试存储库中的触发器

此示例演示如何在名为的 AWS CodeCommit 存储库中测试名为 MyFirstTrigger "" 的触发器 MyDemoRepo。在此示例中，存储库中的事件会触发来自亚马逊简单通知服务 (Amazon SNS) 主题的通知。

命令:

```
aws codecommit test-repository-triggers --repository-name MyDemoRepo  
  --triggers name=MyFirstTrigger,destinationArn=arn:aws:sns:us-  
east-1:111111111111:MyCodeCommitTopic,branches=mainline,preprod,events=all
```

输出 :

```
{  
  "successfulExecutions": [  

```



```
    "MyFirstTrigger"  
  ],  
  "failedExecutions": []  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TestRepositoryTriggers](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从存储库中移除 AWS 标签

以下untag-resource示例从名为的存储库中删除带有指定密钥的标签MyDemoRepo。

```
aws codecommit untag-resource \  
  --resource-arn arn:aws:codecommit:us-west-2:111111111111:MyDemoRepo \  
  --tag-keys Status
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的[从存储库中移除标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-approval-rule-template-content

以下代码示例演示了如何使用 update-approval-rule-template-content。

### AWS CLI

更新批准规则模板的内容

以下update-approval-rule-template-content示例更改了指定批准规则模板的内容，以便为担任角色的CodeCommitReview用户重新定义批准池。

```
aws codecommit update-approval-rule-template-content \  
  --approval-rule-template-name 1-approver-rule \  
  --new-rule-content "{\"Version\": \"2018-11-08\", \"DestinationReferences\  
\": [\"refs/heads/main\"], \"Statements\": [{\"Type\": \"Approvers
```

```
\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":  
[\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}"
```

输出：

```
{  
  "approvalRuleTemplate": {  
    "creationDate": 1571352720.773,  
    "approvalRuleTemplateDescription": "Requires 1 approval for all pull  
requests from the CodeCommitReview pool",  
    "lastModifiedDate": 1571358728.41,  
    "approvalRuleTemplateId": "41de97b7-EXAMPLE",  
    "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\", \"Statements\":  
[{\"Type\": \"Approvers\", \"NumberOfApprovalsNeeded\": 1, \"ApprovalPoolMembers\":  
[\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}"",  
    "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",  
    "ruleContentSha256": "2f6c21a5EXAMPLE",  
    "lastModifiedUser": "arn:aws:iam::123456789012:user/Li_Juan"  
  }  
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[管理批准规则模板](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateApprovalRuleTemplateContent](#) 中的。

## update-approval-rule-template-description

以下代码示例演示了如何使用 update-approval-rule-template-description。

AWS CLI

### 更新批准规则模板的描述

以下 update-approval-rule-template-description 示例将指定批准规则模板的描述更改为 Requires 1 approval for all pull requests from the CodeCommitReview pool。：

```
aws codecommit update-approval-rule-template-description \  
  --approval-rule-template-name 1-approver-rule-for-all-pull-requests \  
  --approval-rule-template-description "Requires 1 approval for all pull requests  
from the CodeCommitReview pool"
```

输出：

```
{
  "approvalRuleTemplate": {
    "creationDate": 1571352720.773,
    "approvalRuleTemplateDescription": "Requires 1 approval for all pull requests
from the CodeCommitReview pool",
    "lastModifiedDate": 1571358728.41,
    "approvalRuleTemplateId": "41de97b7-EXAMPLE",
    "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\", \"Statements\":
[{\ \"Type\": \"Approvers\", \"NumberOfApprovalsNeeded\": 1, \"ApprovalPoolMembers\":
[\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}\"",
    "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
    "ruleContentSha256": "2f6c21a5EXAMPLE",
    "lastModifiedUser": "arn:aws:iam::123456789012:user/Li_Juan"
  }
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[管理批准规则模板](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateApprovalRuleTemplateDescription](#) 中的。

## update-approval-rule-template-name

以下代码示例演示了如何使用 update-approval-rule-template-name。

### AWS CLI

更新批准规则模板的名称

以下 update-approval-rule-template-name 示例将批准规则模板的名称从更改为 1-approver-rule-for-all pull-re 1-approver-rule quests`。

```
aws codecommit update-approval-rule-template-name \
  --old-approval-rule-template-name 1-approver-rule \
  --new-approval-rule-template-name 1-approver-rule-for-all-pull-requests
```

输出：

```
{
  "approvalRuleTemplate": {
```

```

    "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
    "lastModifiedDate": 1571358241.619,
    "approvalRuleTemplateId": "41de97b7-EXAMPLE",
    "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\", \"Statements\":
[{\ \"Type\": \"Approvers\", \"NumberOfApprovalsNeeded\": 1, \"ApprovalPoolMembers\":
[\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
    "creationDate": 1571352720.773,
    "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
    "approvalRuleTemplateDescription": "All pull requests must be approved by one
developer on the team.",
    "ruleContentSha256": "2f6c21a5cEXAMPLE"
  }
}

```

有关更多信息，请参阅《AWS CodeCommit 用户指南》中的“[管理批准规则模板](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateApprovalRuleTemplateName](#) 中的。

## update-comment

以下代码示例演示了如何使用 update-comment。

### AWS CLI

#### 更新对提交的评论

此示例演示如何将内容添加 "Fixed as requested. I'll update the pull request." 到 ID 为的评论中 442b498bEXAMPLE5756813。

```

aws codecommit update-comment \
  --comment-id 442b498bEXAMPLE5756813 \
  --content "Fixed as requested. I'll update the pull request."

```

输出：

```

{
  "comment": {
    "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
    "clientRequestToken": "",
    "commentId": "442b498bEXAMPLE5756813",
    "content": "Fixed as requested. I'll update the pull request.",
    "creationDate": 1508369929.783,
    "deleted": false,

```

```
    "lastModifiedDate": 1508369929.287,  
    "callerReactions": [],  
    "reactionCounts":  
      {  
        "THUMBSUP" : 2  
      }  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateComment](#)中的。

## update-default-branch

以下代码示例演示了如何使用 `update-default-branch`。

### AWS CLI

#### 更改存储库的默认分支

此示例更改了 AWS CodeCommit 存储库的默认分支。该命令只在出现错误时生成输出。

命令:

```
aws codecommit update-default-branch --repository-name MyDemoRepo --default-branch-name MyNewBranch
```

输出:

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDefaultBranch](#)中的。

## update-pull-request-approval-rule-content

以下代码示例演示了如何使用 `update-pull-request-approval-rule-content`。

### AWS CLI

#### 编辑拉取请求的批准规则

以下`update-pull-request-approval-rule-content`示例更新了她指定的批准规则，要求一个用户从包含123456789012 AWS 账户中任何 IAM 用户的批准池中获得批准。

```
aws codecommit update-pull-request-approval-rule-content \
  --pull-request-id 27 \
  --approval-rule-name "Require two approved approvers" \
  --approval-rule-content "{Version: 2018-11-08, Statements: [{Type:
  \"Approvers\", NumberOfApprovalsNeeded: 1, ApprovalPoolMembers:
  [\"CodeCommitApprovers:123456789012:user/*\"]}]}"
```

输出：

```
{
  "approvalRule": {
    "approvalRuleContent": "{Version: 2018-11-08, Statements:
    [{Type: \"Approvers\", NumberOfApprovalsNeeded: 1, ApprovalPoolMembers:
    [\"CodeCommitApprovers:123456789012:user/*\"]}]}",
    "approvalRuleId": "aac33506-EXAMPLE",
    "originApprovalRuleTemplate": {},
    "creationDate": 1570752871.932,
    "lastModifiedDate": 1570754058.333,
    "approvalRuleName": "Require two approved approvers",
    "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
    "ruleContentSha256": "cd93921cEXAMPLE",
  }
}
```

有关更多信息，请参阅AWS CodeCommit 用户指南中的[编辑或删除批准规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePullRequestApprovalRuleContent](#)中的。

## update-pull-request-approval-state

以下代码示例演示了如何使用 update-pull-request-approval-state。

### AWS CLI

#### 批准或撤销对拉取请求的批准

以下update-pull-request-approval-state示例批准了 ID 为、修订版 ID 为27的拉取请求9f29d167EXAMPLE。如果您想改为撤销批准，请将--approval-state参数值设置为。REVOKE

```
aws codecommit update-pull-request-approval-state \
```

```
--pull-request-id 27 \  
--revision-id 9f29d167EXAMPLE \  
--approval-state "APPROVE"
```

此命令不生成任何输出。

有关更多信息，请参阅AWS CodeCommit 用户指南中的[查看拉取请求](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePullRequestApprovalState](#)中的。

## update-pull-request-description

以下代码示例演示了如何使用 update-pull-request-description。

### AWS CLI

#### 更改拉取请求的描述

此示例演示如何更改 ID 为 47 的拉取请求的描述。

```
aws codecommit update-pull-request-description \  
--pull-request-id 47 \  
--description "Updated the pull request to remove unused global variable."
```

输出：

```
{  
  "pullRequest": {  
    "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",  
    "clientRequestToken": "",  
    "creationDate": 1508530823.155,  
    "description": "Updated the pull request to remove unused global variable.",  
    "lastActivityDate": 1508372423.204,  
    "pullRequestId": "47",  
    "pullRequestStatus": "OPEN",  
    "pullRequestTargets": [  
      {  
        "destinationCommit": "9f31c968EXAMPLE",  
        "destinationReference": "refs/heads/main",  
        "mergeMetadata": {  
          "isMerged": false,  
        },  
        "repositoryName": "MyDemoRepo",  
      }  
    ]  
  }  
}
```

```

        "sourceCommit": "99132ab0EXAMPLE",
        "sourceReference": "refs/heads/variables-branch"
    }
  ],
  "title": "Consolidation of global variables"
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePullRequestDescription](#)中的。

## update-pull-request-status

以下代码示例演示了如何使用 `update-pull-request-status`。

### AWS CLI

#### 更改拉取请求的状态

此示例演示如何在名为的 AWS CodeCommit 存储库CLOSED中将 ID 42 为的拉取请求的状态更改为的状态MyDemoRepo。

```

aws codecommit update-pull-request-status \
  --pull-request-id 42 \
  --pull-request-status CLOSED

```

#### 输出：

```

{
  "pullRequest": {
    "approvalRules": [
      {
        "approvalRuleContent": "{\"Version\": \"2018-11-08\", \"Statements\": [
          [
            {
              \"Type\": \"Approvers\",
              \"NumberOfApprovalsNeeded\": 2,
              \"ApprovalPoolMembers\": [
                \"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"
              ]
            }
          ]
        }\",
        "approvalRuleId": "dd8b17fe-EXAMPLE",
        "approvalRuleName": "2-approvers-needed-for-this-change",
        "creationDate": 1571356106.936,
        "lastModifiedDate": 571356106.936,
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
        "ruleContentSha256": "4711b576EXAMPLE"
      }
    ],
  },
}

```



```

    "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
    "clientRequestToken": "",
    "creationDate": 1508530823.165,
    "description": "Updated the pull request to remove unused global variable.",
    "lastActivityDate": 1508372423.12,
    "pullRequestId": "47",
    "pullRequestStatus": "CLOSED",
    "pullRequestTargets": [
      {
        "destinationCommit": "9f31c968EXAMPLE",
        "destinationReference": "refs/heads/main",
        "mergeMetadata": {
          "isMerged": false,
        },
        "repositoryName": "MyDemoRepo",
        "sourceCommit": "99132ab0EXAMPLE",
        "sourceReference": "refs/heads/variables-branch"
      }
    ],
    "title": "Consolidation of global variables"
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePullRequestStatus](#)中的。

## update-pull-request-title

以下代码示例演示了如何使用 update-pull-request-title。

### AWS CLI

#### 更改拉取请求的标题

此示例演示如何更改 ID 为的拉取请求的标题47。

```

aws codecommit update-pull-request-title \
  --pull-request-id 47 \
  --title "Consolidation of global variables - updated review"

```

输出：

```

{
  "pullRequest": {

```

```

    "approvalRules": [
      {
        "approvalRuleContent": "{\"Version\": \"2018-11-08\",
\\\"DestinationReferences\": [\\\"refs/heads/main\\\"],\\\"Statements\": [{\\\"Type
\\\": \\\"Approvers\\\",\\\"NumberOfApprovalsNeeded\": 2,\\\"ApprovalPoolMembers\":
[\\\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\\\"]}]}\",
        "approvalRuleId": "dd8b17fe-EXAMPLE",
        "approvalRuleName": "2-approver-rule-for-main",
        "creationDate": 1571356106.936,
        "lastModifiedDate": 571356106.936,
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
        "originApprovalRuleTemplate": {
          "approvalRuleTemplateId": "dd8b26gr-EXAMPLE",
          "approvalRuleTemplateName": "2-approver-rule-for-main"
        },
        "ruleContentSha256": "4711b576EXAMPLE"
      }
    ],
    "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
    "clientRequestToken": "",
    "creationDate": 1508530823.12,
    "description": "Review the latest changes and updates to the global
variables. I have updated this request with some changes, including removing some
unused variables.",
    "lastActivityDate": 1508372657.188,
    "pullRequestId": "47",
    "pullRequestStatus": "OPEN",
    "pullRequestTargets": [
      {
        "destinationCommit": "9f31c968EXAMPLE",
        "destinationReference": "refs/heads/main",
        "mergeMetadata": {
          "isMerged": false,
        },
        "repositoryName": "MyDemoRepo",
        "sourceCommit": "99132ab0EXAMPLE",
        "sourceReference": "refs/heads/variables-branch"
      }
    ],
    "title": "Consolidation of global variables - updated review"
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePullRequestTitle](#)中的。

## update-repository-description

以下代码示例演示了如何使用 `update-repository-description`。

### AWS CLI

#### 更改存储库的描述

此示例更改了 AWS CodeCommit 存储库的描述。该命令只在出现错误时生成输出。

命令:

```
aws codecommit update-repository-description --repository-name MyDemoRepo --
repository-description "This description was changed"
```

输出 :

```
None.
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateRepositoryDescription](#) 中的。

## update-repository-name

以下代码示例演示了如何使用 `update-repository-name`。

### AWS CLI

#### 更改存储库的名称

此示例更改了 AWS CodeCommit 存储库的名称。该命令只在出现错误时生成输出。更改 AWS CodeCommit 存储库的名称将更改用户连接到存储库所需的 SSH 和 HTTPS 网址。在更新连接设置之前，用户无法连接到此存储库。此外，由于存储库的 ARN 会发生更改，更改存储库名称会使依赖该存储库 ARN 的所有 IAM 用户策略失效。

命令:

```
aws codecommit update-repository-name --old-name MyDemoRepo --new-name
MyRenamedDemoRepo
```

输出 :

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRepositoryName](#)中的。

## CodeDeploy 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 CodeDeploy。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **add-tags-to-on-premises-instances**

以下代码示例演示了如何使用 add-tags-to-on-premises-instances。

AWS CLI

向本地实例添加标签

以下add-tags-to-on-premises-instances示例将同一本地实例标签 AWS CodeDeploy 中的两个本地实例关联到两个本地实例。它不会向注册本地实例 AWS CodeDeploy。

```
aws deploy add-tags-to-on-premises-instances \  
  --instance-names AssetTag12010298EX AssetTag23121309EX \  
  --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddTagsToOnPremisesInstances](#)中的。

## batch-get-application-revisions

以下代码示例演示了如何使用 batch-get-application-revisions。

### AWS CLI

检索有关应用程序修订的信息

以下batch-get-application-revisions示例检索存储在存储 GitHub 库中的有关指定修订的信息。

```
aws deploy batch-get-application-revisions \
  --application-name my-codedeploy-application \
  --revisions "[{\"githubLocation\": {\"commitId\": \
  \"fa85936EXAMPLEa31736c051f10d77297EXAMPLE\"},\"repository\": \"my-github-token/my-
  repository\"},\"revisionType\": \"GitHub\"}]"
```

输出：

```
{
  "revisions": [
    {
      "genericRevisionInfo": {
        "description": "Application revision registered by Deployment ID: d-
A1B2C3111",
        "lastUsedTime": 1556912355.884,
        "registerTime": 1556912355.884,
        "firstUsedTime": 1556912355.884,
        "deploymentGroups": []
      },
      "revisionLocation": {
        "revisionType": "GitHub",
        "githubLocation": {
          "commitId": "fa85936EXAMPLEa31736c051f10d77297EXAMPLE",
          "repository": "my-github-token/my-repository"
        }
      }
    }
  ],
  "applicationName": "my-codedeploy-application",
  "errorMessage": ""
}
```

```
}
```

有关更多信息，请参阅 AWS CodeDeploy API 参考[BatchGetApplicationRevisions](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetApplicationRevisions](#)中的。

## batch-get-applications

以下代码示例演示了如何使用 batch-get-applications。

### AWS CLI

获取有关多个应用程序的信息

以下batch-get-applications示例显示了与用户 AWS 账户关联的多个应用程序的相关信息。

```
aws deploy batch-get-applications --application-names WordPress_App MyOther_App
```

输出：

```
{
  "applicationsInfo": [
    {
      "applicationName": "WordPress_App",
      "applicationId": "d9dd6993-f171-44fa-a811-211e4EXAMPLE",
      "createTime": 1407878168.078,
      "linkedToGitHub": false
    },
    {
      "applicationName": "MyOther_App",
      "applicationId": "8ca57519-31da-42b2-9194-8bb16EXAMPLE",
      "createTime": 1407453571.63,
      "linkedToGitHub": false
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetApplications](#)中的。

## batch-get-deployment-groups

以下代码示例演示了如何使用 batch-get-deployment-groups。

## AWS CLI

检索有关一个或多个部署组的信息

以下batch-get-deployment-groups示例检索与指定 CodeDeploy 应用程序关联的两个部署组的相关信息。

```
aws deploy batch-get-deployment-groups \  
  --application-name my-codedeploy-application \  
  --deployment-group-names ["my-deployment-group-1","my-deployment-group-2"]
```

输出：

```
{  
  "deploymentGroupsInfo": [  
    {  
      "deploymentStyle": {  
        "deploymentOption": "WITHOUT_TRAFFIC_CONTROL",  
        "deploymentType": "IN_PLACE"  
      },  
      "autoRollbackConfiguration": {  
        "enabled": false  
      },  
      "onPremisesTagSet": {  
        "onPremisesTagSetList": []  
      },  
      "serviceRoleArn": "arn:aws:iam::123456789012:role/  
CodeDeployServiceRole",  
      "lastAttemptedDeployment": {  
        "endTime": 1556912366.415,  
        "status": "Failed",  
        "createTime": 1556912355.884,  
        "deploymentId": "d-A1B2C3111"  
      },  
      "autoScalingGroups": [],  
      "deploymentGroupName": "my-deployment-group-1",  
      "ec2TagSet": {  
        "ec2TagSetList": [  
          [  
            {  
              "Type": "KEY_AND_VALUE",  
              "Value": "my-EC2-instance",  
              "Key": "Name"  
            }  
          ]  
        ]  
      }  
    }  
  ]  
}
```

```
        ]
      ]
    ],
    "deploymentGroupId": "a1b2c3d4-5678-90ab-cdef-11111example",
    "triggerConfigurations": [],
    "applicationName": "my-codedeploy-application",
    "computePlatform": "Server",
    "deploymentConfigName": "CodeDeployDefault.AllAtOnce"
  },
  {
    "deploymentStyle": {
      "deploymentOption": "WITHOUT_TRAFFIC_CONTROL",
      "deploymentType": "IN_PLACE"
    },
    "autoRollbackConfiguration": {
      "enabled": false
    },
    "onPremisesTagSet": {
      "onPremisesTagSetList": []
    },
    "serviceRoleArn": "arn:aws:iam::123456789012:role/CodeDeployServiceRole",
    "autoScalingGroups": [],
    "deploymentGroupName": "my-deployment-group-2",
    "ec2TagSet": {
      "ec2TagSetList": [
        [
          {
            "Type": "KEY_AND_VALUE",
            "Value": "my-EC2-instance",
            "Key": "Name"
          }
        ]
      ]
    },
    "deploymentGroupId": "a1b2c3d4-5678-90ab-cdef-22222example",
    "triggerConfigurations": [],
    "applicationName": "my-codedeploy-application",
    "computePlatform": "Server",
    "deploymentConfigName": "CodeDeployDefault.AllAtOnce"
  }
],
"errorMessage": ""
```



```
}
```

有关更多信息，请参阅 AWS CodeDeploy API 参考[BatchGetDeploymentGroups](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[BatchGetDeploymentGroups](#)中的。

## batch-get-deployment-targets

以下代码示例演示了如何使用 batch-get-deployment-targets。

### AWS CLI

检索与部署关联的目标

以下 batch-get-deployment-targets 示例返回与指定部署关联的其中一个目标的相关信息。

```
aws deploy batch-get-deployment-targets \  
  --deployment-id "d-1A2B3C4D5" \  
  --target-ids "i-01a2b3c4d5e6f1111"
```

输出：

```
{  
  "deploymentTargets": [  
    {  
      "deploymentTargetType": "InstanceTarget",  
      "instanceTarget": {  
        "lifecycleEvents": [  
          {  
            "startTime": 1556918592.162,  
            "lifecycleEventName": "ApplicationStop",  
            "status": "Succeeded",  
            "endTime": 1556918592.247,  
            "diagnostics": {  
              "scriptName": "",  
              "errorCode": "Success",  
              "logTail": "",  
              "message": "Succeeded"  
            }  
          },  
          {  
            "startTime": 1556918593.193,  
            "lifecycleEventName": "DownloadBundle",
```

```

        "status": "Succeeded",
        "endTime": 1556918593.981,
        "diagnostics": {
            "scriptName": "",
            "errorCode": "Success",
            "logTail": "",
            "message": "Succeeded"
        }
    },
    {
        "startTime": 1556918594.805,
        "lifecycleEventName": "BeforeInstall",
        "status": "Succeeded",
        "endTime": 1556918681.807,
        "diagnostics": {
            "scriptName": "",
            "errorCode": "Success",
            "logTail": "",
            "message": "Succeeded"
        }
    }
],
    "targetArn": "arn:aws:ec2:us-west-2:123456789012:instance/i-01a2b3c4d5e6f1111",
    "deploymentId": "d-1A2B3C4D5",
    "lastUpdatedAt": 1556918687.504,
    "targetId": "i-01a2b3c4d5e6f1111",
    "status": "Succeeded"
}
]
}

```

有关更多信息，请参阅 AWS CodeDeploy API 参考[BatchGetDeploymentTargets](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[BatchGetDeploymentTargets](#)中的。

## batch-get-deployments

以下代码示例演示了如何使用 batch-get-deployments。

### AWS CLI

获取有关多个部署的信息

以下batch-get-deployments示例显示了与用户 AWS 账户关联的多个部署的相关信息。

```
aws deploy batch-get-deployments --deployment-ids d-A1B2C3111 d-A1B2C3222
```

输出：

```
{
  "deploymentsInfo": [
    {
      "applicationName": "WordPress_App",
      "status": "Failed",
      "deploymentOverview": {
        "Failed": 0,
        "InProgress": 0,
        "Skipped": 0,
        "Succeeded": 1,
        "Pending": 0
      },
      "deploymentConfigName": "CodeDeployDefault.OneAtATime",
      "creator": "user",
      "deploymentGroupName": "WordPress_DG",
      "revision": {
        "revisionType": "S3",
        "s3Location": {
          "bundleType": "zip",
          "version": "uTecLusEXAMPLEFXtfUcyfV8bEXAMPLE",
          "bucket": "CodeDeployDemoBucket",
          "key": "WordPressApp.zip"
        }
      },
      "deploymentId": "d-A1B2C3111",
      "createTime": 1408480721.9,
      "completeTime": 1408480741.822
    },
    {
      "applicationName": "MyOther_App",
      "status": "Failed",
      "deploymentOverview": {
        "Failed": 1,
        "InProgress": 0,
        "Skipped": 0,
        "Succeeded": 0,
        "Pending": 0
      }
    }
  ]
}
```

```

    },
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "creator": "user",
    "errorInformation": {
      "message": "Deployment failed: Constraint default violated: No hosts
succeeded.",
      "code": "HEALTH_CONSTRAINTS"
    },
    "deploymentGroupName": "MyOther_DG",
    "revision": {
      "revisionType": "S3",
      "s3Location": {
        "bundleType": "zip",
        "eTag": "\"dd56cfdEXAMPLE8e768f9d77fEXAMPLE\"",
        "bucket": "CodeDeployDemoBucket",
        "key": "MyOtherApp.zip"
      }
    },
    "deploymentId": "d-A1B2C3222",
    "createTime": 1409764576.589,
    "completeTime": 1409764596.101
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetDeployments](#)中的。

## batch-get-on-premises-instances

以下代码示例演示了如何使用 batch-get-on-premises-instances。

### AWS CLI

获取有关一个或多个本地实例的信息

以下batch-get-on-premises-instances示例获取有关两个本地实例的信息。

```
aws deploy batch-get-on-premises-instances --instance-names AssetTag12010298EX
AssetTag23121309EX
```

输出：

```
{
```

```

    "instanceInfos": [
      {
        "iamUserArn": "arn:aws:iam::123456789012:user/AWS/CodeDeploy/
AssetTag12010298EX",
        "tags": [
          {
            "Value": "CodeDeployDemo-OnPrem",
            "Key": "Name"
          }
        ],
        "instanceName": "AssetTag12010298EX",
        "registerTime": 1425579465.228,
        "instanceArn": "arn:aws:codedeploy:us-west-2:123456789012:instance/
AssetTag12010298EX_4IwLNI2Alh"
      },
      {
        "iamUserArn": "arn:aws:iam::123456789012:user/AWS/CodeDeploy/
AssetTag23121309EX",
        "tags": [
          {
            "Value": "CodeDeployDemo-OnPrem",
            "Key": "Name"
          }
        ],
        "instanceName": "AssetTag23121309EX",
        "registerTime": 1425595585.988,
        "instanceArn": "arn:aws:codedeploy:us-west-2:80398EXAMPLE:instance/
AssetTag23121309EX_PomUy64Was"
      }
    ]
  }

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetOnPremisesInstances](#)中的。

## continue-deployment

以下代码示例演示了如何使用 continue-deployment。

### AWS CLI

无需等待指定的等待时间即可开始重新路由流量。

以下continue-deployment示例开始重新路由来自原始环境中的实例的流量，这些实例已准备好开始将流量转移到替代环境中的实例。

```
aws deploy continue-deployment \  
  --deployment-id "d-A1B2C3111" \  
  --deployment-wait-type "READY_WAIT"
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS CodeDeploy API 参考[ContinueDeployment](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ContinueDeployment](#)中的。

## create-application

以下代码示例演示了如何使用 create-application。

### AWS CLI

#### 创建应用程序

以下create-application示例创建了一个应用程序并将其与用户的 AWS 账户关联。

```
aws deploy create-application --application-name MyOther_App
```

输出：

```
{  
  "applicationId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateApplication](#)中的。

## create-deployment-config

以下代码示例演示了如何使用 create-deployment-config。

### AWS CLI

#### 创建自定义部署配置

以下`create-deployment-config`示例创建自定义部署配置并将其与用户的 AWS 账户关联。

```
aws deploy create-deployment-config \  
  --deployment-config-name ThreeQuartersHealthy \  
  --minimum-healthy-hosts type=FLEET_PERCENT,value=75
```

输出：

```
{  
  "deploymentConfigId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDeploymentConfig](#)中的。

## create-deployment-group

以下代码示例演示了如何使用 `create-deployment-group`。

### AWS CLI

#### 创建部署组

以下`create-deployment-group`示例创建了一个部署组并将其与指定的应用程序和用户 AWS 帐户相关联。

```
aws deploy create-deployment-group \  
  --application-name WordPress_App \  
  --auto-scaling-groups CodeDeployDemo-ASG \  
  --deployment-config-name CodeDeployDefault.OneAtATime \  
  --deployment-group-name WordPress_DG \  
  --ec2-tag-filters Key=Name,Value=CodeDeployDemo,Type=KEY_AND_VALUE \  
  --service-role-arn arn:aws:iam::123456789012:role/CodeDeployDemoRole
```

输出：

```
{  
  "deploymentGroupId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDeploymentGroup](#)中的。

## create-deployment

以下代码示例演示了如何使用 create-deployment。

### AWS CLI

示例 1：使用 EC2/本地计算平台创建 CodeDeploy 部署

以下 create-deployment 示例创建了一个部署并将其与用户的 AWS 账户相关联。

```
aws deploy create-deployment \  
  --application-name WordPress_App \  
  --deployment-config-name CodeDeployDefault.OneAtATime \  
  --deployment-group-name WordPress_DG \  
  --description "My demo deployment" \  
  --s3-location  
bucket=CodeDeployDemoBucket,bundleType=zip,eTag=dd56cfdEXAMPLE8e768f9d77fEXAMPLE,key=WordPress
```

输出：

```
{  
  "deploymentId": "d-A1B2C3111"  
}
```

示例 2：使用 Amazon ECS 计算平台创建 CodeDeploy 部署

以下 create-deployment 示例使用以下两个文件来部署 Amazon ECS 服务。

create-deployment.json 文件的内容：

```
{  
  "applicationName": "ecs-deployment",  
  "deploymentGroupName": "ecs-deployment-dg",  
  "revision": {  
    "revisionType": "S3",  
    "s3Location": {  
      "bucket": "ecs-deployment-bucket",  
      "key": "appspec.yaml",  
      "bundleType": "YAML"  
    }  
  }  
}
```



反过来，该文件会appspect.yaml从名ecs-deployment-bucket为的 S3 存储桶中检索以下文件。

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:region:123456789012:task-definition/ecs-task-def:2"
        LoadBalancerInfo:
          ContainerName: "sample-app"
          ContainerPort: 80
          PlatformVersion: "LATEST"
```

命令:

```
aws deploy create-deployment \
  --cli-input-json file://create-deployment.json \
  --region us-east-1
```

输出:

```
{
  "deploymentId": "d-1234ABCDE"
}
```

有关更多信息，请参阅 AWS CodeDeploy API 参考[CreateDeployment](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDeployment](#)中的。

## delete-application

以下代码示例演示了如何使用 delete-application。

### AWS CLI

#### 删除应用程序

以下delete-application示例删除与用户 AWS 账户关联的指定应用程序。

```
aws deploy delete-application --application-name WordPress_App
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteApplication](#)中的。

## delete-deployment-config

以下代码示例演示了如何使用 delete-deployment-config。

### AWS CLI

#### 删除部署配置

以下delete-deployment-config示例删除了与用户 AWS 账户关联的自定义部署配置。

```
aws deploy delete-deployment-config --deployment-config-name ThreeQuartersHealthy
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDeploymentConfig](#)中的。

## delete-deployment-group

以下代码示例演示了如何使用 delete-deployment-group。

### AWS CLI

#### 删除部署组

以下delete-deployment-group示例删除与指定应用程序关联的部署组。

```
aws deploy delete-deployment-group \  
  --application-name WordPress_App \  
  --deployment-group-name WordPress_DG
```

输出：

```
{  
  "hooksNotCleanedUp": []
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDeploymentGroup](#)中的。

## **delete-git-hub-account-token**

以下代码示例演示了如何使用 `delete-git-hub-account-token`。

### AWS CLI

删除 GitHub 账户连接

以下`delete-git-hub-account-token`示例删除了指定 GitHub 账户的连接。

```
aws deploy delete-git-hub-account-token --token-name my-github-account
```

输出：

```
{
  "tokenName": "my-github-account"
}
```

有关更多信息，请参阅 AWS CodeDeploy API 参考[DeleteGitHubAccountToken](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteGitHubAccountToken](#)中的。

## **deregister-on-premises-instance**

以下代码示例演示了如何使用 `deregister-on-premises-instance`。

### AWS CLI

取消注册本地实例

以下`deregister-on-premises-instance`示例向注销本地实例 AWS CodeDeploy，但它不会删除与该实例关联的 IAM 用户，也不会在本实例标签中取消与 AWS CodeDeploy 该实例的关联。它也不会从实例中卸载 AWS CodeDeploy 代理，也不会从实例中删除本地配置文件。

```
aws deploy deregister-on-premises-instance --instance-name AssetTag12010298EX
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterOnPremisesInstance](#)中的。

## deregister

以下代码示例演示了如何使用 deregister。

### AWS CLI

#### 取消注册本地实例

以下deregister示例向注销本地实例。AWS CodeDeploy它不会删除与实例关联的 IAM 用户。它会在本地标签 AWS CodeDeploy 中取消与实例的关联。它不会从实例中卸载 AWS CodeDeploy 代理，也不会从实例中删除本地配置文件。

```
aws deploy deregister \  
  --instance-name AssetTag12010298EX \  
  --no-delete-iam-user \  
  --region us-west-2
```

输出：

```
Retrieving on-premises instance information... DONE  
IamUserArn: arn:aws:iam::80398EXAMPLE:user/AWS/CodeDeploy/AssetTag12010298EX  
Tags: Key=Name,Value=CodeDeployDemo-OnPrem  
Removing tags from the on-premises instance... DONE  
Deregistering the on-premises instance... DONE  
Run the following command on the on-premises instance to uninstall the codedeploy-  
agent:  
aws deploy uninstall
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的“[取消注册](#)”。

## get-application-revision

以下代码示例演示了如何使用 get-application-revision。

### AWS CLI

#### 获取有关应用程序修订的信息

以下get-application-revision示例显示与指定应用程序关联的应用程序修订版的相关信息。

```
aws deploy get-application-revision \  
  --application-name WordPress_App \  
  --s3-location  
bucket=CodeDeployDemoBucket,bundleType=zip,eTag=dd56cfdEXAMPLE8e768f9d77fEXAMPLE,key=WordPressApp.zip
```

输出：

```
{  
  "applicationName": "WordPress_App",  
  "revisionInfo": {  
    "description": "Application revision registered by Deployment ID: d-  
A1B2C3111",  
    "registerTime": 1411076520.009,  
    "deploymentGroups": "WordPress_DG",  
    "lastUsedTime": 1411076520.009,  
    "firstUsedTime": 1411076520.009  
  },  
  "revision": {  
    "revisionType": "S3",  
    "s3Location": {  
      "bundleType": "zip",  
      "eTag": "dd56cfdEXAMPLE8e768f9d77fEXAMPLE",  
      "bucket": "CodeDeployDemoBucket",  
      "key": "WordPressApp.zip"  
    }  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetApplicationRevision](#)中的。

## get-application

以下代码示例演示了如何使用 get-application。

### AWS CLI

获取有关应用程序的信息

以下get-application示例显示与用户 AWS 账户关联的应用程序的相关信息。

```
aws deploy get-application --application-name WordPress_App
```

输出：

```
{
  "application": {
    "applicationName": "WordPress_App",
    "applicationId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "createTime": 1407878168.078,
    "linkedToGitHub": false
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetApplication](#)中的。

## get-deployment-config

以下代码示例演示了如何使用 get-deployment-config。

### AWS CLI

获取有关部署配置的信息

以下get-deployment-config示例显示与用户 AWS 账户关联的部署配置的相关信息。

```
aws deploy get-deployment-config --deployment-config-name ThreeQuartersHealthy
```

输出：

```
{
  "deploymentConfigInfo": {
    "deploymentConfigId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "minimumHealthyHosts": {
      "type": "FLEET_PERCENT",
      "value": 75
    },
    "createTime": 1411081164.379,
    "deploymentConfigName": "ThreeQuartersHealthy"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDeploymentConfig](#)中的。

## get-deployment-group

以下代码示例演示了如何使用 `get-deployment-group`。

### AWS CLI

查看有关部署组的信息

以下`get-deployment-group`示例显示与指定应用程序关联的部署组的相关信息。

```
aws deploy get-deployment-group \
  --application-name WordPress_App \
  --deployment-group-name WordPress_DG
```

输出：

```
{
  "deploymentGroupInfo": {
    "applicationName": "WordPress_App",
    "autoScalingGroups": [
      "CodeDeployDemo-ASG"
    ],
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "ec2TagFilters": [
      {
        "Type": "KEY_AND_VALUE",
        "Value": "CodeDeployDemo",
        "Key": "Name"
      }
    ],
    "deploymentGroupId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "serviceRoleArn": "arn:aws:iam::123456789012:role/CodeDeployDemoRole",
    "deploymentGroupName": "WordPress_DG"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDeploymentGroup](#)中的。

## get-deployment-instance

以下代码示例演示了如何使用 `get-deployment-instance`。

## AWS CLI

获取有关部署实例的信息

以下`get-deployment-instance`示例显示与指定部署关联的部署实例的相关信息。

```
aws deploy get-deployment-instance --deployment-id d-QA4G4F9EX --instance-id i-902e9fEX
```

输出：

```
{
  "instanceSummary": {
    "instanceId": "arn:aws:ec2:us-east-1:80398EXAMPLE:instance/i-902e9fEX",
    "lifecycleEvents": [
      {
        "status": "Succeeded",
        "endTime": 1408480726.569,
        "startTime": 1408480726.437,
        "lifecycleEventName": "ApplicationStop"
      },
      {
        "status": "Succeeded",
        "endTime": 1408480728.016,
        "startTime": 1408480727.665,
        "lifecycleEventName": "DownloadBundle"
      },
      {
        "status": "Succeeded",
        "endTime": 1408480729.744,
        "startTime": 1408480729.125,
        "lifecycleEventName": "BeforeInstall"
      },
      {
        "status": "Succeeded",
        "endTime": 1408480730.979,
        "startTime": 1408480730.844,
        "lifecycleEventName": "Install"
      },
      {
        "status": "Failed",
        "endTime": 1408480732.603,
        "startTime": 1408480732.1,
```



```

        "lifecycleEventName": "AfterInstall"
      },
      {
        "status": "Skipped",
        "endTime": 1408480732.606,
        "lifecycleEventName": "ApplicationStart"
      },
      {
        "status": "Skipped",
        "endTime": 1408480732.606,
        "lifecycleEventName": "ValidateService"
      }
    ],
    "deploymentId": "d-QA4G4F9EX",
    "lastUpdatedAt": 1408480733.152,
    "status": "Failed"
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDeploymentInstance](#)中的。

## get-deployment-target

以下代码示例演示了如何使用 `get-deployment-target`。

### AWS CLI

返回有关部署目标的信息

以下`get-deployment-target`示例返回与指定部署关联的部署目标的相关信息。

```

aws deploy get-deployment-target \
  --deployment-id "d-A1B2C3111" \
  --target-id "i-a1b2c3d4e5f61111"

```

输出：

```

{
  "deploymentTarget": {
    "deploymentTargetType": "InstanceTarget",
    "instanceTarget": {
      "lastUpdatedAt": 1556918687.504,
      "targetId": "i-a1b2c3d4e5f61111",

```

```
    "targetArn": "arn:aws:ec2:us-west-2:123456789012:instance/i-
a1b2c3d4e5f611111",
    "status": "Succeeded",
    "lifecycleEvents": [
      {
        "status": "Succeeded",
        "diagnostics": {
          "errorCode": "Success",
          "message": "Succeeded",
          "logTail": "",
          "scriptName": ""
        },
        "lifecycleEventName": "ApplicationStop",
        "startTime": 1556918592.162,
        "endTime": 1556918592.247
      },
      {
        "status": "Succeeded",
        "diagnostics": {
          "errorCode": "Success",
          "message": "Succeeded",
          "logTail": "",
          "scriptName": ""
        },
        "lifecycleEventName": "DownloadBundle",
        "startTime": 1556918593.193,
        "endTime": 1556918593.981
      },
      {
        "status": "Succeeded",
        "diagnostics": {
          "errorCode": "Success",
          "message": "Succeeded",
          "logTail": "",
          "scriptName": ""
        },
        "lifecycleEventName": "BeforeInstall",
        "startTime": 1556918594.805,
        "endTime": 1556918681.807
      },
      {
        "status": "Succeeded",
        "diagnostics": {
          "errorCode": "Success",
```

```
        "message": "Succeeded",
        "logTail": "",
        "scriptName": ""
    },
    "lifecycleEventName": "Install",
    "startTime": 1556918682.696,
    "endTime": 1556918683.005
},
{
    "status": "Succeeded",
    "diagnostics": {
        "errorCode": "Success",
        "message": "Succeeded",
        "logTail": "",
        "scriptName": ""
    },
    "lifecycleEventName": "AfterInstall",
    "startTime": 1556918684.135,
    "endTime": 1556918684.216
},
{
    "status": "Succeeded",
    "diagnostics": {
        "errorCode": "Success",
        "message": "Succeeded",
        "logTail": "",
        "scriptName": ""
    },
    "lifecycleEventName": "ApplicationStart",
    "startTime": 1556918685.211,
    "endTime": 1556918685.295
},
{
    "status": "Succeeded",
    "diagnostics": {
        "errorCode": "Success",
        "message": "Succeeded",
        "logTail": "",
        "scriptName": ""
    },
    "lifecycleEventName": "ValidateService",
    "startTime": 1556918686.65,
    "endTime": 1556918686.747
}
```

```
    ],
    "deploymentId": "d-A1B2C3111"
  }
}
```

有关更多信息，请参阅 AWS CodeDeploy API 参考[GetDeploymentTarget](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDeploymentTarget](#)中的。

## get-deployment

以下代码示例演示了如何使用 get-deployment。

### AWS CLI

获取有关部署的信息

以下get-deployment示例显示与用户 AWS 账户关联的部署的相关信息。

```
aws deploy get-deployment --deployment-id d-A1B2C3123
```

输出：

```
{
  "deploymentInfo": {
    "applicationName": "WordPress_App",
    "status": "Succeeded",
    "deploymentOverview": {
      "Failed": 0,
      "InProgress": 0,
      "Skipped": 0,
      "Succeeded": 1,
      "Pending": 0
    },
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "creator": "user",
    "description": "My WordPress app deployment",
    "revision": {
      "revisionType": "S3",
      "s3Location": {
        "bundleType": "zip",
        "eTag": "\"dd56cfdEXAMPLE8e768f9d77fEXAMPLE\"",

```

```

        "bucket": "CodeDeployDemoBucket",
        "key": "WordPressApp.zip"
    }
},
"deploymentId": "d-A1B2C3123",
"deploymentGroupName": "WordPress_DG",
"createTime": 1409764576.589,
"completeTime": 1409764596.101,
"ignoreApplicationStopFailures": false
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDeployment](#)中的。

## get-on-premises-instance

以下代码示例演示了如何使用 `get-on-premises-instance`。

### AWS CLI

获取有关本地实例的信息

以下`get-on-premises-instance`示例检索有关指定本地实例的信息。

```
aws deploy get-on-premises-instance --instance-name AssetTag12010298EX
```

输出：

```

{
  "instanceInfo": {
    "iamUserArn": "arn:aws:iam::123456789012:user/AWS/CodeDeploy/AssetTag12010298EX",
    "tags": [
      {
        "Value": "CodeDeployDemo-OnPrem",
        "Key": "Name"
      }
    ],
    "instanceName": "AssetTag12010298EX",
    "registerTime": 1425579465.228,
    "instanceArn": "arn:aws:codedeploy:us-east-1:123456789012:instance/AssetTag12010298EX_4IwLNI2Alh"
  }
}

```

```
}  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetOnPremisesInstance](#) 中的。

## install

以下代码示例演示了如何使用 `install`。

### AWS CLI

#### 安装本地实例

以下 `install` 示例将本地配置文件从实例上的指定位置复制到 AWS CodeDeploy 代理希望在实例上找到该文件的位置。它还会在实例上安装 AWS CodeDeploy 代理。它不会创建任何 IAM 用户，也不会向该实例注册本地实例 AWS CodeDeploy，也不会 AWS CodeDeploy 为该实例关联任何本地实例标签。

```
aws deploy install \  
  --override-config \  
  --config-file C:\temp\codedeploy.onpremises.yml \  
  --region us-west-2 \  
  --agent-installer s3://aws-codedeploy-us-west-2/latest/codedeploy-agent.msi
```

#### 输出：

```
Creating the on-premises instance configuration file... DONE  
Installing the AWS CodeDeploy Agent... DONE
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的“[安装](#)”。

## list-application-revisions

以下代码示例演示了如何使用 `list-application-revisions`。

### AWS CLI

#### 获取有关应用程序修订的信息

以下 `list-application-revisions` 示例显示与指定应用程序关联的所有应用程序修订版的相关信息。

```
aws deploy list-application-revisions \  
  --application-name WordPress_App \  
  --s3-bucket CodeDeployDemoBucket \  
  --deployed exclude \  
  --s3-key-prefix WordPress_ \  
  --sort-by lastUsedTime \  
  --sort-order descending
```

输出：

```
{  
  "revisions": [  
    {  
      "revisionType": "S3",  
      "s3Location": {  
        "version": "uTecLusvCB_JqHFxtfUcyfV8bEXAMPLE",  
        "bucket": "CodeDeployDemoBucket",  
        "key": "WordPress_App.zip",  
        "bundleType": "zip"  
      }  
    },  
    {  
      "revisionType": "S3",  
      "s3Location": {  
        "version": "tMk.UxgDpMEVb7V187ZM6wVAWEXAMPLE",  
        "bucket": "CodeDeployDemoBucket",  
        "key": "WordPress_App_2-0.zip",  
        "bundleType": "zip"  
      }  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListApplicationRevisions](#)中的。

## list-applications

以下代码示例演示了如何使用 list-applications。

### AWS CLI

获取有关应用程序的信息

以下list-applications示例显示与用户 AWS 账户关联的所有应用程序的相关信息。

```
aws deploy list-applications
```

输出：

```
{
  "applications": [
    "WordPress_App",
    "MyOther_App"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListApplications](#)中的。

## list-deployment-configs

以下代码示例演示了如何使用 list-deployment-configs。

AWS CLI

获取有关部署配置的信息

以下list-deployment-configs示例显示与用户 AWS 账户关联的所有部署配置的相关信息。

```
aws deploy list-deployment-configs
```

输出：

```
{
  "deploymentConfigsList": [
    "ThreeQuartersHealthy",
    "CodeDeployDefault.AllAtOnce",
    "CodeDeployDefault.HalfAtATime",
    "CodeDeployDefault.OneAtATime"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDeploymentConfigs](#)中的。



## list-deployment-groups

以下代码示例演示了如何使用 `list-deployment-groups`。

### AWS CLI

获取有关部署组的信息

以下 `list-deployment-groups` 示例显示与指定应用程序关联的所有部署组的相关信息。

```
aws deploy list-deployment-groups --application-name WordPress_App
```

输出：

```
{
  "applicationName": "WordPress_App",
  "deploymentGroups": [
    "WordPress_DG",
    "WordPress_Beta_DG"
  ]
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDeploymentGroups](#) 中的。

## list-deployment-instances

以下代码示例演示了如何使用 `list-deployment-instances`。

### AWS CLI

获取有关部署实例的信息

以下 `list-deployment-instances` 示例显示与指定部署关联的所有部署实例的相关信息。

```
aws deploy list-deployment-instances \
  --deployment-id d-A1B2C3111 \
  --instance-status-filter Succeeded
```

输出：

```
{
  "instancesList": [
```

```
        "i-EXAMPLE11",  
        "i-EXAMPLE22"  
    ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDeploymentInstances](#)中的。

## list-deployment-targets

以下代码示例演示了如何使用 list-deployment-targets。

### AWS CLI

检索与部署关联的目标 ID 列表

以下list-deployment-targets示例检索与状态为“失败”或“”的部署关联的目标 ID 列表InProgress。

```
aws deploy list-deployment-targets \  
  --deployment-id "d-A1B2C3111" \  
  --target-filters "{\"TargetStatus\": [\"Failed\", \"InProgress\"]}"
```

输出：

```
{  
  "targetIds": [  
    "i-0f1558aaf90e5f1f9"  
  ]  
}
```

有关更多信息，请参阅 AWS CodeDeploy API 参考[ListDeploymentTargets](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDeploymentTargets](#)中的。

## list-deployments

以下代码示例演示了如何使用 list-deployments。

### AWS CLI

获取有关部署的信息

以下 `list-deployments` 示例显示与指定应用程序和部署组关联的所有部署的相关信息。

```
aws deploy list-deployments \  
  --application-name WordPress_App \  
  --create-time-range start=2014-08-19T00:00:00,end=2014-08-20T00:00:00 \  
  --deployment-group-name WordPress_DG \  
  --include-only-statuses Failed
```

输出：

```
{  
  "deployments": [  
    "d-EXAMPLE11",  
    "d-EXAMPLE22",  
    "d-EXAMPLE33"  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDeployments](#) 中的。

## `list-git-hub-account-token-names`

以下代码示例演示了如何使用 `list-git-hub-account-token-names`。

AWS CLI

列出已存储的 GitHub 账户连接的名称

以下 `list-git-hub-account-token-names` 示例列出了与当前 AWS 用户 GitHub 账户的已存储连接的名称。

```
aws deploy list-git-hub-account-token-names
```

输出：

```
{  
  "tokenNameList": [  
    "my-first-token",  
    "my-second-token",  
    "my-third-token"  
  ]  
}
```

```
}
```

有关更多信息，请参阅 AWS CodeDeploy API 参考[ListGitHubAccountTokenNames](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListGitHubAccountTokenNames](#)中的。

## list-on-premises-instances

以下代码示例演示了如何使用 `list-on-premises-instances`。

### AWS CLI

获取有关一个或多个本地实例的信息

以下`list-on-premises-instances`示例检索在中注册的实例的可用本地实例名称列表，这些实例还具有 AWS CodeDeploy 与该实例关联的 AWS CodeDeploy 指定本地实例标签。

```
aws deploy list-on-premises-instances \
  --registration-status Registered \
  --tag-filters Key=Name,Value=CodeDeployDemo-OnPrem,Type=KEY_AND_VALUE
```

输出：

```
{
  "instanceNames": [
    "AssetTag12010298EX"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListOnPremisesInstances](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出资源（应用程序）的标签

以下`list-tags-for-resource`示例列出了应用于中名为 `TestApp` 的应用程序的 CodeDeploy 标签。

```
aws deploy list-tags-for-resource \  
  --resource-arn arn:aws:codedeploy:us-west-2:111122223333:application:testApp
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "Type",  
      "Value": "testType"  
    },  
    {  
      "Key": "Name",  
      "Value": "testName"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS CodeDeploy 用户指南》CodeDeploy [中的为部署组添加实例标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## push

以下代码示例演示了如何使用 push。

### AWS CLI

将 AWS CodeDeploy 兼容的应用程序修订捆绑并部署到 Amazon S3

以下push示例将应用程序修订捆绑并部署到 Amazon S3，然后将该应用程序修订与指定的应用程序关联起来。

```
aws deploy push \  
  --application-name WordPress_App \  
  --description "This is my deployment" \  
  --ignore-hidden-files \  
  --s3-location s3://CodeDeployDemoBucket/WordPressApp.zip \  
  --source /tmp/MyLocalDeploymentFolder/
```

输出描述了如何使用create-deployment命令创建使用上传的应用程序修订版的部署。

To deploy with this revision, run:

```
aws deploy create-deployment --application-name WordPress_App
--deployment-config-name <deployment-config-name> --
deployment-group-name <deployment-group-name> --s3-location
bucket=CodeDeployDemoBucket,key=WordPressApp.zip,bundleType=zip,eTag="cecc9b8EXAMPLE50a6e71
```

- 有关 API 的详细信息，请参阅 [Push](#) in AWS CLI 命令参考。

## register-application-revision

以下代码示例演示了如何使用 `register-application-revision`。

### AWS CLI

注册有关已上传的应用程序修订版的信息

以下 `register-application-revision` 示例使用注册了存储在 Amazon S3 中的已上传应用程序修订的信息。AWS CodeDeploy

```
aws deploy register-application-revision \
--application-name WordPress_App \
--description "Revised WordPress application" \
--s3-location
bucket=CodeDeployDemoBucket,key=RevisedWordPressApp.zip,bundleType=zip,eTag=cecc9b8a08eac65
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterApplicationRevision](#) 中的。

## register-on-premises-instance

以下代码示例演示了如何使用 `register-on-premises-instance`。

### AWS CLI

注册本地实例

以下 `register-on-premises-instance` 示例向注册本地实例 AWS CodeDeploy。它不会创建指定的 IAM 用户，也不会将 AWS CodeDeploy 任何本地实例标签与注册实例相关联。

```
aws deploy register-on-premises-instance \
--instance-name AssetTag12010298EX \
```

```
--iam-user-arn arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser-OnPrem
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterOnPremisesInstance](#)中的。

## register

以下代码示例演示了如何使用 register。

### AWS CLI

#### 注册本地实例

以下register示例向注册本地实例 AWS CodeDeploy，将指定的本地实例标签与注册的实例关联起来，并创建可复制到该实例的本地配置文件。AWS CodeDeploy 它不会创建 IAM 用户，也不会实例上安装 AWS CodeDeploy 代理。

```
aws deploy register \  
  --instance-name AssetTag12010298EX \  
  --iam-user-arn arn:aws:iam::80398EXAMPLE:user/CodeDeployUser-OnPrem \  
  --tags Key=Name,Value=CodeDeployDemo-OnPrem \  
  --region us-west-2
```

输出：

```
Registering the on-premises instance... DONE  
Adding tags to the on-premises instance... DONE  
Copy the on-premises configuration file named codedeploy.onpremises.yml to the on-  
premises instance, and run the following command on the on-premises instance to  
install and configure the AWS CodeDeploy Agent:  
aws deploy install --config-file codedeploy.onpremises.yml
```

- 有关 API 的详细信息，请参阅在《AWS CLI 命令参考》中[注册](#)。

## remove-tags-from-on-premises-instances

以下代码示例演示了如何使用 remove-tags-from-on-premises-instances。

### AWS CLI

从一个或多个本地实例中移除标签

以下 `remove-tags-from-on-premises-instances` 示例取消中指定的本地标签与本地实例 AWS CodeDeploy 的关联。它不会取消注册中的本地实例 AWS CodeDeploy，也不会从实例中卸载 AWS CodeDeploy 代理，也不会从实例中删除本地配置文件，也不会删除与实例关联的 IAM 用户。

```
aws deploy remove-tags-from-on-premises-instances \  
  --instance-names AssetTag12010298EX AssetTag23121309EX \  
  --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RemoveTagsFromOnPremisesInstances](#) 中的。

## stop-deployment

以下代码示例演示了如何使用 `stop-deployment`。

### AWS CLI

尝试停止部署

以下 `stop-deployment` 示例尝试停止与用户 AWS 账户关联的正在进行的部署。

aws 部署停止部署——`deployment-id d-a1b2c3111`

输出：

```
{  
  "status": "Succeeded",  
  "statusMessage": "No more commands will be scheduled for execution in the  
deployment instances"  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StopDeployment](#) 中的。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。



## AWS CLI

为资源 ( 应用程序 ) 添加标签

以下tag-resource示例向中名为 TestApp 的应用程序添加了两个标签，其键名和类型以及值testName 和 testType。 CodeDeploy：

```
aws deploy tag-resource \  
  --resource-arn arn:aws:codedeploy:us-west-2:111122223333:application:testApp \  
  --tags Key=Name,Value=testName Key=Type,Value=testType
```

如果成功，此命令不会产生任何输出。

有关更多信息，请参阅《AWS CodeDeploy 用户指南》CodeDeploy中的[为部署组添加实例标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## uninstall

以下代码示例演示了如何使用 uninstall。

### AWS CLI

卸载本地实例

以下uninstall示例从本地实例卸载 AWS CodeDeploy 代理并从实例中删除本地配置文件。它不会在中取消注册实例 AWS CodeDeploy，也不会取消中的任何本地实例标签与该实例 AWS CodeDeploy 的关联，也不会删除与该实例关联的 IAM 用户。

```
aws deploy uninstall
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的[“卸载”](#)。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从资源 ( 应用程序 ) 中移除标签

以下`untag-resource`示例从中名为 `TestApp` 的应用程序中删除了两个带有“名称”和“类型”键的 `CodeDeploy` 标签。

```
aws deploy untag-resource \  
  --resource-arn arn:aws:codedeploy:us-west-2:111122223333:application:testApp \  
  --tag-keys Name Type
```

如果成功，此命令不会产生任何输出。

有关更多信息，请参阅《AWS CodeDeploy 用户指南》CodeDeploy [中的为部署组添加实例标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-application

以下代码示例演示了如何使用 `update-application`。

### AWS CLI

更改应用程序的详细信息

以下`update-application`示例更改了与用户 AWS 账户关联的应用程序的名称。

```
aws deploy update-application \  
  --application-name WordPress_App \  
  --new-application-name My_WordPress_App
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateApplication](#)中的。

## update-deployment-group

以下代码示例演示了如何使用 `update-deployment-group`。

### AWS CLI

更改有关部署组的信息

以下`update-deployment-group`示例更改了与指定应用程序关联的部署组的设置。

```
aws deploy update-deployment-group \  
  --application-name My_WordPress_App \  
  --deployment-group-name My_Deployment_Group \  
  --new-deployment-group-name My_New_Deployment_Group
```

```
--application-name WordPress_App \  
--auto-scaling-groups My_CodeDeployDemo_ASG \  
--current-deployment-group-name WordPress_DG \  
--deployment-config-name CodeDeployDefault.AllAtOnce \  
--ec2-tag-filters Key=Name,Type=KEY_AND_VALUE,Value=My_CodeDeployDemo \  
--new-deployment-group-name My_WordPress_DepGroup \  
--service-role-arn arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo-2
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDeploymentGroup](#)中的。

## CodeGuru 使用审阅者示例 AWS CLI

以下代码示例向您展示了如何使用 with CodeGuru Reviewer 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-repository**

以下代码示例演示了如何使用 associate-repository。

AWS CLI

示例 1：创建 Bitbucket 存储库关联

以下associate-repository示例使用现有的 Bitbucket 存储库创建存储库关联。

```
aws codeguru-reviewer associate-repository \  

```

```
--repository 'Bitbucket={Owner=sample-owner, Name=mySampleRepo,
ConnectionArn=arn:aws:codestar-connections:us-west-2:123456789012:connection/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 }'
```

输出：

```
{
  "RepositoryAssociation": {
    "ProviderType": "Bitbucket",
    "Name": "mySampleRepo",
    "LastUpdatedTimeStamp": 1596216896.979,
    "AssociationId": "association:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "CreatedTimeStamp": 1596216896.979,
    "ConnectionArn": "arn:aws:codestar-connections:us-
west-2:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "State": "Associating",
    "StateReason": "Pending Repository Association",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "Owner": "sample-owner"
  }
}
```

有关更多信息，请参阅 Amazon Reviewer 用户 [指南中的在 Amazon CodeGuru Reviewer 中创建 Bitbucket 存储库关联](#)。CodeGuru

示例 2：创建 GitHub 企业存储库关联

以下 associate-repository 示例使用现有的 GitHub 企业存储库创建存储库关联。

```
aws codeguru-reviewer associate-repository \
  --repository 'GitHubEnterpriseServer={Owner=sample-owner, Name=mySampleRepo,
ConnectionArn=arn:aws:codestar-connections:us-west-2:123456789012:connection/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 }'
```

输出：

```
{
  "RepositoryAssociation": {
    "ProviderType": "GitHubEnterpriseServer",
    "Name": "mySampleRepo",
    "LastUpdatedTimeStamp": 1596216896.979,
```

```

    "AssociationId": "association:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "CreatedTimeStamp": 1596216896.979,
    "ConnectionArn": "arn:aws:codestar-connections:us-
west-2:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "State": "Associating",
    "StateReason": "Pending Repository Association",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "Owner": "sample-owner"
  }
}

```

有关更多信息，请参阅 Amazon Codeguru [Reviewer 用户指南中的在 Amazon CodeGuru Reviewer 中创建 GitHub 企业服务器存储库关联](#)。

### 示例 3：创建 AWS CodeCommit 存储库关联

以下 `associate-repository` 示例使用现有存储库创建存储 AWS CodeCommit 库关联。

```

aws codeguru-reviewer associate-repository \
  --repository CodeCommit={Name=mySampleRepo}

```

输出：

```

{
  "RepositoryAssociation": {
    "AssociationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Name": "My-ecs-beta-repo",
    "LastUpdatedTimeStamp": 1595634764.029,
    "ProviderType": "CodeCommit",
    "CreatedTimeStamp": 1595634764.029,
    "Owner": "544120495673",
    "State": "Associating",
    "StateReason": "Pending Repository Association",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:544120495673:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  }
}

```

有关更多信息，请参阅 Amazon Reviewer 用户 [指南中的在 Amazon CodeGuru Reviewer 中创建 AWS CodeCommit 存储库关联](#)。CodeGuru

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateRepository](#) 中的。

## create-code-review

以下代码示例演示了如何使用 `create-code-review`。

### AWS CLI

创建代码审查。

以下内容 `create-code-review` 创建了对名为 `my-repository-name` 的 AWS CodeCommit 存储库 `mainline` 分支中的代码的审查。

```
aws codeguru-reviewer create-code-review \  
  --name my-code-review \  
  --repository-association-arn arn:aws:codeguru-reviewer:us-  
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --type '{"RepositoryAnalysis": {"RepositoryHead": {"BranchName": "mainline"}}}'
```

输出：

```
{  
  "CodeReview": {  
    "Name": "my-code-review",  
    "CodeReviewArn": "arn:aws:codeguru-reviewer:us-  
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222:code-  
review:RepositoryAnalysis-my-code-review",  
    "RepositoryName": "my-repository-name",  
    "Owner": "123456789012",  
    "ProviderType": "CodeCommit",  
    "State": "Pending",  
    "StateReason": "CodeGuru Reviewer has received the request, and a code  
review is scheduled.",  
    "CreatedTimeStamp": 1618873489.195,  
    "LastUpdatedTimeStamp": 1618873489.195,  
    "Type": "RepositoryAnalysis",  
    "SourceCodeType": {  
      "RepositoryHead": {  
        "BranchName": "mainline"  
      }  
    },  
    "AssociationArn": "arn:aws:codeguru-reviewer:us-  
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"  
  }  
}
```

有关更多信息，请参阅 Amazon Reviewer 用户指南中的在 Amazon CodeGuru Reviewer [中创建代码审查](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateCodeReview](#) 中的。

## describe-code-review

以下代码示例演示了如何使用 describe-code-review。

### AWS CLI

列出有关代码审查的详细信息。

下面 describe-code-review 列出了有关审查名为 "" 的 AWS CodeCommit 存储库 "主线" 分支中的代码的信息。my-repo-name

```
aws codeguru-reviewer put-recommendation-feedback \
  --code-review-arn arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111:code-
review:RepositoryAnalysis-my-repository-name-branch-abcdefgh12345678 \
  --recommendation-id
3be1b2e5d7ef6e298a06499379ee290c9c596cf688fdcadb08285ddb0dd390eb \
  --reactions ThumbsUp
```

### 输出

```
{
  "CodeReview": {
    "Name": "My-ecs-beta-repo-master-xs6di4kfd4j269dz",
    "CodeReviewArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222:code-
review:RepositoryAnalysis-my-repo-name",
    "RepositoryName": "My-ecs-beta-repo",
    "Owner": "123456789012",
    "ProviderType": "CodeCommit",
    "State": "Pending",
    "StateReason": "CodeGuru Reviewer is reviewing the source code.",
    "CreatedTimeStamp": 1618874226.226,
    "LastUpdatedTimeStamp": 1618874233.689,
    "Type": "RepositoryAnalysis",
    "SourceCodeType": {
      "RepositoryHead": {
```

```

        "BranchName": "mainline"
      }
    },
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  }
}

```

有关更多信息，请参阅 Amazon CodeGuru Reviewer 用户指南中的[查看代码审查详情](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeCodeReview](#)中的。

## describe-recommendation-feedback

以下代码示例演示了如何使用 describe-recommendation-feedback。

### AWS CLI

查看有关推荐反馈的信息

以下 describe-recommendation-feedback 显示了有关推荐反馈的信息。该建议有一个 ThumbsUp 回应。

```

aws codeguru-reviewer describe-recommendation-feedback \
  --code-review-arn arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111:code-
review:RepositoryAnalysis-my-repository-name-branch-abcdefgh12345678 \
  --recommendation-id
3be1b2e5d7ef6e298a06499379ee290c9c596cf688fdcadb08285ddb0dd390eb

```

输出：

```

{
  "RecommendationFeedback": {
    "CodeReviewArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111:code-
review:RepositoryAnalysis-my-repository-name-branch-abcdefgh12345678",
    "RecommendationId":
"3be1b2e5d7ef6e298a06499379ee290c9c596cf688fdcadb08285ddb0dd390eb",
    "Reactions": [
      "ThumbsUp"
    ],
  },
}

```



```
    "UserId": "aws-user-id",
    "CreatedTimeStamp": 1618877070.313,
    "LastUpdatedTimeStamp": 1618877948.881
  }
}
```

有关更多信息，请参阅《Amazon CodeGuru Reviewer 用户指南》中的 [“查看建议并提供反馈”](#) 和 [“步骤 4：提供反馈”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeRecommendationFeedback](#) 中的。

## describe-repository-association

以下代码示例演示了如何使用 describe-repository-association。

### AWS CLI

示例 1：返回有关 GitHub 存储库关联的信息

以下 describe-repository-association 示例返回有关使用 GitHub 企业存储库且处于 Associated 状态的存储库关联的信息。

```
aws codeguru-reviewer describe-repository-association \
  --association-arn arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
  "RepositoryAssociation": {
    "AssociationId": "b822717e-0711-4e8a-bada-0e738289c75e",
    "Name": "mySampleRepo",
    "LastUpdatedTimeStamp": 1588102637.649,
    "ProviderType": "GitHub",
    "CreatedTimeStamp": 1588102615.636,
    "Owner": "sample-owner",
    "State": "Associated",
    "StateReason": "Pull Request Notification configuration successful",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  }
}
```

有关更多信息，请参阅 Amazon Reviewer 用户指南中的在 [Amazon CodeGuru Reviewer 中创建 GitHub 企业服务器存储库关联](#)。CodeGuru

示例 2：返回有关存储库关联失败的信息

以下describe-repository-association示例返回有关使用 GitHub 企业存储库且处于Failed状态的存储库关联的信息。

```
aws codeguru-reviewer describe-repository-association \  
  --association-arn arn:aws:codeguru-reviewer:us-  
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "RepositoryAssociation": {  
    "ProviderType": "GitHubEnterpriseServer",  
    "Name": "mySampleRepo",  
    "LastUpdatedTimeStamp": 1596217036.892,  
    "AssociationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "CreatedTimeStamp": 1596216896.979,  
    "ConnectionArn": "arn:aws:codestar-connections:us-  
west-2:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
    "State": "Failed",  
    "StateReason": "Failed, Please retry.",  
    "AssociationArn": "arn:aws:codeguru-reviewer:us-  
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",  
    "Owner": "sample-owner"  
  }  
}
```

有关更多信息，请参阅 Amazon Reviewer 用户指南中的在 [Amazon CodeGuru Reviewer 中创建 GitHub 企业服务器存储库关联](#)。CodeGuru

示例 3：返回有关正在取消关联的存储库关联的信息

以下describe-repository-association示例返回有关使用 GitHub 企业存储库且处于Disassociating状态的存储库关联的信息。

```
aws codeguru-reviewer describe-repository-association \  
  --association-arn arn:aws:codeguru-reviewer:us-  
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
  "RepositoryAssociation": {
    "ProviderType": "GitHubEnterpriseServer",
    "Name": "mySampleRepo",
    "LastUpdatedTimeStamp": 1596217036.892,
    "AssociationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "CreatedTimeStamp": 1596216896.979,
    "ConnectionArn": "arn:aws:codestar-connections:us-
west-2:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "State": "Disassociating",
    "StateReason": "Source code access removal in progress",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "Owner": "sample-owner"
  }
}
```

有关更多信息，请参阅 Amazon Reviewer 用户指南 [中的在 Amazon CodeGuru Reviewer 中创建 GitHub 企业服务器存储库关联](#)。CodeGuru

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeRepositoryAssociation](#) 中的。

## disassociate-repository

以下代码示例演示了如何使用 disassociate-repository。

AWS CLI

取消存储库关联

以下内容 disassociate-repository 取消关联正在使用存储库的存储 AWS CodeCommit 库关联。

```
aws codeguru-reviewer disassociate-repository \
  --association-arn arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
```

```
"RepositoryAssociation": {
  "AssociationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "Name": "my-repository",
  "Owner": "123456789012",
  "ProviderType": "CodeCommit",
  "State": "Disassociating",
  "LastUpdatedTimeStamp": 1618939174.759,
  "CreatedTimeStamp": 1595636947.096
},
"Tags": {
  "Status": "Secret",
  "Team": "Saanvi"
}
}
```

有关更多信息，请参阅 [Amazon CodeGuru Reviewer 用户指南中的在 CodeGuru Reviewer 中取消关联仓库](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateRepository](#) 中的。

## list-code-reviews

以下代码示例演示了如何使用 list-code-reviews。

### AWS CLI

列出过去 90 天内您在您的 AWS 账户中创建的代码评论。

以下 list-code-reviews 示例列出了在过去 90 天内使用拉取请求创建的代码审查。

```
aws codeguru-reviewer list-code-reviews \
  --type PullRequest
```

输出：

```
{
  "CodeReviewSummaries": [
    {
      "LastUpdatedTimeStamp": 1588897288.054,
      "Name": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "ProviderType": "GitHub",
```

```
    "PullRequestId": "5",
    "MetricsSummary": {
      "MeteredLinesOfCodeCount": 24,
      "FindingsCount": 1
    },
    "CreatedTimeStamp": 1588897068.512,
    "State": "Completed",
    "CodeReviewArn": "arn:aws:codeguru-reviewer:us-west-2:123456789012:code-
review:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Owner": "sample-owner",
    "RepositoryName": "sample-repository-name",
    "Type": "PullRequest"
  },
  {
    "LastUpdatedTimeStamp": 1588869793.263,
    "Name": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "ProviderType": "GitHub",
    "PullRequestId": "4",
    "MetricsSummary": {
      "MeteredLinesOfCodeCount": 29,
      "FindingsCount": 0
    },
    "CreatedTimeStamp": 1588869575.949,
    "State": "Completed",
    "CodeReviewArn": "arn:aws:codeguru-reviewer:us-west-2:123456789012:code-
review:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "Owner": "sample-owner",
    "RepositoryName": "sample-repository-name",
    "Type": "PullRequest"
  },
  {
    "LastUpdatedTimeStamp": 1588870511.211,
    "Name": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "ProviderType": "GitHub",
    "PullRequestId": "4",
    "MetricsSummary": {
      "MeteredLinesOfCodeCount": 2,
      "FindingsCount": 0
    },
    "CreatedTimeStamp": 1588870292.425,
    "State": "Completed",
    "CodeReviewArn": "arn:aws:codeguru-reviewer:us-west-2:123456789012:code-
review:a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "Owner": "sample-owner",
```

```
    "RepositoryName": "sample-repository-name",
    "Type": "PullRequest"
  },
  {
    "LastUpdatedTimeStamp": 1588118522.452,
    "Name": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
    "ProviderType": "GitHub",
    "PullRequestId": "3",
    "MetricsSummary": {
      "MeteredLinesOfCodeCount": 29,
      "FindingsCount": 0
    },
    "CreatedTimeStamp": 1588118301.131,
    "State": "Completed",
    "CodeReviewArn": "arn:aws:codeguru-reviewer:us-west-2:123456789012:code-
review:a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
    "Owner": "sample-owner",
    "RepositoryName": "sample-repository-name",
    "Type": "PullRequest"
  },
  {
    "LastUpdatedTimeStamp": 1588112205.207,
    "Name": "a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
    "ProviderType": "GitHub",
    "PullRequestId": "2",
    "MetricsSummary": {
      "MeteredLinesOfCodeCount": 25,
      "FindingsCount": 0
    },
    "CreatedTimeStamp": 1588111987.443,
    "State": "Completed",
    "CodeReviewArn": "arn:aws:codeguru-reviewer:us-west-2:123456789012:code-
review:a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
    "Owner": "sample-owner",
    "RepositoryName": "sample-repository-name",
    "Type": "PullRequest"
  },
  {
    "LastUpdatedTimeStamp": 1588104489.981,
    "Name": "a1b2c3d4-5678-90ab-cdef-EXAMPLE66666",
    "ProviderType": "GitHub",
    "PullRequestId": "1",
    "MetricsSummary": {
      "MeteredLinesOfCodeCount": 25,
```

```

        "FindingsCount": 0
      },
      "CreatedTimeStamp": 1588104270.223,
      "State": "Completed",
      "CodeReviewArn": "arn:aws:codeguru-reviewer:us-west-2:123456789012:code-
review:a1b2c3d4-5678-90ab-cdef-EXAMPLE66666",
      "Owner": "sample-owner",
      "RepositoryName": "sample-repository-name",
      "Type": "PullRequest"
    }
  ]
}

```

有关更多信息，请参阅 Amazon CodeGuru Reviewer 用户指南中的[查看所有代码审查](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListCodeReviews](#) 中的。

## list-recommendation-feedback

以下代码示例演示了如何使用 list-recommendation-feedback。

### AWS CLI

在关联存储库中列出客户对推荐的反馈

以下 list-recommendation-feedback 列出了客户对代码审查中所有建议的反馈。此代码审查包含来自客户的一条反馈，即 ThumbsUp “”。

```

aws codeguru-reviewer list-recommendation-feedback \
  --code-review-arn arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111:code-
review:RepositoryAnalysis-my-repository-name-branch-abcdefgh12345678

```

输出：

```

{
  "RecommendationFeedbackSummaries": [
    {
      "RecommendationId":
"3be1b2e5d7ef6e298a06499379ee290c9c596cf688fdcadb08285ddb0dd390eb",
      "Reactions": [
        "ThumbsUp"
      ],
    },
  ],
}

```

```
        "UserId": "aws-user-id"
      }
    ]
  }
```

有关更多信息，请参阅 Amazon CodeGuru Reviewer 用户指南中的[步骤 4：提供反馈](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListRecommendationFeedback](#)中的。

## list-recommendations

以下代码示例演示了如何使用 list-recommendations。

### AWS CLI

列出完成代码审查的建议

以下 list-recommendations 示例列出了完成代码审查的建议。本次代码审查有一条建议。

```
aws codeguru-reviewer list-recommendations \
  --code-review-arn arn:aws:codeguru-reviewer:us-west-2:544120495673:code-
  review:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
  "RecommendationSummaries": [
    {
      "Description": "\n\n**Problem** \n You are using a `ConcurrentHashMap`,
but your usage of `containsKey()` and `get()` may not be thread-safe at lines: **63
and 64**. In between the check and the `get()` another thread can remove the key
and the `get()` will return `null`. The remove that can remove the key is at line:
**59**.\n\n**Fix** \n Consider calling `get()`, checking instead of your current
check if the returned object is `null`, and then using that object only, without
calling `get()` again.\n\n**More info** \n [View an example on GitHub](https://
github.com/apache/hadoop/blob/f16cf877e565084c66bc63605659b157c4394dc8/hadoop-tools/
hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/s3guard/S3Guard.java#L302-L304)
(external link).",
      "RecommendationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "StartLine": 63,
      "EndLine": 64,
      "FilePath": "src/main/java/com/company/sample/application/
CreateOrderThread.java"
```



```
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon CodeGuru Reviewer 用户指南中的[步骤 4：提供反馈](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListRecommendations](#)中的。

## list-repository-associations

以下代码示例演示了如何使用 list-repository-associations。

### AWS CLI

列出您 AWS 账户中的仓库关联

以下 list-repository-associations 示例返回您账户中存储库关联摘要对象的列表。您可以按 ProviderType、NameState、和筛选返回的列表 Owner。

```
aws codeguru-reviewer list-repository-associations
```

输出：

```
{  
  "RepositoryAssociationSummaries": [  
    {  
      "LastUpdatedTimeStamp": 1595886609.616,  
      "Name": "test",  
      "AssociationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "Owner": "sample-owner",  
      "State": "Associated",  
      "AssociationArn": "arn:aws:codeguru-reviewer:us-west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "ProviderType": "Bitbucket"  
    },  
    {  
      "LastUpdatedTimeStamp": 1595636969.035,  
      "Name": "CodeDeploy-CodePipeline-ECS-Tutorial",  
      "AssociationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
      "Owner": "123456789012",  
      "State": "Associated",  
      "AssociationArn": "arn:aws:codeguru-reviewer:us-west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
    }  
  ]  
}
```

```
    "ProviderType": "CodeCommit"
  },
  {
    "LastUpdatedTimeStamp": 1595634785.983,
    "Name": "My-ecs-beta-repo",
    "AssociationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "Owner": "123456789012",
    "State": "Associated",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "ProviderType": "CodeCommit"
  },
  {
    "LastUpdatedTimeStamp": 1590712811.77,
    "Name": "MyTestCodeCommit",
    "AssociationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
    "Owner": "123456789012",
    "State": "Associated",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
    "ProviderType": "CodeCommit"
  },
  {
    "LastUpdatedTimeStamp": 1588102637.649,
    "Name": "aws-codeguru-profiler-sample-application",
    "AssociationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
    "Owner": "sample-owner",
    "State": "Associated",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
    "ProviderType": "GitHub"
  },
  {
    "LastUpdatedTimeStamp": 1588028233.995,
    "Name": "codeguru-profiler-demo-app",
    "AssociationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE66666",
    "Owner": "sample-owner",
    "State": "Associated",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE66666",
    "ProviderType": "GitHub"
  }
]
```

```
}
```

有关更多信息，请参阅 Amazon CodeGuru Reviewer 用户 [指南中的在 CodeGuru Reviewer 中查看所有存储库关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListRepositoryAssociations](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出关联存储库上的标签

以下 `list-tags-for-resource` 列出了关联存储库上的标签。这个关联的存储库有两个标签。

```
aws codeguru-reviewer list-tags-for-resource \
  --resource-arn arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
  "Tags": {
    "Status": "Secret",
    "Team": "Saanvi"
  }
}
```

有关更多信息，请参阅 Amazon CodeGuru CodeGuru Reviewer 用户 [指南中的查看审阅者关联存储库 \(AWS CLI\) 的标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## put-recommendation-feedback

以下代码示例演示了如何使用 `put-recommendation-feedback`。

### AWS CLI

在代码审查中添加建议

以下 `put-recommendation-feedback` 是关于代码审查 `ThumbsUp` 的建议。

```
aws codeguru-reviewer put-recommendation-feedback \  
  --code-review-arn \arn:aws:codeguru-reviewer:us-  
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111:code-  
review:RepositoryAnalysis-my-repository-name-branch-abcdefgh12345678 \  
  --recommendation-id  
3be1b2e5d7ef6e298a06499379ee290c9c596cf688fdcadb08285ddb0dd390eb \  
  --reactions ThumbsUp
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon CodeGuru Reviewer 用户指南中的 [步骤 4：提供反馈](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutRecommendationFeedback](#) 中的。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

### AWS CLI

向关联的存储库添加标签

以下内容 `tag-resource` 向关联的存储库添加了两个标签

```
aws codeguru-reviewer tag-resource \  
  --resource-arn arn:aws:codeguru-reviewer:us-  
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --tags Status=Secret,Team=Saarvi
```

此命令不生成任何输出。

有关更多信息，请参阅 [Amazon CodeGuru Reviewer 用户指南中的向 CodeGuru 审阅者关联存储库 \(AWS CLI\) 添加标签和为 CodeGuru 审阅者关联存储库 \(AWS CLI\) 添加或更新标签 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 `untag-resource`。

## AWS CLI

### 取消关联存储库的标签

以下内容`untag-resource`从关联的存储库中删除了两个密钥为“Secret”和“Team”的标签。

```
aws codeguru-reviewer untag-resource \  
  --resource-arn arn:aws:codeguru-reviewer:us-  
west-2:123456789012:association:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --tag-keys Status Team
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon CodeGuru Reviewer 用户指南中的[从 CodeGuru 审阅者关联存储库 \(AWS CLI\) 中移除标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UntagResource](#)中的。

## CodePipeline 使用示例 AWS CLI

以下代码示例向您展示了如何使用`with`来执行操作和实现常见场景 CodePipeline。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **acknowledge-job**

以下代码示例演示了如何使用 `acknowledge-job`。

## AWS CLI

### 检索有关指定作业的信息

此示例返回有关指定任务的信息，包括该作业的状态（如果存在）。这仅用于作业工作人员和自定义操作。要确定随机数值和任务 ID，请使用 `aws codepipeline poll-for-jobs ipeline`。

命令：

```
aws codepipeline acknowledge-job --job-id f4f4ff82-2d11-EXAMPLE --nonce 3
```

输出：

```
{
  "status": "InProgress"
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AcknowledgeJob](#) 中的。

## create-custom-action-type

以下代码示例演示了如何使用 `create-custom-action-type`。

## AWS CLI

### 创建自定义操作

此示例创建了一个自定义操作，用于 AWS CodePipeline 使用已创建的 JSON 文件（此处命名为 `MyCustomAction.json`），该文件包含自定义操作的结构。有关创建自定义操作的要求（包括文件结构）的更多信息，请参阅《AWS CodePipeline 用户指南》。

```
aws codepipeline create-custom-action-type --cli-input-json file://
MyCustomAction.json
```

JSON 文件的内容 `MyCustomAction.json`：

```
{
  "category": "Build",
  "provider": "MyJenkinsProviderName",
  "version": "1",
  "settings": {
```

```
    "entityUrlTemplate": "https://192.0.2.4/job/{Config:ProjectName}/",
    "executionUrlTemplate": "https://192.0.2.4/job/{Config:ProjectName}/
lastSuccessfulBuild/{ExternalExecutionId}/"
  },
  "configurationProperties": [
    {
      "name": "MyJenkinsExampleBuildProject",
      "required": true,
      "key": true,
      "secret": false,
      "queryable": false,
      "description": "The name of the build project must be provided when this
action is added to the pipeline.",
      "type": "String"
    }
  ],
  "inputArtifactDetails": {
    "maximumCount": 1,
    "minimumCount": 0
  },
  "outputArtifactDetails": {
    "maximumCount": 1,
    "minimumCount": 0
  }
}
```

此命令返回自定义操作的结构。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateCustomActionType](#) 中的。

## create-pipeline

以下代码示例演示了如何使用 create-pipeline。

### AWS CLI

#### 创建管道

此示例 AWS CodePipeline 使用已创建的 JSON 文件（此处名为 MySecondPipeline.json）创建了一个管道，该文件包含管道的结构。有关创建管道的要求（包括文件结构）的更多信息，请参阅《AWS CodePipeline 用户指南》。

命令：

```
aws codepipeline create-pipeline --cli-input-json file://MySecondPipeline.json
```

JSON 文件示例内容：

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::111111111111:role/AWS-CodePipeline-Service",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
            "name": "Source",
            "actionTypeId": {
              "category": "Source",
              "owner": "AWS",
              "version": "1",
              "provider": "S3"
            },
            "outputArtifacts": [
              {
                "name": "MyApp"
              }
            ],
            "configuration": {
              "S3Bucket": "awscodepipeline-demo-bucket",
              "S3ObjectKey": "aws-codepipeline-s3-aws-codedeploy_linux.zip"
            },
            "runOrder": 1
          }
        ]
      },
      {
        "name": "Beta",
        "actions": [
          {
            "inputArtifacts": [
              {
                "name": "MyApp"
              }
            ],
            "name": "CodePipelineDemoFleet",
```



```
    "actionTypeId": {
      "category": "Deploy",
      "owner": "AWS",
      "version": "1",
      "provider": "CodeDeploy"
    },
    "outputArtifacts": [],
    "configuration": {
      "ApplicationName": "CodePipelineDemoApplication",
      "DeploymentGroupName": "CodePipelineDemoFleet"
    },
    "runOrder": 1
  }
]
}
],
"artifactStore": {
  "type": "S3",
  "location": "codepipeline-us-east-1-11EXAMPLE11"
},
"name": "MySecondPipeline",
"version": 1
}
}
```

输出：

This command returns the structure of the pipeline.

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePipeline](#)中的。

## delete-custom-action-type

以下代码示例演示了如何使用 delete-custom-action-type。

### AWS CLI

#### 删除自定义操作

此示例使用已创建 AWS CodePipeline 的 JSON 文件（此处名为 DeleteMyCustomAction.json）删除中的自定义操作，该文件包含要删除的操作的操作类型、提供者名称和版本号。使用 list-action-types 命令查看类别、版本和提供程序的正确值。

**命令:**

```
aws codepipeline delete-custom-action-type --cli-input-json file://
DeleteMyCustomAction.json
```

**JSON 文件示例内容 :**

```
{
  "category": "Build",
  "version": "1",
  "provider": "MyJenkinsProviderName"
}
```

**输出 :**

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCustomActionType](#)中的。

**delete-pipeline**

以下代码示例演示了如何使用 delete-pipeline。

**AWS CLI****删除管道**

此示例删除了名为 MySecondPipeline 的管道 AWS CodePipeline。使用 list-pipelines 命令查看与您的 AWS 账户关联的管道列表。

**命令:**

```
aws codepipeline delete-pipeline --name MySecondPipeline
```

**输出 :**

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePipeline](#)中的。

## delete-webhook

以下代码示例演示了如何使用 delete-webhook。

### AWS CLI

#### 删除 Webhook

以下delete-webhook示例删除了 GitHub 版本 1 源操作的 webhook。在删除 webhook 之前，必须使用deregister-webhook-with-third-party命令将其注销。

```
aws codepipeline delete-webhook \  
  --name my-webhook
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodePipeline 用户指南》中的[删除 GitHub 来源的 webhook](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteWebhook](#)中的。

## deregister-webhook-with-third-party

以下代码示例演示了如何使用 deregister-webhook-with-third-party。

### AWS CLI

#### 取消注册 webhook

以下deregister-webhook-with-third-party示例删除了 GitHub 版本 1 源操作的 webhook。必须先取消注册 Webhook，然后才能删除它。

```
aws codepipeline deregister-webhook-with-third-party \  
  --webhook-name my-webhook
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodePipeline 用户指南》中的[删除 GitHub 来源的 webhook](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterWebhookWithThirdParty](#)中的。

## disable-stage-transition

以下代码示例演示了如何使用 disable-stage-transition。

## AWS CLI

### 禁用向管道中某个阶段的过渡

此示例禁止过渡到中 MyFirstPipeline 管道的 Beta 阶段。AWS CodePipeline

命令:

```
aws codepipeline disable-stage-transition --pipeline-name MyFirstPipeline --stage-name Beta --transition-type Inbound
```

输出:

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableStageTransition](#)中的。

## enable-stage-transition

以下代码示例演示了如何使用 enable-stage-transition。

## AWS CLI

### 启用向管道中某个阶段的过渡

此示例允许过渡到中 MyFirstPipeline 管道的 Beta 阶段 AWS CodePipeline。

命令:

```
aws codepipeline enable-stage-transition --pipeline-name MyFirstPipeline --stage-name Beta --transition-type Inbound
```

输出:

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableStageTransition](#)中的。

## get-job-details

以下代码示例演示了如何使用 get-job-details。

## AWS CLI

### 获取工作详情

此示例返回有关 ID 由 f4f4ff82-2d11-Example 表示的作业的详细信息。此命令仅用于自定义操作。调用此命令时，如果自定义操作需要，则 AWS CodePipeline 返回用于存储管道工件的 Amazon S3 存储桶的临时证书。此命令还将返回为操作定义的所有秘密值（如果有）。

命令:

```
aws codepipeline get-job-details --job-id f4f4ff82-2d11-EXAMPLE
```

输出:

```
{
  "jobDetails": {
    "accountId": "111111111111",
    "data": {
      "actionConfiguration": {
        "__type": "ActionConfiguration",
        "configuration": {
          "ProjectName": "MyJenkinsExampleTestProject"
        }
      },
      "actionTypeId": {
        "__type": "ActionTypeId",
        "category": "Test",
        "owner": "Custom",
        "provider": "MyJenkinsProviderName",
        "version": "1"
      },
      "artifactCredentials": {
        "__type": "AWSSessionCredentials",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "secretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "sessionToken":
          "fICcQD6m7oRw0uX0jANBqkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTA1dBMRwDgYDVQQHEwdw
          +a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/
          f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/
          MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpEIbb30hjZncvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQ
          +auNkyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs
      },
      "inputArtifacts": [
```

```
{
  "__type": "Artifact",
  "location": {
    "s3Location": {
      "bucketName": "codepipeline-us-east-1-11EXAMPLE11",
      "objectKey": "MySecondPipeline/MyAppBuild/EXAMPLE"
    },
    "type": "S3"
  },
  "name": "MyAppBuild"
}
],
"outputArtifacts": [],
"pipelineContext": {
  "__type": "PipelineContext",
  "action": {
    "name": "MyJenkinsTest-Action"
  },
  "pipelineName": "MySecondPipeline",
  "stage": {
    "name": "Testing"
  }
}
},
"id": "f4f4ff82-2d11-EXAMPLE"
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetJobDetails](#)中的。

## get-pipeline-state

以下代码示例演示了如何使用 get-pipeline-state。

### AWS CLI

获取有关管道状态的信息

此示例返回名为的管道的最新状态 MyFirstPipeline。

命令:

```
aws codepipeline get-pipeline-state --name MyFirstPipeline
```

输出：

```
{
  "created": 1446137312.204,
  "pipelineName": "MyFirstPipeline",
  "pipelineVersion": 1,
  "stageStates": [
    {
      "actionStates": [
        {
          "actionName": "Source",
          "entityUrl": "https://console.aws.amazon.com/s3/home?#",
          "latestExecution": {
            "lastStatusChange": 1446137358.328,
            "status": "Succeeded"
          }
        }
      ],
      "stageName": "Source"
    },
    {
      "actionStates": [
        {
          "actionName": "CodePipelineDemoFleet",
          "entityUrl": "https://console.aws.amazon.com/codedeploy/home?#/applications/CodePipelineDemoApplication/deployment-groups/CodePipelineDemoFleet",
          "latestExecution": {
            "externalExecutionId": "d-EXAMPLE",
            "externalExecutionUrl": "https://console.aws.amazon.com/codedeploy/home?#/deployments/d-EXAMPLE",
            "lastStatusChange": 1446137493.131,
            "status": "Succeeded",
            "summary": "Deployment Succeeded"
          }
        }
      ],
      "inboundTransitionState": {
        "enabled": true
      },
      "stageName": "Beta"
    }
  ],
  "updated": 1446137312.204
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPipelineState](#)中的。

## get-pipeline

以下代码示例演示了如何使用 get-pipeline。

### AWS CLI

#### 查看管道的结构

此示例返回名为的管道的结构 MyFirstPipeline。

命令:

```
aws codepipeline get-pipeline --name MyFirstPipeline
```

输出:

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::111111111111:role/AWS-CodePipeline-Service",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
            "name": "Source",
            "actionTypeId": {
              "category": "Source",
              "owner": "AWS",
              "version": "1",
              "provider": "S3"
            },
            "outputArtifacts": [
              {
                "name": "MyApp"
              }
            ],
            "configuration": {
```



```

        "S3Bucket": "awscodepipeline-demo-bucket",
        "S3ObjectKey": "aws-codepipeline-s3-aws-
codedeploy_linux.zip"
    },
    "runOrder": 1
  }
]
},
{
  "name": "Beta",
  "actions": [
    {
      "inputArtifacts": [
        {
          "name": "MyApp"
        }
      ],
      "name": "CodePipelineDemoFleet",
      "actionTypeId": {
        "category": "Deploy",
        "owner": "AWS",
        "version": "1",
        "provider": "CodeDeploy"
      },
      "outputArtifacts": [],
      "configuration": {
        "ApplicationName": "CodePipelineDemoApplication",
        "DeploymentGroupName": "CodePipelineDemoFleet"
      },
      "runOrder": 1
    }
  ]
}
],
"artifactStore": {
  "type": "S3",
  "location": "codepipeline-us-east-1-11EXAMPLE11"
},
"name": "MyFirstPipeline",
"version": 1
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPipeline](#)中的。

## list-action-executions

以下代码示例演示了如何使用 `list-action-executions`。

### AWS CLI

#### 列出动作执行情况

以下 `list-action-executions` 示例查看管道的操作执行详细信息，例如操作执行 ID、输入构件、输出构件、执行结果和状态。

```
aws codepipeline list-action-executions \  
  --pipeline-name myPipeline
```

#### 输出：

```
{  
  "actionExecutionDetails": [  
    {  
      "pipelineExecutionId": "EXAMPLE0-adfc-488e-bf4c-1111111720d3",  
      "actionExecutionId": "EXAMPLE4-2ee8-4853-bd6a-111111158148",  
      "pipelineVersion": 12,  
      "stageName": "Deploy",  
      "actionName": "Deploy",  
      "startTime": 1598572628.6,  
      "lastUpdateTime": 1598572661.255,  
      "status": "Succeeded",  
      "input": {  
        "actionTypeId": {  
          "category": "Deploy",  
          "owner": "AWS",  
          "provider": "CodeDeploy",  
          "version": "1"  
        },  
        "configuration": {  
          "ApplicationName": "my-application",  
          "DeploymentGroupName": "my-deployment-group"  
        },  
        "resolvedConfiguration": {  
          "ApplicationName": "my-application",  
          "DeploymentGroupName": "my-deployment-group"  
        },  
        "region": "us-east-1",  
      }  
    }  
  ]  
}
```

```
    "inputArtifacts": [
      {
        "name": "SourceArtifact",
        "s3location": {
          "bucket": "artifact-bucket",
          "key": "myPipeline/SourceArti/key"
        }
      }
    ],
    "namespace": "DeployVariables"
  },
  "output": {
    "outputArtifacts": [],
    "executionResult": {
      "externalExecutionId": "d-EXAMPLEE5",
      "externalExecutionSummary": "Deployment Succeeded",
      "externalExecutionUrl": "https://myaddress.com"
    },
    "outputVariables": {}
  }
},
{
  "pipelineExecutionId": "EXAMPLE0-adfc-488e-bf4c-1111111720d3",
  "actionExecutionId": "EXAMPLE5-abb4-4192-9031-11111113a7b0",
  "pipelineVersion": 12,
  "stageName": "Source",
  "actionName": "Source",
  "startTime": 1598572624.387,
  "lastUpdateTime": 1598572628.16,
  "status": "Succeeded",
  "input": {
    "actionTypeId": {
      "category": "Source",
      "owner": "AWS",
      "provider": "CodeCommit",
      "version": "1"
    },
    "configuration": {
      "BranchName": "production",
      "PollForSourceChanges": "false",
      "RepositoryName": "my-repo"
    },
    "resolvedConfiguration": {
      "BranchName": "production",
```

```

        "PollForSourceChanges": "false",
        "RepositoryName": "my-repo"
    },
    "region": "us-east-1",
    "inputArtifacts": [],
    "namespace": "SourceVariables"
},
"output": {
    "outputArtifacts": [
        {
            "name": "SourceArtifact",
            "s3location": {
                "bucket": "my-bucket",
                "key": "myPipeline/SourceArti/key"
            }
        }
    ],
    "executionResult": {
        "externalExecutionId":
"1111111ad99dcd35914c00b7fbea13995EXAMPLE",
        "externalExecutionSummary": "Edited template.yml",
        "externalExecutionUrl": "https://myaddress.com"
    },
    "outputVariables": {
        "AuthorDate": "2020-05-08T17:45:43Z",
        "BranchName": "production",
        "CommitId": "EXAMPLEad99dcd35914c00b7fbea139951111111",
        "CommitMessage": "Edited template.yml",
        "CommitterDate": "2020-05-08T17:45:43Z",
        "RepositoryName": "my-repo"
    }
}
},
. . . .

```

有关更多信息，请参阅《AWS CodePipeline 用户指南》中的[查看操作执行 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListActionExecutions](#)中的。

## list-action-types

以下代码示例演示了如何使用 list-action-types。

## AWS CLI

### 查看可用的操作类型

该 `list-action-types` 命令单独使用，返回您的 AWS 账户可用的所有操作的结构。此示例使用 `--action-owner-filter` 选项仅返回自定义操作。

命令:

```
aws codepipeline list-action-types --action-owner-filter Custom
```

输出:

```
{
  "actionTypes": [
    {
      "inputArtifactDetails": {
        "maximumCount": 5,
        "minimumCount": 0
      },
      "actionConfigurationProperties": [
        {
          "secret": false,
          "required": true,
          "name": "MyJenkinsExampleBuildProject",
          "key": true,
          "queryable": true
        }
      ],
      "outputArtifactDetails": {
        "maximumCount": 5,
        "minimumCount": 0
      },
      "id": {
        "category": "Build",
        "owner": "Custom",
        "version": "1",
        "provider": "MyJenkinsProviderName"
      },
      "settings": {
        "entityUrlTemplate": "http://192.0.2.4/job/{Config:ProjectName}",
        "executionUrlTemplate": "http://192.0.2.4/job/{Config:ProjectName}/
{ExternalExecutionId}"
      }
    }
  ]
}
```

```
    }
  },
  {
    "inputArtifactDetails": {
      "maximumCount": 5,
      "minimumCount": 0
    },
    "actionConfigurationProperties": [
      {
        "secret": false,
        "required": true,
        "name": "MyJenkinsExampleTestProject",
        "key": true,
        "queryable": true
      }
    ],
    "outputArtifactDetails": {
      "maximumCount": 5,
      "minimumCount": 0
    },
    "id": {
      "category": "Test",
      "owner": "Custom",
      "version": "1",
      "provider": "MyJenkinsProviderName"
    },
    "settings": {
      "entityUrlTemplate": "http://192.0.2.4/job/{Config:ProjectName}",
      "executionUrlTemplate": "http://192.0.2.4/job/{Config:ProjectName}/
{ExternalExecutionId}"
    }
  }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListActionTypes](#)中的。

## list-pipeline-executions

以下代码示例演示了如何使用 list-pipeline-executions。

## AWS CLI

### 查看管道执行历史记录

以下`list-pipeline-executions`示例显示了您 AWS 账户中某个管道的管道执行历史记录。

```
aws codepipeline list-pipeline-executions \  
  --pipeline-name MyPipeline
```

输出：

```
{  
  "pipelineExecutionSummaries": [  
    {  
      "lastUpdateTime": 1496380678.648,  
      "pipelineExecutionId": "7cf7f7cb-3137-539g-j458-d7eu3EXAMPLE",  
      "startTime": 1496380258.243,  
      "status": "Succeeded"  
    },  
    {  
      "lastUpdateTime": 1496591045.634,  
      "pipelineExecutionId": "3137f7cb-8d494hj4-039j-d84l-d7eu3EXAMPLE",  
      "startTime": 1496590401.222,  
      "status": "Succeeded"  
    },  
    {  
      "lastUpdateTime": 1496946071.6456,  
      "pipelineExecutionId": "4992f7jf-7cf7-913k-k334-d7eu3EXAMPLE",  
      "startTime": 1496945471.5645,  
      "status": "Succeeded"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS CodePipeline 用户指南》中的[查看执行历史记录](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPipelineExecutions](#)中的。

## list-pipelines

以下代码示例演示了如何使用 `list-pipelines`。

## AWS CLI

### 查看管道列表

此示例列出了与用户 AWS 账户关联的所有 AWS CodePipeline 管道。

命令:

```
aws codepipeline list-pipelines
```

输出:

```
{
  "pipelines": [
    {
      "updated": 1439504274.641,
      "version": 1,
      "name": "MyFirstPipeline",
      "created": 1439504274.641
    },
    {
      "updated": 1436461837.992,
      "version": 2,
      "name": "MySecondPipeline",
      "created": 1436460801.381
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPipelines](#)中的。

### list-tags-for-resource

以下代码示例演示了如何使用 list-tags-for-resource。

## AWS CLI

### 列出标签

以下list-tags-for-resource示例检索附加到指定管道资源的所有标签的列表。

```
aws codepipeline list-tags-for-resource \
```



```
--resource-arn arn:aws:codepipeline:us-east-1:123456789012:MyPipeline
```

输出：

```
{
  "tags": {
    "Project": "ProjectA",
    "IscontainerBased": "true"
  }
}
```

有关更多信息，请参阅《AWS CodePipeline 用户指南》中的[查看管道标签 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## list-webhooks

以下代码示例演示了如何使用 list-webhooks。

### AWS CLI

列出 Webhook

以下list-webhooks示例检索附加到指定管道资源的所有标签的列表。

```
aws codepipeline list-webhooks \
  --endpoint-url "https://codepipeline.eu-central-1.amazonaws.com" \
  --region "eu-central-1"
```

输出：

```
{
  "webhooks": [
    {
      "url": "https://webhooks.domain.com/
trigger1111111111EXAMPLE1111111111111111111": {
      "authenticationConfiguration": {
        "SecretToken": "Secret"
      },
      "name": "my-webhook",
      "authentication": "GITHUB_HMAC",
      "targetPipeline": "my-Pipeline",
```

```

        "targetAction": "Source",
        "filters": [
            {
                "jsonPath": "$.ref",
                "matchEquals": "refs/heads/{Branch}"
            }
        ]
    },
    "arn": "arn:aws:codepipeline:eu-central-1:123456789012:webhook:my-
webhook"
    }
]
}

```

有关更多信息，请参阅AWS CodePipeline 用户指南中的[列出账户中的 webhook](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListWebhooks](#)中的。

## poll-for-jobs

以下代码示例演示了如何使用 poll-for-jobs。

### AWS CLI

#### 查看任何空缺职位

此示例返回有关任何任务的信息，供求职者采取行动。此示例使用预定义的 JSON 文件 (MyActionTypeInfo.json) 来提供有关作业工作人员处理作业的操作类型的信息。此命令仅用于自定义操作。调用此命令时，会 AWS CodePipeline 返回用于存储管道工件的 Amazon S3 存储桶的临时证书。此命令还将返回为操作定义的所有秘密值 (如果有)。

#### 命令:

```
aws codepipeline poll-for-jobs --cli-input-json file://MyActionTypeInfo.json
```

#### JSON 文件示例内容：

```

{
  "actionTypeId": {
    "category": "Test",
    "owner": "Custom",
    "provider": "MyJenkinsProviderName",

```

```

    "version": "1"
  },
  "maxBatchSize": 5,
  "queryParam": {
    "ProjectName": "MyJenkinsTestProject"
  }
}

```

输出：

```

{
  "jobs": [
    {
      "accountId": "111111111111",
      "data": {
        "actionConfiguration": {
          "__type": "ActionConfiguration",
          "configuration": {
            "ProjectName": "MyJenkinsExampleTestProject"
          }
        },
        "actionTypeId": {
          "__type": "ActionTypeId",
          "category": "Test",
          "owner": "Custom",
          "provider": "MyJenkinsProviderName",
          "version": "1"
        },
        "artifactCredentials": {
          "__type": "AWSSessionCredentials",
          "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
          "secretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY",
          "sessionToken":
            "fICCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMaKGA1UEBhMCVVMxCzAJBgNVBAGTA1dBMRAwDgYDVQQHEwd
            +a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEI03IyNoH/
            f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/
            MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpEIbb30hjZncvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQ
            +auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs
        },
        "inputArtifacts": [
          {
            "__type": "Artifact",
            "location": {

```

```
        "s3Location": {
            "bucketName": "codepipeline-us-east-1-11EXAMPLE11",
            "objectKey": "MySecondPipeline/MyAppBuild/EXAMPLE"
        },
        "type": "S3"
    },
    "name": "MyAppBuild"
}
],
"outputArtifacts": [],
"pipelineContext": {
    "__type": "PipelineContext",
    "action": {
        "name": "MyJenkinsTest-Action"
    },
    "pipelineName": "MySecondPipeline",
    "stage": {
        "name": "Testing"
    }
}
},
"id": "ef66c259-64f9-EXAMPLE",
"nonce": "3"
}
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PollForJobs](#)中的。

## put-webhook

以下代码示例演示了如何使用 put-webhook。

### AWS CLI

#### 创建 webhook

以下put-webhook示例为 GitHub 版本 1 的源操作创建了一个 webhook。创建 webhook 后，必须使用 register-webhook-with-third-party 命令对其进行注册。

```
aws codepipeline put-webhook \  
  --cli-input-json file://webhook_json.json \  
  --cli-input-file file://webhook_json.json
```

```
--region "eu-central-1"
```

webhook\_json.json 的内容：

```
{
  "webhook": {
    "name": "my-webhook",
    "targetPipeline": "pipeline_name",
    "targetAction": "source_action_name",
    "filters": [
      {
        "jsonPath": "$.ref",
        "matchEquals": "refs/heads/{Branch}"
      }
    ],
    "authentication": "GITHUB_HMAC",
    "authenticationConfiguration": {
      "SecretToken": "secret"
    }
  }
}
```

输出：

```
{
  "webhook": {
    "url": "https://webhooks.domain.com/trigger1111111111EXAMPLE1111111111111111111",
    "definition": {
      "authenticationConfiguration": {
        "SecretToken": "secret"
      },
      "name": "my-webhook",
      "authentication": "GITHUB_HMAC",
      "targetPipeline": "pipeline_name",
      "targetAction": "Source",
      "filters": [
        {
          "jsonPath": "$.ref",
          "matchEquals": "refs/heads/{Branch}"
        }
      ]
    }
  },
}
```

```
    "arn": "arn:aws:codepipeline:eu-central-1:123456789012:webhook:my-webhook"
  },
  "tags": [
    {
      "key": "Project",
      "value": "ProjectA"
    }
  ]
}
```

有关更多信息，请参阅《AWS CodePipeline 用户指南》中的[为 GitHub 来源创建 webhook](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutWebhook](#)中的。

## retry-stage-execution

以下代码示例演示了如何使用 `retry-stage-execution`。

### AWS CLI

#### 重试失败的操作

以下`retry-stage-execution`示例重试操作失败的阶段。

```
aws codepipeline retry-stage-execution \
  --pipeline-name MyPipeline \
  --stage-name Deploy \
  --pipeline-execution-id b59babff-5f34-EXAMPLE \
  --retry-mode FAILED_ACTIONS
```

输出：

```
{
  "pipelineExecutionId": "b59babff-5f34-EXAMPLE"
}
```

有关更多信息，请参阅《AWS CodePipeline 用户指南》中的[“重试失败的操作 \(CLI\)”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RetryStageExecution](#)中的。

## start-pipeline-execution

以下代码示例演示了如何使用 `start-pipeline-execution`。

## AWS CLI

### 通过管道运行最新修订版

此示例通过名为“MyFirstPipeline”的管道运行管道源阶段中存在的最新版本。

命令:

```
aws codepipeline start-pipeline-execution --name MyFirstPipeline
```

输出 :

```
{
  "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartPipelineExecution](#)中的。

## stop-pipeline-execution

以下代码示例演示了如何使用 stop-pipeline-execution。

### AWS CLI

#### 停止管道执行

以下stop-pipeline-execution示例默认为等到正在进行的操作完成，然后停止管道。如果执行已处于 Stopping (正在停止) 状态，则无法选择停止并等待。您可以选择停止并放弃已处于 Stopping (正在停止) 状态的执行。

```
aws codepipeline stop-pipeline-execution \  
  --pipeline-name MyFirstPipeline \  
  --pipeline-execution-id d-EXAMPLE \  
  --reason "Stopping pipeline after the build action is done"
```

此命令不返回任何输出。

有关更多信息，请参阅《AWS CodePipeline 用户指南》中的[停止管道执行 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopPipelineExecution](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源添加标签

以下tag-resource示例将一组提供的标签与管道相关联。使用此命令添加或编辑标签。

```
aws codepipeline tag-resource \  
  --resource-arn arn:aws:codepipeline:us-east-1:123456789012:MyPipeline \  
  --tags key=Project,value=ProjectA key=IscontainerBased,value=true
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodePipeline 用户指南》中的[向管道添加标签 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从连接资源中移除 AWS 标签

以下untag-resource示例从指定资源中删除标签。

```
aws codepipeline untag-resource \  
  --resource-arn arn:aws:codepipeline:us-east-1:123456789012:MyPipeline \  
  --tag-keys Project IscontainerBased
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodePipeline 用户指南》中的[从管道中移除标签 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-pipeline

以下代码示例演示了如何使用 update-pipeline。



## AWS CLI

### 更新管道的结构

此示例使用带有--参数的 update-pipeline 命令。cli-input-json 此示例使用预定义的 JSON 文件 (MyFirstPipeline.json) 来更新管道的结构。AWS CodePipeline 识别 JSON 文件中包含的管道名称，然后应用管道结构中已修改字段的任何更改来更新管道。

创建预定义的 JSON 文件时，请遵循以下准则：

如果您正在使用使用 get-pipeline 命令检索的管道结构，则必须从 JSON 文件中的管道结构中删除元数据部分（“元数据”：{} 行以及其中的“已创建”、“PipelineArn”和“已更新”字段）。管道名称无法更改。

命令：

```
aws codepipeline update-pipeline --cli-input-json file://MyFirstPipeline.json
```

JSON 文件内容示例：

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::111111111111:role/AWS-CodePipeline-Service",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
            "name": "Source",
            "actionTypeId": {
              "category": "Source",
              "owner": "AWS",
              "version": "1",
              "provider": "S3"
            },
            "outputArtifacts": [
              {
                "name": "MyApp"
              }
            ],
            "configuration": {
```

```
        "S3Bucket": "awscodepipeline-demo-bucket2",
        "S3ObjectKey": "aws-codepipeline-s3-aws-codedeploy_linux.zip"
    },
    "runOrder": 1
}
]
},
{
    "name": "Beta",
    "actions": [
        {
            "inputArtifacts": [
                {
                    "name": "MyApp"
                }
            ],
            "name": "CodePipelineDemoFleet",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "version": "1",
                "provider": "CodeDeploy"
            },
            "outputArtifacts": [],
            "configuration": {
                "ApplicationName": "CodePipelineDemoApplication",
                "DeploymentGroupName": "CodePipelineDemoFleet"
            },
            "runOrder": 1
        }
    ]
}
],
"artifactStore": {
    "type": "S3",
    "location": "codepipeline-us-east-1-11EXAMPLE11"
},
"name": "MyFirstPipeline",
"version": 1
}
}
```

输出：

```
{
  "pipeline": {
    "artifactStore": {
      "location": "codepipeline-us-east-1-11EXAMPLE11",
      "type": "S3"
    },
    "name": "MyFirstPipeline",
    "roleArn": "arn:aws:iam::111111111111:role/AWS-CodePipeline-Service",
    "stages": [
      {
        "actions": [
          {
            "actionTypeId": {
              "__type": "ActionTypeId",
              "category": "Source",
              "owner": "AWS",
              "provider": "S3",
              "version": "1"
            },
            "configuration": {
              "S3Bucket": "awscodepipeline-demo-bucket2",
              "S3ObjectKey": "aws-codepipeline-s3-aws-codedeploy_linux.zip"
            },
            "inputArtifacts": [],
            "name": "Source",
            "outputArtifacts": [
              {
                "name": "MyApp"
              }
            ],
            "runOrder": 1
          }
        ],
        "name": "Source"
      },
      {
        "actions": [
          {
            "actionTypeId": {
              "__type": "ActionTypeId",
              "category": "Deploy",
              "owner": "AWS",
              "provider": "CodeDeploy",

```

```
        "version": "1"
      },
      "configuration": {
        "ApplicationName": "CodePipelineDemoApplication",
        "DeploymentGroupName": "CodePipelineDemoFleet"
      },
      "inputArtifacts": [
        {
          "name": "MyApp"
        }
      ],
      "name": "CodePipelineDemoFleet",
      "outputArtifacts": [],
      "runOrder": 1
    }
  ],
  "name": "Beta"
}
],
"version": 3
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePipeline](#)中的。

## AWS CodeStar 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS CodeStar。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### **associate-team-member**

以下代码示例演示了如何使用 `associate-team-member`。

#### AWS CLI

向项目添加团队成员

以下 `associate-team-member` 示例使用指定 ID 的 intern 用户成为项目的查看者。

```
aws codestar associate-team-member \  
  --project-id my-project \  
  --user-arn arn:aws:iam::123456789012:user/intern \  
  --project-role Viewer
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateTeamMember](#) 中的。

### **create-project**

以下代码示例演示了如何使用 `create-project`。

#### AWS CLI

创建项目

以下 `create-project` 示例使用 JSON 输入文件创建 CodeStar 项目。

```
aws codestar create-project \  
  --cli-input-json file://create-project.json
```

`create-project.json` 的内容：

```
{  
  "name": "Custom Project",  
  "id": "custom-project",  
  "sourceCode": [  
    {  
      "source": {  
        "s3": {
```

```
        "bucketName": "codestar-artifacts",
        "bucketKey": "nodejs-function.zip"
      }
    },
    "destination": {
      "codeCommit": {
        "name": "codestar-custom-project"
      }
    }
  ],
  "toolchain": {
    "source": {
      "s3": {
        "bucketName": "codestar-artifacts",
        "bucketKey": "toolchain.yml"
      }
    },
    "roleArn": "arn:aws:iam::123456789012:role/service-role/aws-codestar-
service-role",
    "stackParameters": {
      "ProjectId": "custom-project"
    }
  }
}
```

输出：

```
{
  "id": "my-project",
  "arn": "arn:aws:codestar:us-east-2:123456789012:project/custom-project"
}
```

有关包含自定义项目的示例代码和模板的教程，请参阅《< <https://docs.aws.amazon.com/codestar/latest/userguide/cli-tutorial.html> > AWS CodeStar 用户指南》中的“AWS CodeStar AWS 使用 CLI 创建项目”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateProject](#) 中的。

## create-user-profile

以下代码示例演示了如何使用 create-user-profile。

## AWS CLI

### 创建用户个人资料

以下`create-user-profile`示例为具有指定 ARN 的 IAM 用户创建用户个人资料。

```
aws codestar create-user-profile \  
  --user-arn arn:aws:iam::123456789012:user/intern \  
  --display-name Intern \  
  --email-address intern@example.com
```

输出：

```
{  
  "userArn": "arn:aws:iam::123456789012:user/intern",  
  "displayName": "Intern",  
  "emailAddress": "intern@example.com",  
  "sshPublicKey": "",  
  "createdTimestamp": 1572552308.607,  
  "lastModifiedTimestamp": 1572552308.607  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUserProfile](#)中的。

## delete-project

以下代码示例演示了如何使用 `delete-project`。

### AWS CLI

#### 删除项目

以下`delete-project`示例删除了指定的项目。

```
aws codestar delete-project \  
  --project-id my-project
```

输出：

```
{  
  "projectArn": "arn:aws:codestar:us-east-2:123456789012:project/my-project"
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteProject](#)中的。

## delete-user-profile

以下代码示例演示了如何使用 delete-user-profile。

### AWS CLI

删除用户个人资料

以下delete-user-profile示例删除具有指定 ARN 的用户的用户配置文件。

```
aws codestar delete-user-profile \  
  --user-arn arn:aws:iam::123456789012:user/intern
```

输出：

```
{  
  "userArn": "arn:aws:iam::123456789012:user/intern"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUserProfile](#)中的。

## describe-project

以下代码示例演示了如何使用 describe-project。

### AWS CLI

查看项目

以下describe-project示例检索有关指定项目的详细信息。

```
aws codestar describe-project \  
  --id my-project
```

输出：

```
{
```



```
"name": "my project",
"id": "my-project",
"arn": "arn:aws:codestar:us-west-2:123456789012:project/my-project",
"description": "My first CodeStar project.",
"createdTimeStamp": 1572547510.128,
"status": {
  "state": "CreateComplete"
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeProject](#)中的。

## describe-user-profile

以下代码示例演示了如何使用 describe-user-profile。

### AWS CLI

查看用户个人资料

以下describe-user-profile示例检索有关具有指定 ARN 的用户的用户个人资料的详细信息。

```
aws codestar describe-user-profile \
  --user-arn arn:aws:iam::123456789012:user/intern
```

输出：

```
{
  "userArn": "arn:aws:iam::123456789012:user/intern",
  "displayName": "Intern",
  "emailAddress": "intern@example.com",
  "sshPublicKey": "intern",
  "createdTimestamp": 1572552308.607,
  "lastModifiedTimestamp": 1572553495.47
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeUserProfile](#)中的。

## disassociate-team-member

以下代码示例演示了如何使用 disassociate-team-member。

## AWS CLI

### 移除团队成员

以下`disassociate-team-member`示例将具有指定 ARN 的用户从项目中移除。my-project

```
aws codestar disassociate-team-member \  
  --project-id my-project \  
  --user-arn arn:aws:iam::123456789012:user/intern
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateTeamMember](#)中的。

## list-projects

以下代码示例演示了如何使用 `list-projects`。

### AWS CLI

#### 查看项目

以下`list-projects`示例检索当前区域中的项目列表。

```
aws codestar list-projects
```

输出：

```
{  
  "projects": [  
    {  
      "projectId": "intern-projects",  
      "projectArn": "arn:aws:codestar:us-west-2:123456789012:project/intern-  
projects"  
    },  
    {  
      "projectId": "my-project",  
      "projectArn": "arn:aws:codestar:us-west-2:123456789012:project/my-  
project"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListProjects](#)中的。

## list-resources

以下代码示例演示了如何使用 `list-resources`。

### AWS CLI

#### 查看资源

以下`list-resources`示例检索指定项目的资源列表。

```
aws codestar list-resources \  
  --id my-project
```

输出：

```
{  
  "resources": [  
    {  
      "id": "arn:aws:execute-api:us-east-2:123456789012:r3wxmplbv8"  
    },  
    {  
      "id": "arn:aws:codedeploy:us-east-2:123456789012:application:awscodestar-my-project-lambda-ServerlessDeploymentApplication-PF0LXMPL1KA0"  
    },  
    {  
      "id": "arn:aws:s3::aws-codestar-us-east-2-123456789012-my-project-pipe"  
    },  
    {  
      "id": "arn:aws:lambda:us-east-2:123456789012:function:awscodestar-my-project-lambda-GetHelloWorld-16W3LVXMPLNNS"  
    },  
    {  
      "id": "arn:aws:cloudformation:us-east-2:123456789012:stack/awscodestar-my-project-lambda/b4904ea0-fc20-xmpl-bec6-029123b1cc42"  
    },  
    {  
      "id": "arn:aws:cloudformation:us-east-2:123456789012:stack/awscodestar-my-project/1b133f30-fc20-xmpl-a93a-0688c4290cb8"  
    },  
  ]  
}
```

```

    {
      "id": "arn:aws:iam::123456789012:role/CodeStarWorker-my-project-
ToolChain"
    },
    {
      "id": "arn:aws:iam::123456789012:policy/CodeStar_my-
project_PermissionsBoundary"
    },
    {
      "id": "arn:aws:s3:::aws-codestar-us-east-2-123456789012-my-project-app"
    },
    {
      "id": "arn:aws:codepipeline:us-east-2:123456789012:my-project-Pipeline"
    },
    {
      "id": "arn:aws:codedeploy:us-east-2:123456789012:deploymentgroup:my-
project/awscodestar-my-project-lambda-GetHelloWorldDeploymentGroup-P7YWXMLT0QB"
    },
    {
      "id": "arn:aws:iam::123456789012:role/CodeStar-my-project-Execution"
    },
    {
      "id": "arn:aws:iam::123456789012:role/CodeStarWorker-my-project-
CodeDeploy"
    },
    {
      "id": "arn:aws:codebuild:us-east-2:123456789012:project/my-project"
    },
    {
      "id": "arn:aws:iam::123456789012:role/CodeStarWorker-my-project-
CloudFormation"
    },
    {
      "id": "arn:aws:codecommit:us-east-2:123456789012:Go-project"
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResources](#)中的。

## list-tags-for-project

以下代码示例演示了如何使用 list-tags-for-project。

## AWS CLI

### 查看项目的标签

以下`list-tags-for-project`示例检索附加到指定项目的标签。

```
aws codestar list-tags-for-project \  
  --id my-project
```

输出：

```
{  
  "tags": {  
    "Department": "Marketing",  
    "Team": "Website"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForProject](#)中的。

## **list-team-members**

以下代码示例演示了如何使用 `list-team-members`。

## AWS CLI

### 查看团队成员列表

以下`list-team-members`示例检索与指定项目关联的用户列表。

```
aws codestar list-team-members \  
  --project-id my-project
```

输出：

```
{  
  "teamMembers": [  
    {  
      "userArn": "arn:aws:iam::123456789012:user/admin",  
      "projectRole": "Owner",
```

```
        "remoteAccessAllowed": false
      },
      {
        "userArn": "arn:aws:iam::123456789012:user/intern",
        "projectRole": "Contributor",
        "remoteAccessAllowed": false
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTeamMembers](#)中的。

## list-user-profiles

以下代码示例演示了如何使用 list-user-profiles。

### AWS CLI

查看用户个人资料列表

以下list-user-profiles示例检索当前区域中所有用户个人资料的列表。

```
aws codestar list-user-profiles
```

输出：

```
{
  "userProfiles": [
    {
      "userArn": "arn:aws:iam::123456789012:user/admin",
      "displayName": "me",
      "emailAddress": "me@example.com",
      "sshPublicKey": ""
    },
    {
      "userArn": "arn:aws:iam::123456789012:user/intern",
      "displayName": "Intern",
      "emailAddress": "intern@example.com",
      "sshPublicKey": "intern"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUserProfiles](#)中的。

## tag-project

以下代码示例演示了如何使用 tag-project。

### AWS CLI

将标签附加到项目

以下tag-project示例向指定项目添加名为Department且值Marketing为的标签。

```
aws codestar tag-project \  
  --id my-project \  
  --tags Department=Marketing
```

输出：

```
{  
  "tags": {  
    "Department": "Marketing"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagProject](#)中的。

## untag-project

以下代码示例演示了如何使用 untag-project。

### AWS CLI

从项目中移除标签

以下untag-project示例Team从指定项目中移除所有密钥名称为的标签。

```
aws codestar untag-project \  
  --id my-project \  
  --tags Team
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagProject](#)中的。

## update-project

以下代码示例演示了如何使用 update-project。

### AWS CLI

#### 更新项目

以下update-project示例为指定项目添加了描述。

```
aws codestar update-project \  
  --id my-project \  
  --description "My first CodeStar project"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateProject](#)中的。

## update-team-member

以下代码示例演示了如何使用 update-team-member。

### AWS CLI

#### 修改团队成员

以下update-team-member示例使指定用户成为项目的贡献者，并授予他们对项目资源的远程访问权限。

```
aws codestar update-team-member \  
  --project-id my-project \  
  --user-arn arn:aws:iam::123456789012:user/intern \  
  --project-role Contributor -\  
  --remote-access-allowed
```

输出：

```
{  
  "userArn": "arn:aws:iam::123456789012:user/intern",
```



```
"projectRole": "Contributor",
"remoteAccessAllowed": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateTeamMember](#)中的。

## update-user-profile

以下代码示例演示了如何使用 update-user-profile。

### AWS CLI

#### 修改用户配置文件

以下update-user-profile示例将指定的 SSH 密钥添加到指定用户。

```
aws codestar update-user-profile \
  --ssh-public-key intern \
  --user-arn arn:aws:iam::123456789012:user/intern
```

输出：

```
{
  "userArn": "arn:aws:iam::123456789012:user/intern",
  "displayName": "Intern",
  "emailAddress": "intern@example.com",
  "sshPublicKey": "intern",
  "createdTimestamp": 1572552308.607,
  "lastModifiedTimestamp": 1572553495.47
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUserProfile](#)中的。

## AWS CodeStar 使用通知示例 AWS CLI

以下代码示例向您展示了如何使用 with Notifications 来执行操作和实现常见场 AWS CodeStar 景。

### AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 [GitHub](#)，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## **create-notification-rule**

以下代码示例演示了如何使用 `create-notification-rule`。

AWS CLI

创建通知规则

以下 `create-notification-rule` 示例使用名为的 JSON 文件 `rule.json` 为指定 AWS 账户 `MyDemoRepo` 中名 `MyNotificationRule` 为的存储库创建名为的通知规则。创建分支和标签时，带有 FULL 详细信息的通知将发送到指定的目标 Amazon SNS 主题。

```
aws codestar-notifications create-notification-rule \  
  --cli-input-json file://rule.json
```

`rule.json` 的内容：

```
{  
  "Name": "MyNotificationRule",  
  "EventTypeId": [ "codecommit-repository-branches-and-tags-created" ],  
  "Resource": "arn:aws:codecommit:us-east-1:123456789012:MyDemoRepo",  
  "Targets": [ {  
    "TargetType": "SNS",  
    "TargetAddress": "arn:aws:sns:us-east-1:123456789012:MyNotificationTopic"  
  } ],  
  "Status": "ENABLED",  
  "DetailType": "FULL"
```

```
}
```

输出：

```
{
  "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

有关更多信息，请参阅《AWS 开发者工具控制台用户指南》中的[创建通知规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateNotificationRule](#)中的。

## delete-notification-rule

以下代码示例演示了如何使用 delete-notification-rule。

AWS CLI

删除通知规则

以下delete-notification-rule示例删除了指定的通知规则。

```
aws codestar-notifications delete-notification-rule \
  --arn arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE
```

输出：

```
{
  "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

有关更多信息，请参阅《AWS 开发者工具控制台用户指南》中的[删除通知规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNotificationRule](#)中的。

## delete-target

以下代码示例演示了如何使用 delete-target。

## AWS CLI

### 删除通知规则目标

以下delete-target示例将指定目标从所有配置为用作目标的通知规则中移除，然后删除该目标。

```
aws codestar-notifications delete-target \  
  --target-address arn:aws:sns:us-east-1:123456789012:MyNotificationTopic \  
  --force-unsubscribe-all
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 开发者工具控制台用户指南》中的[删除通知规则目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTarget](#)中的。

## describe-notification-rule

以下代码示例演示了如何使用 describe-notification-rule。

### AWS CLI

#### 检索通知规则的详细信息

以下describe-notification-rule示例检索指定通知规则的详细信息。

```
aws codestar-notifications describe-notification-rule \  
  --arn arn:aws:codestar-notifications:us-west-2:123456789012:notificationrule/  
  dc82df7a-EXAMPLE
```

输出：

```
{  
  "LastModifiedTimestamp": 1569199844.857,  
  "EventTypes": [  
    {  
      "ServiceName": "CodeCommit",  
      "EventTypeName": "Branches and tags: Created",  
      "ResourceType": "Repository",  
      "EventTypeId": "codecommit-repository-branches-and-tags-created"  
    }  
  ]  
}
```

```

    ],
    "Status": "ENABLED",
    "DetailType": "FULL",
    "Resource": "arn:aws:codecommit:us-west-2:123456789012:MyDemoRepo",
    "Arn": "arn:aws:codestar-notifications:us-west-w:123456789012:notificationrule/
dc82df7a-EXAMPLE",
    "Targets": [
      {
        "TargetStatus": "ACTIVE",
        "TargetAddress": "arn:aws:sns:us-
west-2:123456789012:MyNotificationTopic",
        "TargetType": "SNS"
      }
    ],
    "Name": "MyNotificationRule",
    "CreatedTimestamp": 1569199844.857,
    "CreatedBy": "arn:aws:iam::123456789012:user/Mary_Major"
  }
}

```

有关更多信息，请参阅《AWS 开发者工具控制台用户指南》中的“[查看通知规则](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeNotificationRule](#) 中的。

## list-event-types

以下代码示例演示了如何使用 list-event-types。

### AWS CLI

获取通知规则的事件类型列表

以下 list-event-types 示例检索 CodeDeploy 应用程序的所有可用通知事件类型的筛选列表。相反，如果您不使用任何过滤器，则该命令将返回所有资源类型的所有通知事件类型。

```

aws codestar-notifications list-event-types \
  --filters Name=SERVICE_NAME,Value=CodeDeploy

```

输出：

```

{
  "EventTypes": [
    {
      "EventTypeId": "codedeploy-application-deployment-succeeded",

```

```

        "ServiceName": "CodeDeploy",
        "EventTypeName": "Deployment: Succeeded",
        "ResourceType": "Application"
    },
    {
        "EventTypeId": "codedeploy-application-deployment-failed",
        "ServiceName": "CodeDeploy",
        "EventTypeName": "Deployment: Failed",
        "ResourceType": "Application"
    },
    {
        "EventTypeId": "codedeploy-application-deployment-started",
        "ServiceName": "CodeDeploy",
        "EventTypeName": "Deployment: Started",
        "ResourceType": "Application"
    }
]
}

```

有关更多信息，请参阅《AWS 开发者工具控制台用户指南》中的[创建通知规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListEventTypes](#) 中的。

## list-notification-rules

以下代码示例演示了如何使用 list-notification-rules。

### AWS CLI

#### 检索通知规则列表

以下 list-notification-rules 示例检索指定 AWS 区域中所有通知规则的列表。

```
aws codestar-notifications list-notification-rules --region us-east-1
```

输出：

```

{
  "NotificationRules": [
    {
      "Id": "dc82df7a-EXAMPLE",
      "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/dc82df7a-EXAMPLE"
    }
  ]
}

```

```
    },  
    {  
      "Id": "8d1f0983-EXAMPLE",  
      "Arn": "arn:aws:codestar-notifications:us-  
east-1:123456789012:notificationrule/8d1f0983-EXAMPLE"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS 开发者工具控制台用户指南》中的[“查看通知规则”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListNotificationRules](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

获取附加到通知规则的标签列表

以下`list-tags-for-resource`示例检索附加到指定通知规则的所有标签的列表。在此示例中，通知规则当前没有与之关联的标签。

```
aws codestar-notifications list-tags-for-resource \  
  --arn arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/  
fe1efd35-EXAMPLE
```

输出：

```
{  
  "Tags": {}  
}
```

有关更多信息，请参阅《AWS 开发者工具控制台用户指南》中的[创建通知规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## list-targets

以下代码示例演示了如何使用 `list-targets`。

## AWS CLI

### 检索通知规则目标列表

以下list-targets示例检索指定 AWS 区域中所有通知规则目标的列表。

```
aws codestar-notifications list-targets \  
  --region us-east-1
```

输出：

```
{  
  "Targets": [  
    {  
      "TargetAddress": "arn:aws:sns:us-  
east-1:123456789012:MySNSTopicForNotificationRules",  
      "TargetType": "SNS",  
      "TargetStatus": "ACTIVE"  
    },  
    {  
      "TargetAddress": "arn:aws:sns:us-  
east-1:123456789012:MySNSTopicForNotificationsAboutMyDemoRepo",  
      "TargetType": "SNS",  
      "TargetStatus": "ACTIVE"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS 开发者工具控制台用户指南》中的[查看通知规则目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTargets](#)中的。

## subscribe

以下代码示例演示了如何使用 subscribe。

## AWS CLI

### 向通知规则中添加目标

以下subscribe示例将 Amazon SNS 主题添加为指定通知规则的目标。

```
aws codestar-notifications subscribe \  
  --region us-east-1
```



```
--arn arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE \
--target TargetType=SNS,TargetAddress=arn:aws:sns:us-
east-1:123456789012:MyNotificationTopic
```

输出：

```
{
  "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

有关更多信息，请参阅AWS 开发者工具控制台用户指南中的[添加或删除作为通知规则目标的 Amazon SNS 主题](#)。

- 有关 API 详细信息，请参阅《AWS CLI Command Reference》中的 [Subscribe](#)。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

向通知规则添加标签

以下tag-resource示例将密钥名称为Team、值Li\_Juan为的标签添加到指定的通知规则。

```
aws codestar-notifications tag-resource \
--arn arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
fe1efd35-EXAMPLE \
--tags Team=Li_Juan
```

输出：

```
{
  "Tags": {
    "Team": "Li_Juan"
  }
}
```

有关更多信息，请参阅《AWS 开发者工具控制台用户指南》中的[创建通知规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## unsubscribe

以下代码示例演示了如何使用 unsubscribe。

### AWS CLI

从通知规则中移除目标

以下unsubscribe示例将作为目标的 Amazon SNS 主题从指定的通知规则中删除。

```
aws codestar-notifications unsubscribe \  
  --arn arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/  
dc82df7a-EXAMPLE \  
  --target TargetType=SNS,TargetAddress=arn:aws:sns:us-  
east-1:123456789012:MyNotificationTopic
```

输出：

```
{  
  "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/  
dc82df7a-EXAMPLE"  
  "TargetAddress": "arn:aws:sns:us-east-1:123456789012:MyNotificationTopic"  
}
```

有关更多信息，请参阅AWS 开发者工具控制台用户指南中的[添加或删除作为通知规则目标的 Amazon SNS 主题](#)。

- 有关 API 详细信息，请参阅《AWS CLI Command Reference》中的 [Unsubscribe](#)。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从通知规则中移除标签

以下untag-resource示例Team从指定的通知规则中删除带有密钥名称的标签。

```
aws codestar-notifications untag-resource \  
  --arn arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/  
fe1efd35-EXAMPLE \  
  --tag-keys Team
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 开发者工具控制台用户指南》中的[编辑通知规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-notification-rule

以下代码示例演示了如何使用 update-notification-rule。

### AWS CLI

#### 更新通知规则

以下update-notification-rule示例123456789012使用名为的 JSON 文件更新 AWS 账户MyNotificationRule中名为的通知规则update.json。

```
aws codestar-notifications update-notification-rule \  
  --cli-input-json file://update.json
```

update.json 的内容：

```
{  
  "Name": "MyUpdatedNotificationRule",  
  "EventTypeIds": [  
    "codecommit-repository-branches-and-tags-created"  
  ],  
  "Resource": "arn:aws:codecommit:us-east-1:123456789012:MyDemoRepo",  
  "Targets": [  
    {  
      "TargetType": "SNS",  
      "TargetAddress": "arn:aws:sns:us-  
east-1:123456789012:MyNotificationTopic"  
    }  
  ],  
  "Status": "ENABLED",  
  "DetailType": "FULL"
```

```
}
```

输出：

```
{
  "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

有关更多信息，请参阅《AWS 开发者工具控制台用户指南》中的[编辑通知规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateNotificationRule](#)中的。

## CodeConnections 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 CodeConnections。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-connection**

以下代码示例演示了如何使用 `create-connection`。

AWS CLI

创建连接

以下 `create-connection` 示例说明如何创建与第三方存储库的连接。此示例创建了一个连接，其中第三方提供商是 Bitbucket。

通过 AWS CLI 创建的连接或默认 AWS CloudFormation 处于待处理状态。使用 CLI 或创建连接后 AWS CloudFormation，使用控制台编辑连接以使其状态变为“可用”。

```
aws codestar-connections create-connection \  
  --provider-type Bitbucket \  
  --connection-name MyConnection
```

输出：

```
{  
  "ConnectionArn": "arn:aws:codestar-connections:us-  
east-1:123456789012:connection/aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"  
}
```

有关更多信息，请参阅《开发者工具控制台用户指南》中的[创建连接](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateConnection](#) 中的。

## create-host

以下代码示例演示了如何使用 create-host。

### AWS CLI

#### 创建主机

以下 create-host 示例说明如何创建主机来表示安装第三方提供商的基础架构的终端节点。此示例创建了一台主机，其中安装的第三方提供商是 GitHub 企业服务器。

默认情况下，通过 AWS CLI 创建的主机处于待处理状态。使用 CLI 创建主机后，使用控制台或 CLI 将主机设置为可用状态。

```
aws codestar-connections create-host \  
  --name MyHost \  
  --provider-type GitHubEnterpriseServer \  
  --provider-endpoint "https://my-instance.dev"
```

输出：

```
{  
  "HostArn": "arn:aws:codestar-connections:us-east-1:123456789012:host/My-  
Host-28aef605"
```

```
}
```

有关更多信息，请参阅《开发者工具控制台用户指南》中的[创建主机 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateHost](#)中的。

## delete-connection

以下代码示例演示了如何使用 delete-connection。

### AWS CLI

#### 删除连接

以下delete-connection示例说明如何删除连接。

```
aws codestar-connections delete-connection \  
    --connection-arn arn:aws:codestar-connections:us-west-2:123456789012:connection/  
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f
```

此命令不生成任何输出。

有关更多信息，请参阅《开发者工具控制台用户指南》中的[删除连接 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteConnection](#)中的。

## delete-host

以下代码示例演示了如何使用 delete-host。

### AWS CLI

#### 删除主机

以下delete-host示例说明如何删除主机。必须先删除与主机关联的所有连接，然后才能删除主机。

```
aws codestar-connections delete-host \  
    --host-arn "arn:aws:codestar-connections:us-east-1 :123456789012:host/My-  
Host-28aef605"
```

此命令不生成任何输出。

有关更多信息，请参阅《开发者工具控制台用户指南》中的[删除主机 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteHost](#)中的。

## get-connection

以下代码示例演示了如何使用 get-connection。

### AWS CLI

获取有关连接的信息

以下get-connection示例显示了有关连接的详细信息。

```
aws codestar-connections get-connection \  
  --connection-arn arn:aws:codestar-connections:us-east-1:123456789012:connection/  
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f
```

输出：

```
{  
  "Connection": {  
    "ConnectionName": "MyConnection",  
    "ConnectionArn": "arn:aws:codestar-connections:us-  
east-1:123456789012:connection/aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f",  
    "ProviderType": "Bitbucket",  
    "OwnerAccountId": "123456789012",  
    "ConnectionStatus": "AVAILABLE"  
  }  
}
```

有关更多信息，请参阅《开发者工具控制台用户指南》中的“[查看连接详情](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetConnection](#)中的。

## get-host

以下代码示例演示了如何使用 get-host。

### AWS CLI

获取有关房东的信息

以下`get-host`示例显示了有关主机的详细信息：

```
aws codestar-connections get-host \  
  --host-arn arn:aws:codestar-connections:us-east-1:123456789012:host/  
MyHost-28aef605
```

输出：

```
{  
  "Name": "MyHost",  
  "Status": "AVAILABLE",  
  "ProviderType": "GitHubEnterpriseServer",  
  "ProviderEndpoint": "https://test-instance-1.dev/"  
}
```

有关更多信息，请参阅《开发者工具控制台用户指南》中的[查看主机详细信息 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetHost](#)中的。

## list-connections

以下代码示例演示了如何使用 `list-connections`。

### AWS CLI

列出连接

以下`list-connections`示例检索您账户中与 Bitbucket 提供商类型对应的所有连接的列表。：

```
aws codestar-connections list-connections \  
  --provider-type Bitbucket \  
  --max-results 5 \  
  --next-token: next-token
```

输出：

```
{  
  "Connections": [  
    {  
      "ConnectionName": "my-connection",
```



```

        "ProviderType": "Bitbucket",
        "Status": "PENDING",
        "ARN": "arn:aws:codestar-connections:us-east-1:123456789012:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f",
        "OwnerAccountId": "123456789012"
    },
    {
        "ConnectionName": "my-other-connection",
        "ProviderType": "Bitbucket",
        "Status": "AVAILABLE",
        "ARN": "arn:aws:codestar-connections:us-east-1:123456789012:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f",
        "OwnerAccountId": "123456789012"
    },
],
"NextToken": "next-token"
}

```

有关更多信息，请参阅《开发者工具控制台用户指南》中的[列出连接 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListConnections](#)中的。

## list-hosts

以下代码示例演示了如何使用 list-hosts。

### AWS CLI

列出房东

以下list-hosts示例检索您账户中所有主机的列表。

```
aws codestar-connections list-hosts
```

输出：

```

{
  "Hosts": [
    {
      "Name": "My-Host",
      "HostArn": "arn:aws:codestar-connections:us-east-1:123456789012:host/My-
Host-28aef605",

```

```
        "ProviderType": "GitHubEnterpriseServer",
        "ProviderEndpoint": "https://my-instance.test.dev",
        "Status": "AVAILABLE"
    }
]
}
```

有关更多信息，请参阅《开发者工具控制台用户指南》中的[列出主机 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListHosts](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

#### 列出标签

以下`list-tags-for-resource`示例检索附加到指定连接资源的所有标签的列表。

```
aws codestar-connections list-tags-for-resource \
  --resource-arn arn:aws:codestar-connections:us-east-1:123456789012:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f
```

#### 输出：

```
{
  "Tags": [
    {
      "Key": "Project",
      "Value": "ProjectA"
    },
    {
      "Key": "ReadOnly",
      "Value": "true"
    }
  ]
}
```

有关更多信息，请参阅《开发者工具控制台用户指南》中的[查看连接资源的标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源添加标签

以下tag-resource示例将一组提供的标签与连接相关联。使用此命令添加或编辑标签。

```
aws codestar-connections tag-resource \  
  --resource-arn arn:aws:codestar-connections:us-east-1:123456789012:connection/  
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f \  
  --tags Key=Project,Value=ProjectA Key=IscontainerBased,Value=true
```

此命令不生成任何输出。

有关更多信息，请参阅《开发者工具控制台用户指南》中的[向连接资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从连接资源中移除 AWS 标签

以下内容从指定资源中untag-resource删除标签。

```
aws codestar-connections untag-resource \  
  --resource-arn arn:aws:codestar-connections:us-east-1:123456789012:connection/  
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f \  
  --tag-keys Project ReadOnly
```

输出：

```
{
```

```
"Tags": []  
}
```

有关更多信息，请参阅《开发者工具控制台用户指南》中的[从连接资源中移除标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## 使用 Amazon Cognito 身份示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon Cognito Identity 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-identity-pool**

以下代码示例演示了如何使用 create-identity-pool。

AWS CLI

使用 Cognito 身份池提供者创建身份池

此示例创建了一个名为的身份池 MyIdentityPool。它有一个 Cognito 身份池提供者。不允许使用未经身份验证的身份。

命令:

```
aws cognito-identity create-identity-pool --identity-pool-name  
MyIdentityPool --no-allow-unauthenticated-identities --cognito-
```

```
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-west-2_aaaaaaaa",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

输出：

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
    {
      "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-west-2_1111111111",
      "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
      "ServerSideTokenCheck": false
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateIdentityPool](#)中的。

## delete-identities

以下代码示例演示了如何使用 delete-identities。

### AWS CLI

删除身份池

此示例删除身份池。

命令：

```
aws cognito-identity delete-identity-pool --identity-ids-to-delete "us-west-2:11111111-1111-1111-1111-111111111111"
```

输出：

```
{
  "UnprocessedIdentityIds": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteIdentities](#)中的。

## delete-identity-pool

以下代码示例演示了如何使用 delete-identity-pool。

### AWS CLI

删除身份池

以下 delete-identity-pool 示例删除指定身份池。

命令:

```
aws cognito-identity delete-identity-pool \  
  --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteIdentityPool](#)中的。

## describe-identity-pool

以下代码示例演示了如何使用 describe-identity-pool。

### AWS CLI

描述身份池

此示例描述了身份池。

命令:

```
aws cognito-identity describe-identity-pool --identity-pool-id "us-  
west-2:11111111-1111-1111-1111-111111111111"
```

输出 :

```
{  
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",  
  "IdentityPoolName": "MyIdentityPool",  
  "AllowUnauthenticatedIdentities": false,  
  "CognitoIdentityProviders": [  
    {
```

```
        "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-west-2_1111111111",
        "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
        "ServerSideTokenCheck": false
    }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeIdentityPool](#)中的。

## get-identity-pool-roles

以下代码示例演示了如何使用 get-identity-pool-roles。

### AWS CLI

获取身份池角色

此示例获取身份池角色。

命令:

```
aws cognito-identity get-identity-pool-roles --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111"
```

输出:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "Roles": {
    "authenticated": "arn:aws:iam::111111111111:role/Cognito_MyIdentityPoolAuth_Role",
    "unauthenticated": "arn:aws:iam::111111111111:role/Cognito_MyIdentityPoolUnauth_Role"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetIdentityPoolRoles](#)中的。

## list-identity-pools

以下代码示例演示了如何使用 list-identity-pools。

## AWS CLI

### 列出身份池

此示例列出身份池。最多可列出 20 个身份。

命令:

```
aws cognito-identity list-identity-pools --max-results 20
```

输出:

```
{
  "IdentityPools": [
    {
      "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
      "IdentityPoolName": "MyIdentityPool"
    },
    {
      "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
      "IdentityPoolName": "AnotherIdentityPool"
    },
    {
      "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
      "IdentityPoolName": "IdentityPoolRegionA"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListIdentityPools](#)中的。

## set-identity-pool-roles

以下代码示例演示了如何使用 set-identity-pool-roles。

## AWS CLI

### 设置身份池角色

以下set-identity-pool-roles示例设置身份池角色。

```
aws cognito-identity set-identity-pool-roles \
```



```
--identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111" \  
--roles authenticated="arn:aws:iam::111111111111:role/  
Cognito_MyIdentityPoolAuth_Role"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetIdentityPoolRoles](#)中的。

## update-identity-pool

以下代码示例演示了如何使用 update-identity-pool。

### AWS CLI

#### 更新身份池

此示例更新身份池。它将名称设置为 MyIdentityPool。它将 Cognito 添加为身份提供者。它不允许使用未经身份验证的身份。

命令:

```
aws cognito-identity update-identity-pool --identity-pool-id "us-  
west-2:11111111-1111-1111-1111-111111111111" --identity-pool-name  
"MyIdentityPool" --no-allow-unauthenticated-identities --cognito-  
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-  
west-2_11111111",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

输出:

```
{  
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",  
  "IdentityPoolName": "MyIdentityPool",  
  "AllowUnauthenticatedIdentities": false,  
  "CognitoIdentityProviders": [  
    {  
      "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-west-2_11111111",  
      "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",  
      "ServerSideTokenCheck": false  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateIdentityPool](#)中的。

## 使用 Amazon Cognito 身份提供商示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon Cognito 身份提供商配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **add-custom-attributes**

以下代码示例演示了如何使用 add-custom-attributes。

#### AWS CLI

##### 添加自定义属性

此示例将自定义属性 CustomAttr 1 添加到用户池。它是 String 类型，最少需要 1 个字符，最多 15 个字符。但其并非必要项目。

命令:

```
aws cognito-idp add-custom-attributes --user-pool-id us-west-2_aaaaaaaaa --custom-attributes
Name="CustomAttr1",AttributeDataType="String",DeveloperOnlyAttribute=false,Required=false,S
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddCustomAttributes](#)中的。

#### **admim-disable-user**

以下代码示例演示了如何使用 admim-disable-user。

## AWS CLI

### 禁用用户

此示例禁用用户 jane@example.com。

命令:

```
aws cognito-idp admin-disable-user --user-pool-id us-west-2_aaaaaaaaa --username jane@example.com
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminDisableUser](#)中的。

### admim-enable-user

以下代码示例演示了如何使用 admim-enable-user。

## AWS CLI

### 启用用户

此示例启用用户名 jane@example.com。

命令:

```
aws cognito-idp admin-enable-user --user-pool-id us-west-2_aaaaaaaaa --username jane@example.com
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminEnableUser](#)中的。

### admin-add-user-to-group

以下代码示例演示了如何使用 admin-add-user-to-group。

## AWS CLI

### 将用户添加到群组

此示例将用户 Jane 添加到群组 MyGroup。

命令:

```
aws cognito-idp admin-add-user-to-group --user-pool-id us-west-2_aaaaaaaaa --username Jane --group-name MyGroup
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminAddUserToGroup](#)中的。

## admin-confirm-sign-up

以下代码示例演示了如何使用 admin-confirm-sign-up。

### AWS CLI

确认用户注册

这个例子证实了用户 jane@example.com。

命令:

```
aws cognito-idp admin-confirm-sign-up --user-pool-id us-west-2_aaaaaaaaa --username jane@example.com
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminConfirmSignUp](#)中的。

## admin-create-user

以下代码示例演示了如何使用 admin-create-user。

### AWS CLI

创建用户

以下admin-create-user示例使用指定设置的电子邮件地址和电话号码创建用户。

```
aws cognito-idp admin-create-user \  
  --user-pool-id us-west-2_aaaaaaaaa \  
  --username diego \  
  --user-attributes Name=email,Value=diego@example.com \  
  Name=phone_number,Value="+15555551212" \  
  --message-action SUPPRESS
```

输出：

```
{
  "User": {
    "Username": "diego",
    "Attributes": [
      {
        "Name": "sub",
        "Value": "7325c1de-b05b-4f84-b321-9adc6e61f4a2"
      },
      {
        "Name": "phone_number",
        "Value": "+15555551212"
      },
      {
        "Name": "email",
        "Value": "diego@example.com"
      }
    ],
    "UserCreateDate": 1548099495.428,
    "UserLastModifiedDate": 1548099495.428,
    "Enabled": true,
    "UserStatus": "FORCE_CHANGE_PASSWORD"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminCreateUser](#)中的。

## admin-delete-user-attributes

以下代码示例演示了如何使用 admin-delete-user-attributes。

### AWS CLI

#### 删除用户属性

此示例删除了用户 diego@example.com 的自定义属性 CustomAttr 1。

#### 命令:

```
aws cognito-idp admin-delete-user-attributes --user-pool-id us-west-2_aaaaaaaaa --
username diego@example.com --user-attribute-names "custom:CustomAttr1"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminDeleteUserAttributes](#)中的。

## admin-delete-user

以下代码示例演示了如何使用 `admin-delete-user`。

### AWS CLI

删除用户

此示例删除一个用户。

命令:

```
aws cognito-idp admin-delete-user --user-pool-id us-west-2_aaaaaaaaa --username
diego@example.com
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminDeleteUser](#)中的。

## admin-forget-device

以下代码示例演示了如何使用 `admin-forget-device`。

### AWS CLI

忘记设备

这个例子忘记了设备作为用户名 `jane@example.com`

命令:

```
aws cognito-idp admin-forget-device --user-pool-id us-west-2_aaaaaaaaa --username
jane@example.com --device-key us-west-2_abcd_1234-5678
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminForgetDevice](#)中的。

## admin-get-device

以下代码示例演示了如何使用 `admin-get-device`。

### AWS CLI

获取设备

这个例子为用户名获取了一台设备 `jane@example.com`

命令:

```
aws cognito-idp admin-get-device --user-pool-id us-west-2_aaaaaaaaa --username
jane@example.com --device-key us-west-2_abcd_1234-5678
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminGetDevice](#)中的。

## admin-get-user

以下代码示例演示了如何使用 `admin-get-user`。

### AWS CLI

获取用户

此示例获取有关用户名 `jane@example.com` 的信息。

命令:

```
aws cognito-idp admin-get-user --user-pool-id us-west-2_aaaaaaaaa --username
jane@example.com
```

输出:

```
{
  "Username": "4320de44-2322-4620-999b-5e2e1c8df013",
  "Enabled": true,
  "UserStatus": "FORCE_CHANGE_PASSWORD",
  "UserCreateDate": 1548108509.537,
  "UserAttributes": [
    {
      "Name": "sub",
      "Value": "4320de44-2322-4620-999b-5e2e1c8df013"
    },
    {
      "Name": "email_verified",
      "Value": "true"
    },
    {
```

```
    "Name": "phone_number_verified",
    "Value": "true"
  },
  {
    "Name": "phone_number",
    "Value": "+01115551212"
  },
  {
    "Name": "email",
    "Value": "jane@example.com"
  }
],
"UserLastModifiedDate": 1548108509.537
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminGetUser](#)中的。

## admin-initiate-auth

以下代码示例演示了如何使用 admin-initiate-auth。

### AWS CLI

#### 发起身份验证

此示例使用 ADMIN\_NO\_SRP\_AUTH 流程发起对用户名 jane@example.com 的身份验证

客户端必须启用适用于基于服务器的身份验证 ( ADMIN\_NO\_SRP\_AUTH ) 的登录 API。

使用返回值中的会话信息调用 admin-respond-to-auth-challenge。

#### 命令:

```
aws cognito-idp admin-initiate-auth --user-pool-id us-west-2_aaaaaaaaa --client-id 3n4b5urk1ft4fl3mg5e62d9ado --auth-flow ADMIN_NO_SRP_AUTH --auth-parameters USERNAME=jane@example.com,PASSWORD=password
```

#### 输出:

```
{
  "ChallengeName": "NEW_PASSWORD_REQUIRED",
  "Session": "SESSION",
```



```
"ChallengeParameters": {
  "USER_ID_FOR_SRP": "84514837-dcbc-4af1-abff-f3c109334894",
  "requiredAttributes": "[]",
  "userAttributes": "{\"email_verified\": \"true\", \"phone_number_verified\": \"true\", \"phone_number\": \"+01xxx5550100\", \"email\": \"jane@example.com\"}"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminInitiateAuth](#)中的。

## admin-list-devices

以下代码示例演示了如何使用 admin-list-devices。

### AWS CLI

为用户列出设备

此示例列出了用户名为 jane@example.com 的设备。

命令:

```
aws cognito-idp admin-list-devices --user-pool-id us-west-2_aaaaaaaaa --username jane@example.com
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminListDevices](#)中的。

## admin-list-groups-for-user

以下代码示例演示了如何使用 admin-list-groups-for-user。

### AWS CLI

列出用户的群组

此示例列出了用户名为 jane@example.com 的群组。

命令:

```
aws cognito-idp admin-list-groups-for-user --user-pool-id us-west-2_aaaaaaaaa --username diego@example.com
```

输出：

```
{
  "Groups": [
    {
      "Description": "Sample group",
      "Precedence": 1,
      "LastModifiedDate": 1548097827.125,
      "RoleArn": "arn:aws:iam::111111111111:role/SampleRole",
      "GroupName": "SampleGroup",
      "UserPoolId": "us-west-2_aaaaaaaaa",
      "CreationDate": 1548097827.125
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminListGroupsForUser](#)中的。

## admin-list-user-auth-events

以下代码示例演示了如何使用 admin-list-user-auth-events。

### AWS CLI

列出用户的授权事件

此示例列出了用户名 diego@example.com 的授权事件。

命令：

```
aws cognito-idp admin-list-user-auth-events --user-pool-id us-west-2_aaaaaaaaa --
username diego@example.com
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminListUserAuthEvents](#)中的。

## admin-remove-user-from-group

以下代码示例演示了如何使用 admin-remove-user-from-group。

### AWS CLI

从群组中移除用户

此示例将 jane@example.com 从中删除 SampleGroup。

命令:

```
aws cognito-idp admin-remove-user-from-group --user-pool-id us-west-2_aaaaaaaaa --username jane@example.com --group-name SampleGroup
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminRemoveUserFromGroup](#)中的。

## admin-reset-user-password

以下代码示例演示了如何使用 admin-reset-user-password。

AWS CLI

重置用户密码

此示例重置了 diego@example.com 的密码。

命令:

```
aws cognito-idp admin-reset-user-password --user-pool-id us-west-2_aaaaaaaaa --username diego@example.com
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminResetUserPassword](#)中的。

## admin-set-user-mfa-preference

以下代码示例演示了如何使用 admin-set-user-mfa-preference。

AWS CLI

设置用户 MFA 首选项

此示例将 SMS MFA 首选项设置为用户名 diego@example.com。

命令:

```
aws cognito-idp admin-set-user-mfa-preference --user-pool-id us-west-2_aaaaaaaaa --username diego@example.com --sms-mfa-settings Enabled=false,PreferredMfa=false
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminSetUserMfaPreference](#)中的。

## admin-set-user-settings

以下代码示例演示了如何使用 `admin-set-user-settings`。

### AWS CLI

设置用户设置

此示例将用户名 `diego@example.com` 的 MFA 配送首选项设置为 EMAIL。

命令:

```
aws cognito-idp admin-set-user-settings --user-pool-id us-west-2_aaaaaaaaa --username diego@example.com --mfa-options DeliveryMedium=EMAIL
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminSetUserSettings](#)中的。

## admin-update-auth-event-feedback

以下代码示例演示了如何使用 `admin-update-auth-event-feedback`。

### AWS CLI

为授权事件提供反馈

此示例将由 `event-id` 标识的授权事件的反馈值设置为“有效”。

命令:

```
aws cognito-idp admin-update-auth-event-feedback --user-pool-id us-west-2_aaaaaaaaa --username diego@example.com --event-id c2c2cf89-c0d3-482d-aba6-99d78a5b0bfe --feedback-value Valid
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminUpdateAuthEventFeedback](#)中的。

## admin-update-device-status

以下代码示例演示了如何使用 `admin-update-device-status`。

## AWS CLI

### 更新设备状态

此示例将由 device-key 标识的设备的设备记忆状态设置为 not\_remembered。

命令:

```
aws cognito-idp admin-update-device-status --user-pool-id us-west-2_aaaaaaaaa
--username diego@example.com --device-key xxxx --device-remembered-status
not_remembered
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminUpdateDeviceStatus](#)中的。

## admin-update-user-attributes

以下代码示例演示了如何使用 admin-update-user-attributes。

### AWS CLI

#### 更新用户属性

此示例更新了用户 diego@example.com 的自定义用户属性 CustomAttr 1。

命令:

```
aws cognito-idp admin-update-user-attributes --user-pool-id us-
west-2_aaaaaaaaa --username diego@example.com --user-attributes
Name="custom:CustomAttr1",Value="Purple"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminUpdateUserAttributes](#)中的。

## change-password

以下代码示例演示了如何使用 change-password。

### AWS CLI

#### 更改密码

此示例更改了密码。

命令:

```
aws cognito-idp change-password --previous-password OldPassword --proposed-password  
NewPassword --access-token ACCESS_TOKEN
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ChangePassword](#)中的。

## confirm-forgot-password

以下代码示例演示了如何使用 `confirm-forgot-password`。

AWS CLI

确认忘记的密码

此示例确认忘记了用户名 `diego@example.com` 的密码。

命令:

```
aws cognito-idp confirm-forgot-password --client-id 3n4b5urk1ft4f13mg5e62d9ado --  
username=diego@example.com --password PASSWORD --confirmation-code CONF_CODE
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ConfirmForgotPassword](#)中的。

## confirm-sign-up

以下代码示例演示了如何使用 `confirm-sign-up`。

AWS CLI

确认注册

此示例确认用户名 `diego@example.com` 的注册。

命令:

```
aws cognito-idp confirm-sign-up --client-id 3n4b5urk1ft4f13mg5e62d9ado --  
username=diego@example.com --confirmation-code CONF_CODE
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ConfirmSignUp](#)中的。

## create-group

以下代码示例演示了如何使用 create-group。

### AWS CLI

#### 创建群组

此示例创建了一个带有描述的群组。

命令:

```
aws cognito-idp create-group --user-pool-id us-west-2_aaaaaaaaa --group-name
MyNewGroup --description "New group."
```

输出:

```
{
  "Group": {
    "GroupName": "MyNewGroup",
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "Description": "New group.",
    "LastModifiedDate": 1548270073.795,
    "CreationDate": 1548270073.795
  }
}
```

#### 创建具有角色和优先级的群组

此示例创建了一个带有描述的群组。它还包括角色和优先级。

命令:

```
aws cognito-idp create-group --user-pool-id us-west-2_aaaaaaaaa --group-
name MyNewGroupWithRole --description "New group with a role." --role-arn
arn:aws:iam::111111111111:role/MyNewGroupRole --precedence 2
```

输出:

```
{
  "Group": {
    "GroupName": "MyNewGroupWithRole",
```

```
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "Description": "New group with a role.",
    "RoleArn": "arn:aws:iam::111111111111:role/MyNewGroupRole",
    "Precedence": 2,
    "LastModifiedDate": 1548270211.761,
    "CreationDate": 1548270211.761
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateGroup](#)中的。

## create-user-import-job

以下代码示例演示了如何使用 create-user-import-job。

### AWS CLI

#### 创建用户导入任务

此示例创建了一个名为的用户导入作业 MyImportJob。

有关导入用户的更多信息，请参阅从 CSV 文件将用户导入用户池。

#### 命令:

```
aws cognito-idp create-user-import-job --user-pool-id us-west-2_aaaaaaaaa --
job-name MyImportJob --cloud-watch-logs-role-arn arn:aws:iam::111111111111:role/
CognitoCloudWatchLogsRole
```

#### 输出:

```
{
  "UserImportJob": {
    "JobName": "MyImportJob",
    "JobId": "import-qQ0DCt2fRh",
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "PreSignedUrl": "PRE_SIGNED_URL",
    "CreationDate": 1548271795.471,
    "Status": "Created",
    "CloudWatchLogsRoleArn": "arn:aws:iam::111111111111:role/
CognitoCloudWatchLogsRole",
    "ImportedUsers": 0,
  }
}
```



```
    "SkippedUsers": 0,  
    "FailedUsers": 0  
  }  
}
```

使用预签名 URL 上传带有 curl 的.csv 文件：

命令:

```
curl -v -T "PATH_TO_CSV_FILE" -H "x-amz-server-side-encryption:aws:kms"  
"PRE_SIGNED_URL"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUserImportJob](#)中的。

## create-user-pool-client

以下代码示例演示了如何使用 create-user-pool-client。

### AWS CLI

#### 创建用户池客户端

此示例创建了一个具有两个显式授权流程的新用户池客户端：USER\_PASSWORD\_AUTH 和 ADMIN\_NO\_SRP\_AUTH。

命令:

```
aws cognito-idp create-user-pool-client --user-pool-id us-west-2_aaaaaaaaa  
--client-name MyNewClient --no-generate-secret --explicit-auth-flows  
"USER_PASSWORD_AUTH" "ADMIN_NO_SRP_AUTH"
```

输出：

```
{  
  "UserPoolClient": {  
    "UserPoolId": "us-west-2_aaaaaaaaa",  
    "ClientName": "MyNewClient",  
    "ClientId": "6p3bs000no6a4ue1idruvd05ad",  
    "LastModifiedDate": 1548697449.497,  
    "CreationDate": 1548697449.497,  
    "RefreshTokenValidity": 30,  
    "ExplicitAuthFlows": [  
      "USER_PASSWORD_AUTH",  
      "ADMIN_NO_SRP_AUTH"  
    ]  
  }  
}
```

```
        "USER_PASSWORD_AUTH",
        "ADMIN_NO_SRP_AUTH"
    ],
    "AllowedOAuthFlowsUserPoolClient": false
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUserPoolClient](#)中的。

## create-user-pool-domain

以下代码示例演示了如何使用 `create-user-pool-domain`。

### AWS CLI

#### 创建用户池域

此示例使用两个明确的授权流程创建了一个新的用户池域：USER\_PASSWORD\_AUTH 和 ADMIN\_NO\_SRP\_AUTH。

#### 命令：

```
aws cognito-idp create-user-pool-domain --user-pool-id us-west-2_aaaaaaaaa --domain
my-new-domain
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUserPoolDomain](#)中的。

## create-user-pool

以下代码示例演示了如何使用 `create-user-pool`。

### AWS CLI

#### 创建最低配置的用户池

此示例创建了一个 MyUserPool 使用默认值命名的用户池。没有必要的属性，也没有应用程序客户端。MFA 和高级安全功能已禁用。

#### 命令：

```
aws cognito-idp create-user-pool --pool-name MyUserPool
```

输出：

```
{
  "UserPool": {
    "SchemaAttributes": [
      {
        "Name": "sub",
        "StringAttributeConstraints": {
          "MinLength": "1",
          "MaxLength": "2048"
        },
        "DeveloperOnlyAttribute": false,
        "Required": true,
        "AttributeDataType": "String",
        "Mutable": false
      },
      {
        "Name": "name",
        "StringAttributeConstraints": {
          "MinLength": "0",
          "MaxLength": "2048"
        },
        "DeveloperOnlyAttribute": false,
        "Required": false,
        "AttributeDataType": "String",
        "Mutable": true
      },
      {
        "Name": "given_name",
        "StringAttributeConstraints": {
          "MinLength": "0",
          "MaxLength": "2048"
        },
        "DeveloperOnlyAttribute": false,
        "Required": false,
        "AttributeDataType": "String",
        "Mutable": true
      },
      {
        "Name": "family_name",
        "StringAttributeConstraints": {
          "MinLength": "0",
          "MaxLength": "2048"
        }
      }
    ]
  }
}
```

```
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "middle_name",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "nickname",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "preferred_username",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "profile",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },

```

```
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "picture",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "website",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "email",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "AttributeDataType": "Boolean",
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "Name": "email_verified",
    "Mutable": true
  }
}
```

```
  },
  {
    "Name": "gender",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "birthdate",
    "StringAttributeConstraints": {
      "MinLength": "10",
      "MaxLength": "10"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "zoneinfo",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "locale",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  }
}
```

```
    },
    {
      "Name": "phone_number",
      "StringAttributeConstraints": {
        "MinLength": "0",
        "MaxLength": "2048"
      },
      "DeveloperOnlyAttribute": false,
      "Required": false,
      "AttributeDataType": "String",
      "Mutable": true
    },
    {
      "AttributeDataType": "Boolean",
      "DeveloperOnlyAttribute": false,
      "Required": false,
      "Name": "phone_number_verified",
      "Mutable": true
    },
    {
      "Name": "address",
      "StringAttributeConstraints": {
        "MinLength": "0",
        "MaxLength": "2048"
      },
      "DeveloperOnlyAttribute": false,
      "Required": false,
      "AttributeDataType": "String",
      "Mutable": true
    },
    {
      "Name": "updated_at",
      "NumberAttributeConstraints": {
        "MinValue": "0"
      },
      "DeveloperOnlyAttribute": false,
      "Required": false,
      "AttributeDataType": "Number",
      "Mutable": true
    }
  ],
  "MfaConfiguration": "OFF",
  "Name": "MyUserPool",
  "LastModifiedDate": 1547833345.777,
```

```

    "AdminCreateUserConfig": {
      "UnusedAccountValidityDays": 7,
      "AllowAdminCreateUserOnly": false
    },
    "EmailConfiguration": {},
    "Policies": {
      "PasswordPolicy": {
        "RequireLowercase": true,
        "RequireSymbols": true,
        "RequireNumbers": true,
        "MinimumLength": 8,
        "RequireUppercase": true
      }
    },
    "CreationDate": 1547833345.777,
    "EstimatedNumberOfUsers": 0,
    "Id": "us-west-2_aaaaaaaaa",
    "LambdaConfig": {}
  }
}

```

### 创建具有两个必要属性的用户池

此示例创建了一个用户池 MyUserPool。该池配置为接受电子邮件作为用户名属性。它还使用 Amazon Simple Email Service 将电子邮件源地址设置为经过验证的地址。

#### 命令:

```

aws cognito-idp create-user-pool --pool-name MyUserPool --username-attributes "email" --email-configuration=SourceArn="arn:aws:ses:us-east-1:111111111111:identity/jane@example.com",ReplyToEmailAddress="jane@example.com"

```

#### 输出:

```

{
  "UserPool": {
    "SchemaAttributes": [
      {
        "Name": "sub",
        "StringAttributeConstraints": {
          "MinLength": "1",
          "MaxLength": "2048"
        }
      }
    ]
  }
}

```



```
    },
    "DeveloperOnlyAttribute": false,
    "Required": true,
    "AttributeDataType": "String",
    "Mutable": false
  },
  {
    "Name": "name",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "given_name",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "family_name",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "middle_name",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    }
  }
}
```

```
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "nickname",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "preferred_username",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "profile",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "picture",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    }
  }
}
```

```
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "website",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "email",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "AttributeDataType": "Boolean",
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "Name": "email_verified",
    "Mutable": true
  },
  {
    "Name": "gender",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
```

```
    "Mutable": true
  },
  {
    "Name": "birthdate",
    "StringAttributeConstraints": {
      "MinLength": "10",
      "MaxLength": "10"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "zoneinfo",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "locale",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "phone_number",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
```

```
    "Mutable": true
  },
  {
    "AttributeDataType": "Boolean",
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "Name": "phone_number_verified",
    "Mutable": true
  },
  {
    "Name": "address",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "updated_at",
    "NumberAttributeConstraints": {
      "MinValue": "0"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "Number",
    "Mutable": true
  }
],
"MfaConfiguration": "OFF",
"Name": "MyUserPool",
"LastModifiedDate": 1547837788.189,
"AdminCreateUserConfig": {
  "UnusedAccountValidityDays": 7,
  "AllowAdminCreateUserOnly": false
},
"EmailConfiguration": {
  "ReplyToEmailAddress": "jane@example.com",
  "SourceArn": "arn:aws:ses:us-east-1:111111111111:identity/
jane@example.com"
},
"Policies": {
```

```
    "PasswordPolicy": {
      "RequireLowercase": true,
      "RequireSymbols": true,
      "RequireNumbers": true,
      "MinimumLength": 8,
      "RequireUppercase": true
    }
  },
  "UsernameAttributes": [
    "email"
  ],
  "CreationDate": 1547837788.189,
  "EstimatedNumberOfUsers": 0,
  "Id": "us-west-2_aaaaaaaaa",
  "LambdaConfig": {}
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUserPool](#)中的。

## delete-group

以下代码示例演示了如何使用 delete-group。

### AWS CLI

#### 删除群组

此示例删除群组。

命令:

```
aws cognito-idp delete-group --user-pool-id us-west-2_aaaaaaaaa --group-name
MyGroupName
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteGroup](#)中的。

## delete-identity-provider

以下代码示例演示了如何使用 delete-identity-provider。

## AWS CLI

### 删除身份提供商

此示例删除了身份提供商。

命令:

```
aws cognito-idp delete-identity-provider --user-pool-id us-west-2_aaaaaaaaa --
provider-name Facebook
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteIdentityProvider](#)中的。

## **delete-resource-server**

以下代码示例演示了如何使用 delete-resource-server。

## AWS CLI

### 删除资源服务器

此示例删除了一个名为 weather.example.com 的资源服务器。

命令:

```
aws cognito-idp delete-resource-server --user-pool-id us-west-2_aaaaaaaaa --
identifier weather.example.com
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteResourceServer](#)中的。

## **delete-user-attributes**

以下代码示例演示了如何使用 delete-user-attributes。

## AWS CLI

### 删除用户属性

此示例删除了用户属性“FAVORITE\_ANIMAL”。

命令:

```
aws cognito-idp delete-user-attributes --access-token ACCESS_TOKEN --user-attribute-names "FAVORITE_ANIMAL"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUserAttributes](#)中的。

## delete-user-pool-client

以下代码示例演示了如何使用 delete-user-pool-client。

### AWS CLI

删除用户池客户端

此示例删除用户池客户端。

命令:

```
aws cognito-idp delete-user-pool-client --user-pool-id us-west-2_aaaaaaaaa --client-id 38fjsnc484p94kpbsnet7mpld0
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUserPoolClient](#)中的。

## delete-user-pool-domain

以下代码示例演示了如何使用 delete-user-pool-domain。

### AWS CLI

删除用户池域

以下delete-user-pool-domain示例删除名为的用户池域 my-domain

```
aws cognito-idp delete-user-pool-domain \
  --user-pool-id us-west-2_aaaaaaaaa \
  --domain my-domain
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUserPoolDomain](#)中的。

## delete-user-pool

以下代码示例演示了如何使用 delete-user-pool。



## AWS CLI

### 删除用户池

此示例使用用户池 ID `us-west-2_aaaaaaaaa` 删除用户池。

命令:

```
aws cognito-idp delete-user-pool --user-pool-id us-west-2_aaaaaaaaa
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUserPool](#)中的。

## **delete-user**

以下代码示例演示了如何使用 `delete-user`。

## AWS CLI

### 删除用户

此示例删除一个用户。

命令:

```
aws cognito-idp delete-user --access-token ACCESS_TOKEN
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUser](#)中的。

## **describe-identity-provider**

以下代码示例演示了如何使用 `describe-identity-provider`。

## AWS CLI

### 描述身份提供者

此示例描述了一个名为 Facebook 的身份提供商。

命令:

```
aws cognito-idp describe-identity-provider --user-pool-id us-west-2_aaaaaaaaa --
provider-name Facebook
```

输出：

```
{
  "IdentityProvider": {
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "ProviderName": "Facebook",
    "ProviderType": "Facebook",
    "ProviderDetails": {
      "attributes_url": "https://graph.facebook.com/me?fields=",
      "attributes_url_add_attributes": "true",
      "authorize_scopes": "myscope",
      "authorize_url": "https://www.facebook.com/v2.9/dialog/oauth",
      "client_id": "11111",
      "client_secret": "11111",
      "token_request_method": "GET",
      "token_url": "https://graph.facebook.com/v2.9/oauth/access_token"
    },
    "AttributeMapping": {
      "username": "id"
    },
    "IdpIdentifiers": [],
    "LastModifiedDate": 1548105901.736,
    "CreationDate": 1548105901.736
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeIdentityProvider](#)中的。

## describe-resource-server

以下代码示例演示了如何使用 `describe-resource-server`。

### AWS CLI

描述资源服务器

此示例描述了资源服务器 `weather.example.com`。

命令：

```
aws cognito-idp describe-resource-server --user-pool-id us-west-2_aaaaaaaaa --
identifier weather.example.com
```

输出：

```
{
  "ResourceServer": {
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "Identifier": "weather.example.com",
    "Name": "Weather",
    "Scopes": [
      {
        "ScopeName": "weather.update",
        "ScopeDescription": "Update weather forecast"
      },
      {
        "ScopeName": "weather.read",
        "ScopeDescription": "Read weather forecasts"
      },
      {
        "ScopeName": "weather.delete",
        "ScopeDescription": "Delete a weather forecast"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeResourceServer](#)中的。

## describe-risk-configuration

以下代码示例演示了如何使用 `describe-risk-configuration`。

### AWS CLI

#### 描述风险配置

此示例描述了与池 `us-west-2_aaaaaaaaa` 相关的风险配置。

命令：

```
aws cognito-idp describe-risk-configuration --user-pool-id us-west-2_aaaaaaaaa
```

输出：

```
{
  "RiskConfiguration": {
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "CompromisedCredentialsRiskConfiguration": {
      "EventFilter": [
        "SIGN_IN",
        "SIGN_UP",
        "PASSWORD_CHANGE"
      ],
      "Actions": {
        "EventAction": "BLOCK"
      }
    },
    "AccountTakeoverRiskConfiguration": {
      "NotifyConfiguration": {
        "From": "diego@example.com",
        "ReplyTo": "diego@example.com",
        "SourceArn": "arn:aws:ses:us-east-1:111111111111:identity/diego@example.com",
        "BlockEmail": {
          "Subject": "Blocked sign-in attempt",
          "HtmlBody": "<!DOCTYPE html>\n<html>\n<head>\n\t<title>HTML email context</title>\n\t<meta charset=\"utf-8\">\n</head>\n<body>\n<pre>We blocked an unrecognized sign-in to your account with this information:\n<ul>\n<li>Time: {login-time}</li>\n<li>Device: {device-name}</li>\n<li>Location: {city}, {country}</li>\n</ul>\nIf this sign-in was not by you, you should change your password and notify us by clicking on <a href={one-click-link-invalid}>this link</a>\nIf this sign-in was by you, you can follow <a href={one-click-link-valid}>this link</a> to let us know</pre>\n</body>\n</html>",
          "TextBody": "We blocked an unrecognized sign-in to your account with this information:\nTime: {login-time}\nDevice: {device-name}\nLocation: {city}, {country}\nIf this sign-in was not by you, you should change your password and notify us by clicking on {one-click-link-invalid}\nIf this sign-in was by you, you can follow {one-click-link-valid} to let us know"
        },
        "NoActionEmail": {
          "Subject": "New sign-in attempt",
          "HtmlBody": "<!DOCTYPE html>\n<html>\n<head>\n\t<title>HTML email context</title>\n\t<meta charset=\"utf-8\">\n</head>\n<body>\n<pre>We observed an unrecognized sign-in to your account with this information:\n<ul>\n<li>Time: {login-time}</li>\n<li>Device: {device-name}</li>\n<li>Location: {city}, {country}</li>\n</ul>\nIf this sign-in was not by you, you should change your"
        }
      }
    }
  }
}
```

```

password and notify us by clicking on <a href={one-click-link-invalid}>this link</a>\nIf this sign-in was by you, you can follow <a href={one-click-link-valid}>this link</a> to let us know</pre>\n</body>\n</html>",
    "TextBody": "We observed an unrecognized sign-in to your account with this information:\nTime: {login-time}\nDevice: {device-name}\nLocation: {city}, {country}\nIf this sign-in was not by you, you should change your password and notify us by clicking on {one-click-link-invalid}\nIf this sign-in was by you, you can follow {one-click-link-valid} to let us know"
  },
  "MfaEmail": {
    "Subject": "New sign-in attempt",
    "HtmlBody": "<!DOCTYPE html>\n<html>\n<head>\n\t<title>HTML email context</title>\n\t<meta charset=\"utf-8\">\n</head>\n<body>\n<pre>We required you to use multi-factor authentication for the following sign-in attempt:\n<ul>\n<li>Time: {login-time}</li>\n<li>Device: {device-name}</li>\n<li>Location: {city}, {country}</li>\n</ul>\nIf this sign-in was not by you, you should change your password and notify us by clicking on <a href={one-click-link-invalid}>this link</a>\nIf this sign-in was by you, you can follow <a href={one-click-link-valid}>this link</a> to let us know</pre>\n</body>\n</html>",
    "TextBody": "We required you to use multi-factor authentication for the following sign-in attempt:\nTime: {login-time}\nDevice: {device-name}\nLocation: {city}, {country}\nIf this sign-in was not by you, you should change your password and notify us by clicking on {one-click-link-invalid}\nIf this sign-in was by you, you can follow {one-click-link-valid} to let us know"
  }
},
"Actions": {
  "LowAction": {
    "Notify": true,
    "EventAction": "NO_ACTION"
  },
  "MediumAction": {
    "Notify": true,
    "EventAction": "MFA_IF_CONFIGURED"
  },
  "HighAction": {
    "Notify": true,
    "EventAction": "MFA_IF_CONFIGURED"
  }
}
}
}
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeRiskConfiguration](#)中的。

## describe-user-import-job

以下代码示例演示了如何使用 `describe-user-import-job`。

### AWS CLI

描述用户导入任务

此示例描述了用户输入作业。

有关导入用户的更多信息，请参阅从 CSV 文件将用户导入用户池。

命令:

```
aws cognito-idp describe-user-import-job --user-pool-id us-west-2_aaaaaaaaa --job-id import-TZqNQvDRnW
```

输出 :

```
{
  "UserImportJob": {
    "JobName": "import-Test1",
    "JobId": "import-TZqNQvDRnW",
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "PreSignedUrl": "PRE_SIGNED URL",
    "CreationDate": 1548271708.512,
    "Status": "Created",
    "CloudWatchLogsRoleArn": "arn:aws:iam::111111111111:role/CognitoCloudWatchLogsRole",
    "ImportedUsers": 0,
    "SkippedUsers": 0,
    "FailedUsers": 0
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeUserImportJob](#)中的。

## describe-user-pool-client

以下代码示例演示了如何使用 `describe-user-pool-client`。

## AWS CLI

### 描述用户池客户端

此示例描述了用户池客户端。

命令:

```
aws cognito-idp describe-user-pool-client --user-pool-id us-west-2_aaaaaaaaa --client-id 38fjsnc484p94kpqsnet7mpld0
```

输出:

```
{
  "UserPoolClient": {
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "ClientName": "MyApp",
    "ClientId": "38fjsnc484p94kpqsnet7mpld0",
    "ClientSecret": "CLIENT_SECRET",
    "LastModifiedDate": 1548108676.163,
    "CreationDate": 1548108676.163,
    "RefreshTokenValidity": 30,
    "ReadAttributes": [
      "address",
      "birthdate",
      "custom:CustomAttr1",
      "custom:CustomAttr2",
      "email",
      "email_verified",
      "family_name",
      "gender",
      "given_name",
      "locale",
      "middle_name",
      "name",
      "nickname",
      "phone_number",
      "phone_number_verified",
      "picture",
      "preferred_username",
      "profile",
      "updated_at",
      "website",
      "zoneinfo"
    ]
  }
}
```

```
    ],
    "WriteAttributes": [
      "address",
      "birthdate",
      "custom:CustomAttr1",
      "custom:CustomAttr2",
      "email",
      "family_name",
      "gender",
      "given_name",
      "locale",
      "middle_name",
      "name",
      "nickname",
      "phone_number",
      "picture",
      "preferred_username",
      "profile",
      "updated_at",
      "website",
      "zoneinfo"
    ],
    "ExplicitAuthFlows": [
      "ADMIN_NO_SRP_AUTH",
      "USER_PASSWORD_AUTH"
    ],
    "AllowedOAuthFlowsUserPoolClient": false
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeUserPoolClient](#)中的。

## describe-user-pool-domain

以下代码示例演示了如何使用 describe-user-pool-domain。

### AWS CLI

描述用户池客户端

此示例描述了一个名为 my-domain 的用户池域。

命令:



```
aws cognito-idp describe-user-pool-domain --domain my-domain
```

输出：

```
{
  "DomainDescription": {
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "AWSAccountId": "111111111111",
    "Domain": "my-domain",
    "S3Bucket": "aws-cognito-prod-pdx-assets",
    "CloudFrontDistribution": "aaaaaaaaaaaaa.cloudfront.net",
    "Version": "20190128175402",
    "Status": "ACTIVE",
    "CustomDomainConfig": {}
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeUserPoolDomain](#)中的。

## describe-user-pool

以下代码示例演示了如何使用 describe-user-pool。

### AWS CLI

#### 描述用户池

此示例描述了用户池 ID 为 us-west-2\_aaaaaaaaa 的用户池。

命令：

```
aws cognito-idp describe-user-pool --user-pool-id us-west-2_aaaaaaaaa
```

输出：

```
{
  "UserPool": {
    "SmsVerificationMessage": "Your verification code is {####}. ",
    "SchemaAttributes": [
      {
        "Name": "sub",
        "StringAttributeConstraints": {
```

```
        "MinLength": "1",
        "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": true,
    "AttributeDataType": "String",
    "Mutable": false
},
{
    "Name": "name",
    "StringAttributeConstraints": {
        "MinLength": "0",
        "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
},
{
    "Name": "given_name",
    "StringAttributeConstraints": {
        "MinLength": "0",
        "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
},
{
    "Name": "family_name",
    "StringAttributeConstraints": {
        "MinLength": "0",
        "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
},
{
    "Name": "middle_name",
    "StringAttributeConstraints": {
```

```
        "MinLength": "0",
        "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
},
{
    "Name": "nickname",
    "StringAttributeConstraints": {
        "MinLength": "0",
        "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
},
{
    "Name": "preferred_username",
    "StringAttributeConstraints": {
        "MinLength": "0",
        "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
},
{
    "Name": "profile",
    "StringAttributeConstraints": {
        "MinLength": "0",
        "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
},
{
    "Name": "picture",
    "StringAttributeConstraints": {
```

```
        "MinLength": "0",
        "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
},
{
    "Name": "website",
    "StringAttributeConstraints": {
        "MinLength": "0",
        "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
},
{
    "Name": "email",
    "StringAttributeConstraints": {
        "MinLength": "0",
        "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": true,
    "AttributeDataType": "String",
    "Mutable": true
},
{
    "AttributeDataType": "Boolean",
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "Name": "email_verified",
    "Mutable": true
},
{
    "Name": "gender",
    "StringAttributeConstraints": {
        "MinLength": "0",
        "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
```

```
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "birthdate",
    "StringAttributeConstraints": {
      "MinLength": "10",
      "MaxLength": "10"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "zoneinfo",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "locale",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
    "Required": false,
    "AttributeDataType": "String",
    "Mutable": true
  },
  {
    "Name": "phone_number",
    "StringAttributeConstraints": {
      "MinLength": "0",
      "MaxLength": "2048"
    },
    "DeveloperOnlyAttribute": false,
```

```
        "Required": false,
        "AttributeDataType": "String",
        "Mutable": true
    },
    {
        "AttributeDataType": "Boolean",
        "DeveloperOnlyAttribute": false,
        "Required": false,
        "Name": "phone_number_verified",
        "Mutable": true
    },
    {
        "Name": "address",
        "StringAttributeConstraints": {
            "MinLength": "0",
            "MaxLength": "2048"
        },
        "DeveloperOnlyAttribute": false,
        "Required": false,
        "AttributeDataType": "String",
        "Mutable": true
    },
    {
        "Name": "updated_at",
        "NumberAttributeConstraints": {
            "MinValue": "0"
        },
        "DeveloperOnlyAttribute": false,
        "Required": false,
        "AttributeDataType": "Number",
        "Mutable": true
    }
],
"EmailVerificationSubject": "Your verification code",
"MfaConfiguration": "OFF",
"Name": "MyUserPool",
"EmailVerificationMessage": "Your verification code is {#####}. ",
"SmsAuthenticationMessage": "Your authentication code is {#####}. ",
"LastModifiedDate": 1547763720.822,
"AdminCreateUserConfig": {
    "InviteMessageTemplate": {
        "EmailMessage": "Your username is {username} and temporary password is
{#####}. ",
        "EmailSubject": "Your temporary password",
```

```

        "SMSMessage": "Your username is {username} and temporary password is
{####}. "
    },
    "UnusedAccountValidityDays": 7,
    "AllowAdminCreateUserOnly": false
},
"EmailConfiguration": {
    "ReplyToEmailAddress": "myemail@mydomain.com"
    "SourceArn": "arn:aws:ses:us-east-1:000000000000:identity/
myemail@mydomain.com"
},
"AutoVerifiedAttributes": [
    "email"
],
"Policies": {
    "PasswordPolicy": {
        "RequireLowercase": true,
        "RequireSymbols": true,
        "RequireNumbers": true,
        "MinimumLength": 8,
        "RequireUppercase": true
    }
},
"UserPoolTags": {},
"UsernameAttributes": [
    "email"
],
"CreationDate": 1547763720.822,
"EstimatedNumberOfUsers": 1,
"Id": "us-west-2_aaaaaaaaa",
"LambdaConfig": {}
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeUserPool](#)中的。

## forget-device

以下代码示例演示了如何使用 forget-device。

### AWS CLI

#### 忘记设备

此示例忘记了设备和设备。

命令:

```
aws cognito-idp forget-device --device-key us-west-2_abcd_1234-5678
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ForgetDevice](#)中的。

## forgot-password

以下代码示例演示了如何使用 forgot-password。

AWS CLI

强制更改密码

以下forgot-password示例向 jane@example.com 发送了一条消息，要求他们更改密码。

```
aws cognito-idp forgot-password --client-id 38fjsnc484p94kpbsnet7mpld0 --username jane@example.com
```

输出：

```
{
  "CodeDeliveryDetails": {
    "Destination": "j***@e***.com",
    "DeliveryMedium": "EMAIL",
    "AttributeName": "email"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ForgotPassword](#)中的。

## get-csv-header

以下代码示例演示了如何使用 get-csv-header。

AWS CLI

创建 csv 标题

此示例创建了一个 csv 标头。



有关导入用户的更多信息，请参阅从 CSV 文件将用户导入用户池。

命令：

```
aws cognito-idp get-csv-header --user-pool-id us-west-2_aaaaaaaaa
```

输出：

```
{
  "UserPoolId": "us-west-2_aaaaaaaaa",
  "CSVHeader": [
    "name",
    "given_name",
    "family_name",
    "middle_name",
    "nickname",
    "preferred_username",
    "profile",
    "picture",
    "website",
    "email",
    "email_verified",
    "gender",
    "birthdate",
    "zoneinfo",
    "locale",
    "phone_number",
    "phone_number_verified",
    "address",
    "updated_at",
    "cognito:mfa_enabled",
    "cognito:username"
  ]
}
```

... 从 CSV 文件将用户导入用户池：<https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pools-using-import-tool.html>

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetCsvHeader](#) 中的。

## get-group

以下代码示例演示了如何使用 get-group。

## AWS CLI

获取有关群组的信息

此示例获取有关名为的群组的信息 MyGroup。

命令:

```
aws cognito-idp get-group --user-pool-id us-west-2_aaaaaaaa --group-name MyGroup
```

输出 :

```
{
  "Group": {
    "GroupName": "MyGroup",
    "UserPoolId": "us-west-2_aaaaaaaa",
    "Description": "A sample group.",
    "LastModifiedDate": 1548270073.795,
    "CreationDate": 1548270073.795
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetGroup](#)中的。

## get-signing-certificate

以下代码示例演示了如何使用 get-signing-certificate。

## AWS CLI

获取签名证书

此示例获取用户池的签名证书。

命令:

```
aws cognito-idp get-signing-certificate --user-pool-id us-west-2_aaaaaaaa
```

输出 :

```
{
  "Certificate": "CERTIFICATE_DATA"
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSigningCertificate](#)中的。

## get-ui-customization

以下代码示例演示了如何使用 `get-ui-customization`。

### AWS CLI

获取用户界面自定义信息

此示例获取用户池的用户界面自定义信息。

命令:

```
aws cognito-idp get-ui-customization --user-pool-id us-west-2_aaaaaaaaa
```

输出:

```
{
  "UICustomization": {
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "ClientId": "ALL",
    "ImageUrl": "https://aaaaaaaaaaaaa.cloudfront.net/us-west-2_aaaaaaaaa/
ALL/20190128231240/assets/images/image.jpg",
    "CSS": ".logo-customizable {\n\tmax-width: 60%;\n\tmax-height: 30%;
\n}\n\n.banner-customizable {\n\tpadding: 25px 0px 25px 10px;\n\tbackground-color:
lightgray;\n}\n\n.label-customizable {\n\tfont-weight: 300;\n}\n\n.textDescription-
customizable {\n\tpadding-top: 10px;\n\tpadding-bottom: 10px;\n\tdisplay: block;
\n\tfont-size: 16px;\n}\n\n.idpDescription-customizable {\n\tpadding-top: 10px;\n
\tpadding-bottom: 10px;\n\tdisplay: block;\n\tfont-size: 16px;\n}\n\n.legalText-
customizable {\n\tcolor: #747474;\n\tfont-size: 11px;\n}\n\n.submitButton-customizable
{\n\tfont-size: 14px;\n\tfont-weight: bold;\n\tmargin: 20px 0px 10px 0px;\n
\theight: 40px;\n\twidth: 100%;\n\tcolor: #fff;\n\tbackground-color: #337ab7;
\n}\n\n.submitButton-customizable:hover {\n\tcolor: #fff;\n\tbackground-color:
#286090;\n}\n\n.errorMessage-customizable {\n\tpadding: 5px;\n\tfont-size: 14px;
\n\twidth: 100%;\n\tbackground: #F5F5F5;\n\tborder: 2px solid #D64958;\n\tcolor:
#D64958;\n}\n\n.inputField-customizable {\n\twidth: 100%;\n\theight: 34px;\n\tcolor:
#555;\n\tbackground-color: #fff;\n\tborder: 1px solid #ccc;\n}\n\n.inputField-
customizable:focus {\n\tborder-color: #66afe9;\n\toutline: 0;\n}\n\n.idpButton-
customizable {\n\theight: 40px;\n\twidth: 100%;\n\ttext-align: center;\n\tmargin-
```

```

bottom: 15px;\n\tcolor: #fff;\n\tbackground-color: #5bc0de;\n\tborder-color:
#46b8da;\n}\n.idpButton-customizable:hover {\n\tcolor: #fff;\n\tbackground-color:
#31b0d5;\n}\n.socialButton-customizable {\n\theight: 40px;\n\ttext-align: left;
\n\twidth: 100%;\n\tmargin-bottom: 15px;\n}\n.redirect-customizable {\n\ttext-
align: center;\n}\n.passwordCheck-notValid-customizable {\n\tcolor: #DF3312;
\n}\n.passwordCheck-valid-customizable {\n\tcolor: #19BF00;\n}\n.background-
customizable {\n\tbackground-color: #faf;\n}\n",
  "CSSVersion": "20190128231240"
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetUiCustomization](#)中的。

## list-user-import-jobs

以下代码示例演示了如何使用 `list-user-import-jobs`。

### AWS CLI

#### 列出用户导入任务

此示例列出了用户导入任务。

有关导入用户的更多信息，请参阅从 CSV 文件将用户导入用户池。

#### 命令:

```
aws cognito-idp list-user-import-jobs --user-pool-id us-west-2_aaaaaaaaa --max-
results 20
```

#### 输出:

```
{
  "UserImportJobs": [
    {
      "JobName": "Test2",
      "JobId": "import-d00nwGA3mV",
      "UserPoolId": "us-west-2_aaaaaaaaa",
      "PreSignedUrl": "PRE_SIGNED_URL",
      "CreationDate": 1548272793.069,
      "Status": "Created",
      "CloudWatchLogsRoleArn": "arn:aws:iam::111111111111:role/
CognitoCloudWatchLogsRole",

```

```

    "ImportedUsers": 0,
    "SkippedUsers": 0,
    "FailedUsers": 0
  },
  {
    "JobName": "Test1",
    "JobId": "import-qQ0DCt2fRh",
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "PreSignedUrl": "PRE_SIGNED_URL",
    "CreationDate": 1548271795.471,
    "Status": "Created",
    "CloudWatchLogsRoleArn": "arn:aws:iam::111111111111:role/
CognitoCloudWatchLogsRole",
    "ImportedUsers": 0,
    "SkippedUsers": 0,
    "FailedUsers": 0
  },
  {
    "JobName": "import-Test1",
    "JobId": "import-TZqNQvDRnW",
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "PreSignedUrl": "PRE_SIGNED_URL",
    "CreationDate": 1548271708.512,
    "StartDate": 1548277247.962,
    "CompletionDate": 1548277248.912,
    "Status": "Failed",
    "CloudWatchLogsRoleArn": "arn:aws:iam::111111111111:role/
CognitoCloudWatchLogsRole",
    "ImportedUsers": 0,
    "SkippedUsers": 0,
    "FailedUsers": 1,
    "CompletionMessage": "Too many users have failed or been skipped during
the import."
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUserImportJobs](#)中的。

## list-user-pools

以下代码示例演示了如何使用 list-user-pools。

## AWS CLI

### 列出用户池

此示例最多列出 20 个用户池。

命令:

```
aws cognito-idp list-user-pools --max-results 20
```

输出:

```
{
  "UserPools": [
    {
      "CreationDate": 1547763720.822,
      "LastModifiedDate": 1547763720.822,
      "LambdaConfig": {},
      "Id": "us-west-2_aaaaaaaaa",
      "Name": "MyUserPool"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUserPools](#)中的。

### list-users-in-group

以下代码示例演示了如何使用 list-users-in-group。

## AWS CLI

### 列出群组中的用户

此示例列出了群组中的用户 MyGroup。

命令:

```
aws cognito-idp list-users-in-group --user-pool-id us-west-2_aaaaaaaaa --group-name
MyGroup
```

输出：

```
{
  "Users": [
    {
      "Username": "acf10624-80bb-401a-ac61-607bee2110ec",
      "Attributes": [
        {
          "Name": "sub",
          "Value": "acf10624-80bb-401a-ac61-607bee2110ec"
        },
        {
          "Name": "custom:CustomAttr1",
          "Value": "New Value!"
        },
        {
          "Name": "email",
          "Value": "jane@example.com"
        }
      ],
      "UserCreateDate": 1548102770.284,
      "UserLastModifiedDate": 1548103204.893,
      "Enabled": true,
      "UserStatus": "CONFIRMED"
    },
    {
      "Username": "22704aa3-fc10-479a-97eb-2af5806bd327",
      "Attributes": [
        {
          "Name": "sub",
          "Value": "22704aa3-fc10-479a-97eb-2af5806bd327"
        },
        {
          "Name": "email_verified",
          "Value": "true"
        },
        {
          "Name": "email",
          "Value": "diego@example.com"
        }
      ],
      "UserCreateDate": 1548089817.683,
      "UserLastModifiedDate": 1548089817.683,
      "Enabled": true,
    }
  ]
}
```

```
        "UserStatus": "FORCE_CHANGE_PASSWORD"
      }
    ]
  }
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUsersInGroup](#)中的。

## list-users

以下代码示例演示了如何使用 `list-users`。

### AWS CLI

列出用户

此示例最多列出 20 个用户。

命令:

```
aws cognito-idp list-users --user-pool-id us-west-2_aaaaaaaaa --limit 20
```

输出:

```
{
  "Users": [
    {
      "Username": "22704aa3-fc10-479a-97eb-2af5806bd327",
      "Enabled": true,
      "UserStatus": "FORCE_CHANGE_PASSWORD",
      "UserCreateDate": 1548089817.683,
      "UserLastModifiedDate": 1548089817.683,
      "Attributes": [
        {
          "Name": "sub",
          "Value": "22704aa3-fc10-479a-97eb-2af5806bd327"
        },
        {
          "Name": "email_verified",
          "Value": "true"
        },
        {
          "Name": "email",
```



```
    "Value": "mary@example.com"
  }
]
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUsers](#)中的。

## resend-confirmation-code

以下代码示例演示了如何使用 resend-confirmation-code。

### AWS CLI

#### 重新发送确认码

以下 resend-confirmation-code 示例向用户 jane 发送确认码。

```
aws cognito-idp resend-confirmation-code \
  --client-id 12a3b456c7de890f11g123hijk \
  --username jane
```

输出：

```
{
  "CodeDeliveryDetails": {
    "Destination": "j***@e***.com",
    "DeliveryMedium": "EMAIL",
    "AttributeName": "email"
  }
}
```

有关更多信息，请参阅《Amazon Cognito 开发人员指南》中的[注册并确认用户账户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ResendConfirmationCode](#)中的。

## respond-to-auth-challenge

以下代码示例演示了如何使用 respond-to-auth-challenge。

## AWS CLI

### 响应身份验证质询

此示例响应使用 `initiate-auth` 发起的身份验证质询。这是对 `NEW_PASSWORD_REQUIRED` 质询的响应。它为用户 `jane@example.com` 设置了密码。

命令:

```
aws cognito-idp respond-to-auth-challenge --client-id 3n4b5urk1ft4f13mg5e62d9ado
--challenge-name NEW_PASSWORD_REQUIRED --challenge-responses
USERNAME=jane@example.com,NEW_PASSWORD="password" --session "SESSION_TOKEN"
```

输出:

```
{
  "ChallengeParameters": {},
  "AuthenticationResult": {
    "AccessToken": "ACCESS_TOKEN",
    "ExpiresIn": 3600,
    "TokenType": "Bearer",
    "RefreshToken": "REFRESH_TOKEN",
    "IdToken": "ID_TOKEN",
    "NewDeviceMetadata": {
      "DeviceKey": "us-west-2_fec070d2-fa88-424a-8ec8-b26d7198eb23",
      "DeviceGroupKey": "-wt2ha1Zd"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RespondToAuthChallenge](#)中的。

## set-risk-configuration

以下代码示例演示了如何使用 `set-risk-configuration`。

### AWS CLI

#### 设置风险配置

此示例为用户池设置风险配置。它将注册事件操作设置为 `NO_ACTION`。

**命令:**

```
aws cognito-idp set-risk-configuration --user-pool-id us-west-2_aaaaaaaaa --compromised-credentials-risk-configuration EventFilter=SIGN_UP,Actions={EventAction=NO_ACTION}
```

**输出:**

```
{
  "RiskConfiguration": {
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "CompromisedCredentialsRiskConfiguration": {
      "EventFilter": [
        "SIGN_UP"
      ],
      "Actions": {
        "EventAction": "NO_ACTION"
      }
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetRiskConfiguration](#)中的。

**set-ui-customization**

以下代码示例演示了如何使用 set-ui-customization。

**AWS CLI****设置用户界面自定义**

此示例自定义用户池的 CSS 设置。

**命令:**

```
aws cognito-idp set-ui-customization --user-pool-id us-west-2_aaaaaaaaa --css ".logo-customizable {\n\tmax-width: 60%;\n\tmax-height: 30%;\n}\n\n.banner-customizable {\n\tpadding: 25px 0px 25px 10px;\n\tbackground-color: lightgray;\n}\n\n.label-customizable {\n\tfont-weight: 300;\n}\n\n.textDescription-customizable {\n\tpadding-top: 10px;\n\tpadding-bottom: 10px;\n\tdisplay: block;\n\tfont-size: 16px;\n}\n\n.idpDescription-customizable {\n\tpadding-top: 10px;\n\tpadding-
```

```

bottom: 10px;\n\tdisplay: block;\n\tfont-size: 16px;\n}\n\n.legalText-customizable
{\n\tcolor: #747474;\n\tfont-size: 11px;\n}\n\n.submitButton-customizable
{\n\tfont-size: 14px;\n\tfont-weight: bold;\n\tmargin: 20px 0px 10px 0px;\n
\theight: 40px;\n\twidth: 100%;\n\tcolor: #fff;\n\tbackground-color: #337ab7;
\n}\n\n.submitButton-customizable:hover {\n\tcolor: #fff;\n\tbackground-color:
#286090;\n}\n\n.errorMessage-customizable {\n\tpadding: 5px;\n\tfont-size: 14px;
\n\twidth: 100%;\n\tbackground: #F5F5F5;\n\tborder: 2px solid #D64958;\n\tcolor:
#D64958;\n}\n\n.inputField-customizable {\n\twidth: 100%;\n\theight: 34px;\n\tcolor:
#555;\n\tbackground-color: #fff;\n\tborder: 1px solid #ccc;\n}\n\n.inputField-
customizable:focus {\n\tborder-color: #66afe9;\n\toutline: 0;\n}\n\n.idpButton-
customizable {\n\theight: 40px;\n\twidth: 100%;\n\ttext-align: center;\n\tmargin-
bottom: 15px;\n\tcolor: #fff;\n\tbackground-color: #5bc0de;\n\tborder-color:
#46b8da;\n}\n\n.idpButton-customizable:hover {\n\tcolor: #fff;\n\tbackground-color:
#31b0d5;\n}\n\n.socialButton-customizable {\n\theight: 40px;\n\ttext-align: left;
\n\twidth: 100%;\n\tmargin-bottom: 15px;\n}\n\n.redirect-customizable {\n\ttext-
align: center;\n}\n\n.passwordCheck-notValid-customizable {\n\tcolor: #DF3312;
\n}\n\n.passwordCheck-valid-customizable {\n\tcolor: #19BF00;\n}\n\n.background-
customizable {\n\tbackground-color: #faf;\n}\n\n"

```

输出：

```

{
  "UICustomization": {
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "ClientId": "ALL",
    "CSS": ".logo-customizable {\n\tmax-width: 60%;\n\tmax-height: 30%;
\n}\n\n.banner-customizable {\n\tpadding: 25px 0px 25px 10px;\n\tbackground-color:
lightgray;\n}\n\n.label-customizable {\n\tfont-weight: 300;\n}\n\n.textDescription-
customizable {\n\tpadding-top: 10px;\n\tpadding-bottom: 10px;\n\tdisplay: block;
\n\tfont-size: 16px;\n}\n\n.idpDescription-customizable {\n\tpadding-top: 10px;\n
\tpadding-bottom: 10px;\n\tdisplay: block;\n\tfont-size: 16px;\n}\n\n.legalText-
customizable {\n\tcolor: #747474;\n\tfont-size: 11px;\n}\n\n.submitButton-customizable
{\n\tfont-size: 14px;\n\tfont-weight: bold;\n\tmargin: 20px 0px 10px 0px;\n
\theight: 40px;\n\twidth: 100%;\n\tcolor: #fff;\n\tbackground-color: #337ab7;
\n}\n\n.submitButton-customizable:hover {\n\tcolor: #fff;\n\tbackground-color:
#286090;\n}\n\n.errorMessage-customizable {\n\tpadding: 5px;\n\tfont-size: 14px;
\n\twidth: 100%;\n\tbackground: #F5F5F5;\n\tborder: 2px solid #D64958;\n\tcolor:
#D64958;\n}\n\n.inputField-customizable {\n\twidth: 100%;\n\theight: 34px;\n\tcolor:
#555;\n\tbackground-color: #fff;\n\tborder: 1px solid #ccc;\n}\n\n.inputField-
customizable:focus {\n\tborder-color: #66afe9;\n\toutline: 0;\n}\n\n.idpButton-
customizable {\n\theight: 40px;\n\twidth: 100%;\n\ttext-align: center;\n\tmargin-
bottom: 15px;\n\tcolor: #fff;\n\tbackground-color: #5bc0de;\n\tborder-color:
#46b8da;\n}\n\n.idpButton-customizable:hover {\n\tcolor: #fff;\n\tbackground-color:

```

```
#31b0d5;\n}\n.socialButton-customizable {\n\theight: 40px;\n\ttext-align: left;\n\twidth: 100%;\n\tmargin-bottom: 15px;\n}\n.redirect-customizable {\n\ttext-align: center;\n}\n.passwordCheck-notValid-customizable {\n\tcolor: #DF3312;\n}\n.passwordCheck-valid-customizable {\n\tcolor: #19BF00;\n}\n.background-customizable {\n\tbackground-color: #faf;\n}\n",
    "CSSVersion": "20190129172214"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetUiCustomization](#)中的。

## set-user-mfa-preference

以下代码示例演示了如何使用 set-user-mfa-preference。

### AWS CLI

#### 设置用户 MFA 设置

以下set-user-mfa-preference示例修改了 MFA 交付选项。它将 MFA 传送媒体更改为短信。

```
aws cognito-idp set-user-mfa-preference \
  --access-token "eyJra12345EXAMPLE" \
  --software-token-mfa-settings Enabled=true,PreferredMfa=true \
  --sms-mfa-settings Enabled=false,PreferredMfa=false
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Cognito 开发者指南中的[向用户池添加 MFA](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetUserMfaPreference](#)中的。

## set-user-settings

以下代码示例演示了如何使用 set-user-settings。

### AWS CLI

#### 设置用户设置

此示例将 MFA 交付首选项设置为电子邮件。

**命令:**

```
aws cognito-idp set-user-settings --access-token ACCESS_TOKEN --mfa-options
DeliveryMedium=EMAIL
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetUserSettings](#)中的。

**sign-up**

以下代码示例演示了如何使用 sign-up。

## AWS CLI

## 注册用户

此示例注册 jane@example.com。

**命令:**

```
aws cognito-idp sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username jane@example.com --password PASSWORD --user-attributes
Name="email",Value="jane@example.com" Name="name",Value="Jane"
```

**输出:**

```
{
  "UserConfirmed": false,
  "UserSub": "e04d60a6-45dc-441c-a40b-e25a787d4862"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SignUp](#)中的。

**start-user-import-job**

以下代码示例演示了如何使用 start-user-import-job。

## AWS CLI

## 启动用户导入任务

此示例启动用户输入作业。

有关导入用户的更多信息，请参阅从 CSV 文件将用户导入用户池。

命令:

```
aws cognito-idp start-user-import-job --user-pool-id us-west-2_aaaaaaaaa --job-id
import-TZqNQvDRnW
```

输出 :

```
{
  "UserImportJob": {
    "JobName": "import-Test10",
    "JobId": "import-lmpxS0uIzH",
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "PreSignedUrl": "PRE_SIGNED_URL",
    "CreationDate": 1548278378.928,
    "StartDate": 1548278397.334,
    "Status": "Pending",
    "CloudWatchLogsRoleArn": "arn:aws:iam::111111111111:role/
CognitoCloudWatchLogsRole",
    "ImportedUsers": 0,
    "SkippedUsers": 0,
    "FailedUsers": 0
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartUserImportJob](#)中的。

## stop-user-import-job

以下代码示例演示了如何使用 stop-user-import-job。

AWS CLI

停止用户导入任务

此示例停止用户输入作业。

有关导入用户的更多信息，请参阅从 CSV 文件将用户导入用户池。

命令:

```
aws cognito-idp stop-user-import-job --user-pool-id us-west-2_aaaaaaaaaa --job-id
import-TZqNQvDRnW
```

输出：

```
{
  "UserImportJob": {
    "JobName": "import-Test5",
    "JobId": "import-Fx0kARISFL",
    "UserPoolId": "us-west-2_aaaaaaaaaa",
    "PreSignedUrl": "PRE_SIGNED_URL",
    "CreationDate": 1548278576.259,
    "StartDate": 1548278623.366,
    "CompletionDate": 1548278626.741,
    "Status": "Stopped",
    "CloudWatchLogsRoleArn": "arn:aws:iam::111111111111:role/
CognitoCloudWatchLogsRole",
    "ImportedUsers": 0,
    "SkippedUsers": 0,
    "FailedUsers": 0,
    "CompletionMessage": "The Import Job was stopped by the developer."
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopUserImportJob](#)中的。

## update-auth-event-feedback

以下代码示例演示了如何使用 update-auth-event-feedback。

### AWS CLI

#### 更新身份验证事件反馈

此示例更新了授权事件反馈。它将事件标记为“有效”。

命令：

```
aws cognito-idp update-auth-event-feedback --user-pool-id us-west-2_aaaaaaaaaa --
username diego@example.com --event-id EVENT_ID --feedback-token FEEDBACK_TOKEN --
feedback-value "Valid"
```



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAuthEventFeedback](#)中的。

## update-device-status

以下代码示例演示了如何使用 update-device-status。

### AWS CLI

#### 更新设备状态

此示例将设备的状态更新为“未记住”。

#### 命令:

```
aws cognito-idp update-device-status --access-token ACCESS_TOKEN --device-key
DEVICE_KEY --device-remembered-status "not_remembered"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDeviceStatus](#)中的。

## update-group

以下代码示例演示了如何使用 update-group。

### AWS CLI

#### 更新群组

此示例更新了的描述和优先级 MyGroup。

#### 命令:

```
aws cognito-idp update-group --user-pool-id us-west-2_aaaaaaaaaa --group-name MyGroup
--description "New description" --precedence 2
```

#### 输出:

```
{
  "Group": {
    "GroupName": "MyGroup",
    "UserPoolId": "us-west-2_aaaaaaaaaa",
    "Description": "New description",
```

```
    "RoleArn": "arn:aws:iam::111111111111:role/MyRole",
    "Precedence": 2,
    "LastModifiedDate": 1548800862.812,
    "CreationDate": 1548097827.125
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateGroup](#)中的。

## update-resource-server

以下代码示例演示了如何使用 update-resource-server。

### AWS CLI

#### 更新资源服务器

此示例更新资源服务器 Weather。它添加了一个新的作用域。

命令:

```
aws cognito-idp update-resource-server --user-pool-id us-west-2_aaaaaaaaaa
--identifier weather.example.com --name Weather --scopes
ScopeName=NewScope,ScopeDescription="New scope description"
```

输出:

```
{
  "ResourceServer": {
    "UserPoolId": "us-west-2_aaaaaaaaaa",
    "Identifier": "weather.example.com",
    "Name": "Happy",
    "Scopes": [
      {
        "ScopeName": "NewScope",
        "ScopeDescription": "New scope description"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateResourceServer](#)中的。

## update-user-attributes

以下代码示例演示了如何使用 `update-user-attributes`。

### AWS CLI

#### 更新用户属性

此示例更新了用户属性“昵称”。

命令:

```
aws cognito-idp update-user-attributes --access-token ACCESS_TOKEN --user-attributes
Name="nickname",Value="Dan"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUserAttributes](#)中的。

## update-user-pool-client

以下代码示例演示了如何使用 `update-user-pool-client`。

### AWS CLI

#### 更新用户池客户端

此示例更新了用户池客户端的名称。它还添加了一个可写的属性“昵称”。

命令:

```
aws cognito-idp update-user-pool-client --user-pool-id us-west-2_aaaaaaaaa --client-
id 3n4b5urk1ft4fl3mg5e62d9ado --client-name "NewClientName" --write-attributes
"nickname"
```

输出:

```
{
  "UserPoolClient": {
    "UserPoolId": "us-west-2_aaaaaaaaa",
    "ClientName": "NewClientName",
    "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
    "LastModifiedDate": 1548802761.334,
    "CreationDate": 1548178931.258,
    "RefreshTokenValidity": 30,
  }
}
```

```
    "WriteAttributes": [  
        "nickname"  
    ],  
    "AllowedOAuthFlowsUserPoolClient": false  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUserPoolClient](#)中的。

## update-user-pool

以下代码示例演示了如何使用 update-user-pool。

### AWS CLI

#### 更新用户池

此示例向用户池添加标签。

命令:

```
aws cognito-idp update-user-pool --user-pool-id us-west-2_aaaaaaaaa --user-pool-tags  
Team=Blue,Area=West
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUserPool](#)中的。

## 使用 Amazon Comprehend 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon Comprehend 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### batch-detect-dominant-language

以下代码示例演示了如何使用 batch-detect-dominant-language。

#### AWS CLI

##### 检测多个输入文本的主导语言

以下batch-detect-dominant-language示例分析了多个输入文本并返回每个文本的主导语言。还会为每个预测输出预训练模型的置信度分数。

```
aws comprehend batch-detect-dominant-language \  
  --text-list "Physics is the natural science that involves the study of matter  
  and its motion and behavior through space and time, along with related concepts  
  such as energy and force."
```

##### 输出：

```
{  
  "ResultList": [  
    {  
      "Index": 0,  
      "Languages": [  
        {  
          "LanguageCode": "en",  
          "Score": 0.9986501932144165  
        }  
      ]  
    }  
  ],  
  "ErrorList": []  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[主要语言](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchDetectDominantLanguage](#)中的。

### batch-detect-entities

以下代码示例演示了如何使用 batch-detect-entities。

## AWS CLI

### 检测来自多个输入文本的实体

以下batch-detect-entities示例分析多个输入文本并返回每个文本的命名实体。预训练模型的置信度分数也是每个预测的输出。

```
aws comprehend batch-detect-entities \  
  --language-code en \  
  --text-list "Dear Jane, Your AnyCompany Financial Services LLC credit card  
account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due by July  
31st." "Please send customer feedback to Sunshine Spa, 123 Main St, Anywhere or to  
Alice at AnySpa@example.com."
```

输出：

```
{  
  "ResultList": [  
    {  
      "Index": 0,  
      "Entities": [  
        {  
          "Score": 0.9985517859458923,  
          "Type": "PERSON",  
          "Text": "Jane",  
          "BeginOffset": 5,  
          "EndOffset": 9  
        },  
        {  
          "Score": 0.9767839312553406,  
          "Type": "ORGANIZATION",  
          "Text": "AnyCompany Financial Services, LLC",  
          "BeginOffset": 16,  
          "EndOffset": 50  
        },  
        {  
          "Score": 0.9856694936752319,  
          "Type": "OTHER",  
          "Text": "1111-XXXX-1111-XXXX",  
          "BeginOffset": 71,  
          "EndOffset": 90  
        },  
        {
```

```
        "Score": 0.9652159810066223,
        "Type": "QUANTITY",
        "Text": ".53",
        "BeginOffset": 116,
        "EndOffset": 119
      },
      {
        "Score": 0.9986667037010193,
        "Type": "DATE",
        "Text": "July 31st",
        "BeginOffset": 135,
        "EndOffset": 144
      }
    ]
  },
  {
    "Index": 1,
    "Entities": [
      {
        "Score": 0.720084547996521,
        "Type": "ORGANIZATION",
        "Text": "Sunshine Spa",
        "BeginOffset": 33,
        "EndOffset": 45
      },
      {
        "Score": 0.9865870475769043,
        "Type": "LOCATION",
        "Text": "123 Main St",
        "BeginOffset": 47,
        "EndOffset": 58
      },
      {
        "Score": 0.5895616412162781,
        "Type": "LOCATION",
        "Text": "Anywhere",
        "BeginOffset": 60,
        "EndOffset": 68
      },
      {
        "Score": 0.6809214353561401,
        "Type": "PERSON",
        "Text": "Alice",
        "BeginOffset": 75,
```

```

        "EndOffset": 80
      },
      {
        "Score": 0.9979087114334106,
        "Type": "OTHER",
        "Text": "AnySpa@example.com",
        "BeginOffset": 84,
        "EndOffset": 99
      }
    ]
  },
  "ErrorList": []
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[实体](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchDetectEntities](#) 中的。

## batch-detect-key-phrases

以下代码示例演示了如何使用 batch-detect-key-phrases。

### AWS CLI

检测多个文本输入的关键短语

以下 batch-detect-key-phrases 示例分析了多个输入文本，并返回每个文本的关键名词短语。还会输出预训练模型对每个预测的置信度分数。

```

aws comprehend batch-detect-key-phrases \
  --language-code en \
  --text-list "Hello Zhang Wei, I am John, writing to you about the trip for next
Saturday." "Dear Jane, Your AnyCompany Financial Services LLC credit card account
1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due by July 31st."
"Please send customer feedback to Sunshine Spa, 123 Main St, Anywhere or to Alice
at AnySpa@example.com."

```

输出：

```

{
  "ResultList": [
    {

```



```
    "Index": 0,
    "KeyPhrases": [
      {
        "Score": 0.99700927734375,
        "Text": "Zhang Wei",
        "BeginOffset": 6,
        "EndOffset": 15
      },
      {
        "Score": 0.9929308891296387,
        "Text": "John",
        "BeginOffset": 22,
        "EndOffset": 26
      },
      {
        "Score": 0.9997230172157288,
        "Text": "the trip",
        "BeginOffset": 49,
        "EndOffset": 57
      },
      {
        "Score": 0.9999470114707947,
        "Text": "next Saturday",
        "BeginOffset": 62,
        "EndOffset": 75
      }
    ]
  },
  {
    "Index": 1,
    "KeyPhrases": [
      {
        "Score": 0.8358274102210999,
        "Text": "Dear Jane",
        "BeginOffset": 0,
        "EndOffset": 9
      },
      {
        "Score": 0.989359974861145,
        "Text": "Your AnyCompany Financial Services",
        "BeginOffset": 11,
        "EndOffset": 45
      }
    ]
  }
}
```

```
        "Score": 0.8812323808670044,  
        "Text": "LLC credit card account 1111-XXXX-1111-XXXX",  
        "BeginOffset": 47,  
        "EndOffset": 90  
    },  
    {  
        "Score": 0.9999381899833679,  
        "Text": "a minimum payment",  
        "BeginOffset": 95,  
        "EndOffset": 112  
    },  
    {  
        "Score": 0.9997439980506897,  
        "Text": ".53",  
        "BeginOffset": 116,  
        "EndOffset": 119  
    },  
    {  
        "Score": 0.996875524520874,  
        "Text": "July 31st",  
        "BeginOffset": 135,  
        "EndOffset": 144  
    }  
    ]  
},  
{  
    "Index": 2,  
    "KeyPhrases": [  
        {  
            "Score": 0.9990295767784119,  
            "Text": "customer feedback",  
            "BeginOffset": 12,  
            "EndOffset": 29  
        },  
        {  
            "Score": 0.9994127750396729,  
            "Text": "Sunshine Spa",  
            "BeginOffset": 33,  
            "EndOffset": 45  
        },  
        {  
            "Score": 0.9892991185188293,  
            "Text": "123 Main St",  
            "BeginOffset": 47,
```

```

        "EndOffset": 58
      },
      {
        "Score": 0.9969810843467712,
        "Text": "Alice",
        "BeginOffset": 75,
        "EndOffset": 80
      },
      {
        "Score": 0.9703696370124817,
        "Text": "AnySpa@example.com",
        "BeginOffset": 84,
        "EndOffset": 99
      }
    ]
  ],
  "ErrorList": []
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[关键词](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchDetectKeyPhrases](#)中的。

## batch-detect-sentiment

以下代码示例演示了如何使用 batch-detect-sentiment。

### AWS CLI

检测多个输入文本的流行情绪

以下batch-detect-sentiment示例分析了多个输入文本并返回主流情绪（ POSITIVE每个文本的NEGATIVE、、、或）。NEUTRAL MIXED

```

aws comprehend batch-detect-sentiment \
  --text-list "That movie was very boring, I can't believe it was over four hours
long." "It is a beautiful day for hiking today." "My meal was okay, I'm excited to
try other restaurants." \
  --language-code en

```

输出：

```
{
  "ResultList": [
    {
      "Index": 0,
      "Sentiment": "NEGATIVE",
      "SentimentScore": {
        "Positive": 0.00011316669406369328,
        "Negative": 0.9995445609092712,
        "Neutral": 0.00014722718333359808,
        "Mixed": 0.00019498742767609656
      }
    },
    {
      "Index": 1,
      "Sentiment": "POSITIVE",
      "SentimentScore": {
        "Positive": 0.9981263279914856,
        "Negative": 0.00015240783977787942,
        "Neutral": 0.0013876151060685515,
        "Mixed": 0.00033366199932061136
      }
    },
    {
      "Index": 2,
      "Sentiment": "MIXED",
      "SentimentScore": {
        "Positive": 0.15930435061454773,
        "Negative": 0.11471917480230331,
        "Neutral": 0.26897063851356506,
        "Mixed": 0.45700588822364807
      }
    }
  ],
  "ErrorList": []
}
```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的[情绪](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchDetectSentiment](#)中的。

## batch-detect-syntax

以下代码示例演示了如何使用 batch-detect-syntax。

## AWS CLI

### 检查多个输入文本中单词的语法和语音部分

以下batch-detect-syntax示例分析了多个输入文本的语法并返回语音的不同部分。预训练模型的置信度分数也是每个预测的输出。

```
aws comprehend batch-detect-syntax \  
  --text-list "It is a beautiful day." "Can you please pass the salt?" "Please pay  
the bill before the 31st." \  
  --language-code en
```

输出：

```
{  
  "ResultList": [  
    {  
      "Index": 0,  
      "SyntaxTokens": [  
        {  
          "TokenId": 1,  
          "Text": "It",  
          "BeginOffset": 0,  
          "EndOffset": 2,  
          "PartOfSpeech": {  
            "Tag": "PRON",  
            "Score": 0.9999740719795227  
          }  
        },  
        {  
          "TokenId": 2,  
          "Text": "is",  
          "BeginOffset": 3,  
          "EndOffset": 5,  
          "PartOfSpeech": {  
            "Tag": "VERB",  
            "Score": 0.999937117099762  
          }  
        },  
        {  
          "TokenId": 3,  
          "Text": "a",  
          "BeginOffset": 6,
```

```
        "EndOffset": 7,
        "PartOfSpeech": {
            "Tag": "DET",
            "Score": 0.9999926686286926
        }
    },
    {
        "TokenId": 4,
        "Text": "beautiful",
        "BeginOffset": 8,
        "EndOffset": 17,
        "PartOfSpeech": {
            "Tag": "ADJ",
            "Score": 0.9987891912460327
        }
    },
    {
        "TokenId": 5,
        "Text": "day",
        "BeginOffset": 18,
        "EndOffset": 21,
        "PartOfSpeech": {
            "Tag": "NOUN",
            "Score": 0.9999778866767883
        }
    },
    {
        "TokenId": 6,
        "Text": ".",
        "BeginOffset": 21,
        "EndOffset": 22,
        "PartOfSpeech": {
            "Tag": "PUNCT",
            "Score": 0.9999974966049194
        }
    }
]
},
{
    "Index": 1,
    "SyntaxTokens": [
        {
            "TokenId": 1,
            "Text": "Can",
```

```
        "BeginOffset": 0,
        "EndOffset": 3,
        "PartOfSpeech": {
            "Tag": "AUX",
            "Score": 0.9999770522117615
        }
    },
    {
        "TokenId": 2,
        "Text": "you",
        "BeginOffset": 4,
        "EndOffset": 7,
        "PartOfSpeech": {
            "Tag": "PRON",
            "Score": 0.9999986886978149
        }
    },
    {
        "TokenId": 3,
        "Text": "please",
        "BeginOffset": 8,
        "EndOffset": 14,
        "PartOfSpeech": {
            "Tag": "INTJ",
            "Score": 0.9681622385978699
        }
    },
    {
        "TokenId": 4,
        "Text": "pass",
        "BeginOffset": 15,
        "EndOffset": 19,
        "PartOfSpeech": {
            "Tag": "VERB",
            "Score": 0.9999874830245972
        }
    },
    {
        "TokenId": 5,
        "Text": "the",
        "BeginOffset": 20,
        "EndOffset": 23,
        "PartOfSpeech": {
            "Tag": "DET",
```

```
        "Score": 0.9999827146530151
      }
    },
    {
      "TokenId": 6,
      "Text": "salt",
      "BeginOffset": 24,
      "EndOffset": 28,
      "PartOfSpeech": {
        "Tag": "NOUN",
        "Score": 0.9995040893554688
      }
    },
    {
      "TokenId": 7,
      "Text": "?",
      "BeginOffset": 28,
      "EndOffset": 29,
      "PartOfSpeech": {
        "Tag": "PUNCT",
        "Score": 0.999998152256012
      }
    }
  ]
},
{
  "Index": 2,
  "SyntaxTokens": [
    {
      "TokenId": 1,
      "Text": "Please",
      "BeginOffset": 0,
      "EndOffset": 6,
      "PartOfSpeech": {
        "Tag": "INTJ",
        "Score": 0.9997857809066772
      }
    },
    {
      "TokenId": 2,
      "Text": "pay",
      "BeginOffset": 7,
      "EndOffset": 10,
      "PartOfSpeech": {
```



```
        "Tag": "VERB",
        "Score": 0.9999252557754517
    }
},
{
    "TokenId": 3,
    "Text": "the",
    "BeginOffset": 11,
    "EndOffset": 14,
    "PartOfSpeech": {
        "Tag": "DET",
        "Score": 0.9999842643737793
    }
},
{
    "TokenId": 4,
    "Text": "bill",
    "BeginOffset": 15,
    "EndOffset": 19,
    "PartOfSpeech": {
        "Tag": "NOUN",
        "Score": 0.9999588131904602
    }
},
{
    "TokenId": 5,
    "Text": "before",
    "BeginOffset": 20,
    "EndOffset": 26,
    "PartOfSpeech": {
        "Tag": "ADP",
        "Score": 0.9958304762840271
    }
},
{
    "TokenId": 6,
    "Text": "the",
    "BeginOffset": 27,
    "EndOffset": 30,
    "PartOfSpeech": {
        "Tag": "DET",
        "Score": 0.9999947547912598
    }
},
},
```

```
    {
      "TokenId": 7,
      "Text": "31st",
      "BeginOffset": 31,
      "EndOffset": 35,
      "PartOfSpeech": {
        "Tag": "NOUN",
        "Score": 0.9924124479293823
      }
    },
    {
      "TokenId": 8,
      "Text": ".",
      "BeginOffset": 35,
      "EndOffset": 36,
      "PartOfSpeech": {
        "Tag": "PUNCT",
        "Score": 0.9999955892562866
      }
    }
  ]
},
"ErrorList": []
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[语法分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchDetectSyntax](#)中的。

## batch-detect-targeted-sentiment

以下代码示例演示了如何使用 batch-detect-targeted-sentiment。

### AWS CLI

检测多个输入文本的情绪和每个命名实体

以下batch-detect-targeted-sentiment示例分析了多个输入文本，并返回命名实体以及附加到每个实体的流行情绪。预训练模型的置信度分数也是每个预测的输出。

```
aws comprehend batch-detect-targeted-sentiment \
  --language-code en \
```

```
--text-list "That movie was really boring, the original was way more
entertaining" "The trail is extra beautiful today." "My meal was just okay."
```

输出：

```
{
  "ResultList": [
    {
      "Index": 0,
      "Entities": [
        {
          "DescriptiveMentionIndex": [
            0
          ],
          "Mentions": [
            {
              "Score": 0.9999009966850281,
              "GroupScore": 1.0,
              "Text": "movie",
              "Type": "MOVIE",
              "MentionSentiment": {
                "Sentiment": "NEGATIVE",
                "SentimentScore": {
                  "Positive": 0.13887299597263336,
                  "Negative": 0.8057460188865662,
                  "Neutral": 0.05525200068950653,
                  "Mixed": 0.00012799999967683107
                }
              }
            },
            {
              "BeginOffset": 5,
              "EndOffset": 10
            }
          ]
        }
      ],
      {
        "DescriptiveMentionIndex": [
          0
        ],
        "Mentions": [
          {
            "Score": 0.9921110272407532,
            "GroupScore": 1.0,
            "Text": "original",
```

```

        "Type": "MOVIE",
        "MentionSentiment": {
            "Sentiment": "POSITIVE",
            "SentimentScore": {
                "Positive": 0.9999989867210388,
                "Negative": 9.999999974752427e-07,
                "Neutral": 0.0,
                "Mixed": 0.0
            }
        },
        "BeginOffset": 34,
        "EndOffset": 42
    }
]
}
]
},
{
    "Index": 1,
    "Entities": [
        {
            "DescriptiveMentionIndex": [
                0
            ],
            "Mentions": [
                {
                    "Score": 0.7545599937438965,
                    "GroupScore": 1.0,
                    "Text": "trail",
                    "Type": "OTHER",
                    "MentionSentiment": {
                        "Sentiment": "POSITIVE",
                        "SentimentScore": {
                            "Positive": 1.0,
                            "Negative": 0.0,
                            "Neutral": 0.0,
                            "Mixed": 0.0
                        }
                    },
                    "BeginOffset": 4,
                    "EndOffset": 9
                }
            ]
        }
    ]
},

```

```
{
  "DescriptiveMentionIndex": [
    0
  ],
  "Mentions": [
    {
      "Score": 0.9999960064888,
      "GroupScore": 1.0,
      "Text": "today",
      "Type": "DATE",
      "MentionSentiment": {
        "Sentiment": "NEUTRAL",
        "SentimentScore": {
          "Positive": 9.000000318337698e-06,
          "Negative": 1.999999949504854e-06,
          "Neutral": 0.9999859929084778,
          "Mixed": 3.99999989900971e-06
        }
      }
    },
    {
      "BeginOffset": 29,
      "EndOffset": 34
    }
  ]
},
{
  "Index": 2,
  "Entities": [
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "Score": 0.9999880194664001,
          "GroupScore": 1.0,
          "Text": "My",
          "Type": "PERSON",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Positive": 0.0,
              "Negative": 0.0,
            }
          }
        }
      ]
    }
  ]
}
```

```

        "Neutral": 1.0,
        "Mixed": 0.0
      }
    },
    "BeginOffset": 0,
    "EndOffset": 2
  }
]
},
{
  "DescriptiveMentionIndex": [
    0
  ],
  "Mentions": [
    {
      "Score": 0.9995260238647461,
      "GroupScore": 1.0,
      "Text": "meal",
      "Type": "OTHER",
      "MentionSentiment": {
        "Sentiment": "NEUTRAL",
        "SentimentScore": {
          "Positive": 0.04695599898695946,
          "Negative": 0.003226999891921878,
          "Neutral": 0.6091709733009338,
          "Mixed": 0.34064599871635437
        }
      }
    },
    {
      "BeginOffset": 3,
      "EndOffset": 7
    }
  ]
}
]
}
],
"ErrorList": []
}

```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的[定向情绪](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[BatchDetectTargetedSentiment](#)中的。

## classify-document

以下代码示例演示了如何使用 `classify-document`。

### AWS CLI

使用特定于模型的端点对文档进行分类

以下 `classify-document` 示例对带有自定义模型端点的文档进行分类。本示例中的模型是在一个数据集上训练的，该数据集包含标为垃圾邮件或非垃圾邮件或“ham”的短信。

```
aws comprehend classify-document \
  --endpoint-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier-
  endpoint/example-classifier-endpoint \
  --text "CONGRATULATIONS! TXT 1235550100 to win $5000"
```

输出：

```
{
  "Classes": [
    {
      "Name": "spam",
      "Score": 0.9998599290847778
    },
    {
      "Name": "ham",
      "Score": 0.00014001205272506922
    }
  ]
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[自定义分类](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ClassifyDocument](#) 中的。

## contains-pii-entities

以下代码示例演示了如何使用 `contains-pii-entities`。

### AWS CLI

分析输入文本中是否存在 PII 信息

以下contains-pii-entities示例分析输入文本中是否存在个人信息 (PII)，并返回已识别的 PII 实体类型的标签，例如姓名、地址、银行账号或电话号码。

```
aws comprehend contains-pii-entities \  
  --language-code en \  
  --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC  
credit card  
  account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due by  
July 31st. Based on your autopay settings,  
  we will withdraw your payment on the due date from your bank account number  
XXXXXX1111 with the routing number XXXXX0000.  
  Customer feedback for Sunshine Spa, 100 Main St, Anywhere. Send comments to  
Alice at AnySpa@example.com."
```

输出：

```
{  
  "Labels": [  
    {  
      "Name": "NAME",  
      "Score": 1.0  
    },  
    {  
      "Name": "EMAIL",  
      "Score": 1.0  
    },  
    {  
      "Name": "BANK_ACCOUNT_NUMBER",  
      "Score": 0.9995794296264648  
    },  
    {  
      "Name": "BANK_ROUTING",  
      "Score": 0.9173126816749573  
    },  
    {  
      "Name": "CREDIT_DEBIT_NUMBER",  
      "Score": 1.0  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[个人信息 \(PII\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ContainsPiiEntities](#)中的。



## create-dataset

以下代码示例演示了如何使用 create-dataset。

### AWS CLI

#### 创建飞轮数据集

以下create-dataset示例为飞轮创建数据集。该数据集将用作--dataset-type标签指定的其他训练数据。

```
aws comprehend create-dataset \  
  --flywheel-arn arn:aws:comprehend:us-west-2:111122223333:flywheel/flywheel-  
entity \  
  --dataset-name example-dataset \  
  --dataset-type "TRAIN" \  
  --input-data-config file://inputConfig.json
```

file://inputConfig.json 的内容：

```
{  
  "DataFormat": "COMPREHEND_CSV",  
  "DocumentClassifierInputDataConfig": {  
    "S3Uri": "s3://DOC-EXAMPLE-BUCKET/training-data.csv"  
  }  
}
```

输出：

```
{  
  "DatasetArn": "arn:aws:comprehend:us-west-2:111122223333:flywheel/flywheel-  
entity/dataset/example-dataset"  
}
```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的[飞轮概述](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDataset](#)中的。

## create-document-classifier

以下代码示例演示了如何使用 create-document-classifier。

## AWS CLI

### 创建文档分类器对文档进行分类

以下 `create-document-classifier` 示例启动文档分类器模型的训练过程。训练数据文件 `training.csv` 位于 `--input-data-config` 标签处。`training.csv` 是一个两列文档，其中第一列提供标签或分类，第二列提供文档。

```
aws comprehend create-document-classifier \  
  --document-classifier-name example-classifier \  
  --data-access-arn arn:aws:comprehend:us-west-2:111122223333:pii-entities-  
detection-job/123456abcdeb0e11022f22a11EXAMPLE \  
  --input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET/" \  
  --language-code en
```

输出：

```
{  
  "DocumentClassifierArn": "arn:aws:comprehend:us-west-2:111122223333:document-  
classifier/example-classifier"  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[自定义分类](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDocumentClassifier](#)中的。

## create-endpoint

以下代码示例演示了如何使用 `create-endpoint`。

### AWS CLI

#### 为自定义模型创建端点

以下`create-endpoint`示例为之前训练的自定义模型创建了一个用于同步推理的端点。

```
aws comprehend create-endpoint \  
  --endpoint-name example-classifier-endpoint-1 \  
  --model-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/  
example-classifier \  
  --desired-inference-units 1
```

输出：

```
{
  "EndpointArn": "arn:aws:comprehend:us-west-2:111122223333:document-classifier-
endpoint/example-classifier-endpoint-1"
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[管理 Amazon Comprehend 端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateEndpoint](#)中的。

## create-entity-recognizer

以下代码示例演示了如何使用 create-entity-recognizer。

### AWS CLI

#### 创建自定义实体识别器

以下 create-entity-recognizer 示例开始自定义实体识别器模型的训练过程。此示例使用包含训练文档的 CSV 文件和 CSV 实体列表 entity\_list.csv 来训练模型。raw\_text.csv entity-list.csv 包含以下各列：文本和类型。

```
aws comprehend create-entity-recognizer \
  --recognizer-name example-entity-recognizer
  --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role \
  --input-data-config "EntityTypes=[{Type=DEVICE}],Documents={S3Uri=s3://DOC-
EXAMPLE-BUCKET/trainingdata/raw_text.csv},EntityList={S3Uri=s3://DOC-EXAMPLE-BUCKET/
trainingdata/entity_list.csv}"
  --language-code en
```

输出：

```
{
  "EntityRecognizerArn": "arn:aws:comprehend:us-west-2:111122223333:example-
entity-recognizer/entityrecognizer1"
}
```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的[自定义实体识别](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateEntityRecognizer](#)中的。

## create-flywheel

以下代码示例演示了如何使用 create-flywheel。

### AWS CLI

#### 创建飞轮

以下create-flywheel示例创建了一个飞轮来协调文档分类或实体识别模型的持续训练。本示例中的飞轮是为了管理--active-model-arn标签指定的现有训练模型。创建飞轮时，会在--input-data-lake标签处创建一个数据湖。

```
aws comprehend create-flywheel \  
  --flywheel-name example-flywheel \  
  --active-model-arn arn:aws:comprehend:us-west-2:111122223333:document-  
classifier/example-model/version/1 \  
  --data-access-role-arn arn:aws:iam::111122223333:role/service-role/  
AmazonComprehendServiceRole-example-role \  
  --data-lake-s3-uri "s3://DOC-EXAMPLE-BUCKET"
```

输出：

```
{  
  "FlywheelArn": "arn:aws:comprehend:us-west-2:111122223333:flywheel/example-  
flywheel"  
}
```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的[飞轮概述](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFlywheel](#)中的。

## delete-document-classifier

以下代码示例演示了如何使用 delete-document-classifier。

### AWS CLI

#### 删除自定义文档分类器

以下 `delete-document-classifier` 示例删除了自定义文档分类器模型。

```
aws comprehend delete-document-classifier \  
  --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-  
classifier/example-classifier-1
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[管理 Amazon Comprehend 端点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDocumentClassifier](#)中的。

## **delete-endpoint**

以下代码示例演示了如何使用 `delete-endpoint`。

### AWS CLI

删除自定义模型的终端节点

以下`delete-endpoint`示例删除了特定于模型的端点。必须删除所有端点才能删除模型。

```
aws comprehend delete-endpoint \  
  --endpoint-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier-  
endpoint/example-classifier-endpoint-1
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[管理 Amazon Comprehend 端点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteEndpoint](#)中的。

## **delete-entity-recognizer**

以下代码示例演示了如何使用 `delete-entity-recognizer`。

### AWS CLI

删除自定义实体识别器模型

以下delete-entity-recognizer示例删除自定义实体识别器模型。

```
aws comprehend delete-entity-recognizer \  
  --entity-recognizer-arn arn:aws:comprehend:us-west-2:111122223333:entity-  
recognizer/example-entity-recognizer-1
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[管理 Amazon Comprehend 端点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteEntityRecognizer](#)中的。

## delete-flywheel

以下代码示例演示了如何使用 delete-flywheel。

### AWS CLI

删除飞轮

以下delete-flywheel示例删除了飞轮。数据湖或与飞轮关联的模型不会被删除。

```
aws comprehend delete-flywheel \  
  --flywheel-arn arn:aws:comprehend:us-west-2:111122223333:flywheel/example-  
flywheel-1
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊 Comprehend 开发者指南》中的 Flywheel [概述](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFlywheel](#)中的。

## delete-resource-policy

以下代码示例演示了如何使用 delete-resource-policy。

### AWS CLI

删除基于资源的策略

以下delete-resource-policy示例从 Amazon Comprehend 资源中删除基于资源的策略。

```
aws comprehend delete-resource-policy \  
  --resource-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/  
example-classifier-1/version/1
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的在[AWS 账户之间复制自定义模型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteResourcePolicy](#)中的。

## describe-dataset

以下代码示例演示了如何使用 describe-dataset。

### AWS CLI

#### 描述飞轮数据集

以下describe-dataset示例获取飞轮数据集的属性。

```
aws comprehend describe-dataset \  
  --dataset-arn arn:aws:comprehend:us-west-2:111122223333:flywheel/flywheel-  
entity/dataset/example-dataset
```

输出：

```
{  
  "DatasetProperties": {  
    "DatasetArn": "arn:aws:comprehend:us-west-2:111122223333:flywheel/flywheel-  
entity/dataset/example-dataset",  
    "DatasetName": "example-dataset",  
    "DatasetType": "TRAIN",  
    "DatasetS3Uri": "s3://DOC-EXAMPLE-BUCKET/flywheel-entity/  
schemaVersion=1/12345678A123456Z/datasets/example-dataset/20230616T203710Z/",  
    "Status": "CREATING",  
    "CreationTime": "2023-06-16T20:37:10.400000+00:00"  
  }  
}
```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的[飞轮概述](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDataset](#)中的。

## describe-document-classification-job

以下代码示例演示了如何使用 `describe-document-classification-job`。

### AWS CLI

#### 描述文档分类作业

以下 `describe-document-classification-job` 示例将获取异步文档分类作业的属性。

```
aws comprehend describe-document-classification-job \  
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{  
  "DocumentClassificationJobProperties": {  
    "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:document-  
classification-job/123456abcdeb0e11022f22a11EXAMPLE",  
    "JobName": "exampleclassificationjob",  
    "JobStatus": "COMPLETED",  
    "SubmitTime": "2023-06-14T17:09:51.788000+00:00",  
    "EndTime": "2023-06-14T17:15:58.582000+00:00",  
    "DocumentClassifierArn": "arn:aws:comprehend:us-  
west-2:111122223333:document-classifier/mymodel/version/1",  
    "InputDataConfig": {  
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/jobdata/",  
      "InputFormat": "ONE_DOC_PER_LINE"  
    },  
    "OutputDataConfig": {  
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/111122223333-  
CLN-123456abcdeb0e11022f22a11EXAMPLE/output/output.tar.gz"  
    },  
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/  
AmazonComprehendServiceRole-servicerole"  
  }  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[自定义分类](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDocumentClassificationJob](#)中的。



## describe-document-classifier

以下代码示例演示了如何使用 `describe-document-classifier`。

### AWS CLI

#### 描述文档分类器

以下 `describe-document-classifier` 示例将获取自定义文档分类器模型的属性。

```
aws comprehend describe-document-classifier \  
  --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-  
classifier/example-classifier-1
```

输出：

```
{  
  "DocumentClassifierProperties": {  
    "DocumentClassifierArn": "arn:aws:comprehend:us-  
west-2:111122223333:document-classifier/example-classifier-1",  
    "LanguageCode": "en",  
    "Status": "TRAINED",  
    "SubmitTime": "2023-06-13T19:04:15.735000+00:00",  
    "EndTime": "2023-06-13T19:42:31.752000+00:00",  
    "TrainingStartTime": "2023-06-13T19:08:20.114000+00:00",  
    "TrainingEndTime": "2023-06-13T19:41:35.080000+00:00",  
    "InputDataConfig": {  
      "DataFormat": "COMPREHEND_CSV",  
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/trainingdata"  
    },  
    "OutputDataConfig": {},  
    "ClassifierMetadata": {  
      "NumberOfLabels": 3,  
      "NumberOfTrainedDocuments": 5016,  
      "NumberOfTestDocuments": 557,  
      "EvaluationMetrics": {  
        "Accuracy": 0.9856,  
        "Precision": 0.9919,  
        "Recall": 0.9459,  
        "F1Score": 0.9673,  
        "MicroPrecision": 0.9856,  
        "MicroRecall": 0.9856,  
        "MicroF1Score": 0.9856,  
        "HammingLoss": 0.0144  
      }  
    }  
  }  
}
```

```

    }
  },
  "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role",
  "Mode": "MULTI_CLASS"
}
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[创建和管理自定义模型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDocumentClassifier](#)中的。

## describe-dominant-language-detection-job

以下代码示例演示了如何使用 describe-dominant-language-detection-job。

### AWS CLI

描述占主导地位的语言检测工作。

以下describe-dominant-language-detection-job示例获取异步主导语言检测作业的属性。

```
aws comprehend describe-dominant-language-detection-job \
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{
  "DominantLanguageDetectionJobProperties": {
    "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:dominant-language-
detection-job/123456abcdeb0e11022f22a11EXAMPLE",
    "JobName": "languageanalysis1",
    "JobStatus": "IN_PROGRESS",
    "SubmitTime": "2023-06-09T18:10:38.037000+00:00",
    "InputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET",
      "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/111122223333-
LANGUAGE-123456abcdeb0e11022f22a11EXAMPLE/output/output.tar.gz"
    }
  }
}
```

```
    },
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
  }
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDominantLanguageDetectionJob](#) 中的。

## describe-endpoint

以下代码示例演示了如何使用 describe-endpoint。

### AWS CLI

描述特定的终端节点

以下 describe-endpoint 示例获取特定于模型的端点的属性。

```
aws comprehend describe-endpoint \
  --endpoint-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier-
endpoint/example-classifier-endpoint
```

输出：

```
{
  "EndpointProperties": {
    "EndpointArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier-endpoint/example-classifier-endpoint",
    "Status": "IN_SERVICE",
    "ModelArn": "arn:aws:comprehend:us-west-2:111122223333:document-classifier/
exampleclassifier1",
    "DesiredModelArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/exampleclassifier1",
    "DesiredInferenceUnits": 1,
    "CurrentInferenceUnits": 1,
    "CreationTime": "2023-06-13T20:32:54.526000+00:00",
    "LastModifiedTime": "2023-06-13T20:32:54.526000+00:00"
  }
}
```

```
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[管理 Amazon Comprehend 端点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEndpoint](#)中的。

## describe-entities-detection-job

以下代码示例演示了如何使用 describe-entities-detection-job。

### AWS CLI

#### 描述实体检测作业

以下describe-entities-detection-job示例获取异步实体检测作业的属性。

```
aws comprehend describe-entities-detection-job \
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{
  "EntitiesDetectionJobProperties": {
    "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:entities-detection-
job/123456abcdeb0e11022f22a11EXAMPLE",
    "JobName": "example-entity-detector",
    "JobStatus": "COMPLETED",
    "SubmitTime": "2023-06-08T21:30:15.323000+00:00",
    "EndTime": "2023-06-08T21:40:23.509000+00:00",
    "InputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/AsyncBatchJobs/",
      "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/thefolder/111122223333-
NER-123456abcdeb0e11022f22a11EXAMPLE/output/output.tar.gz"
    },
    "LanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::12345678012:role/service-role/
AmazonComprehendServiceRole-example-role"
  }
}
```

```
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEntitiesDetectionJob](#)中的。

## describe-entity-recognizer

以下代码示例演示了如何使用 describe-entity-recognizer。

### AWS CLI

#### 描述实体识别器

以下describe-entity-recognizer示例获取自定义实体识别器模型的属性。

```
aws comprehend describe-entity-recognizer \
    entity-recognizer-arn arn:aws:comprehend:us-west-2:111122223333:entity-
    recognizer/business-recongizer-1/version/1
```

输出：

```
{
  "EntityRecognizerProperties": {
    "EntityRecognizerArn": "arn:aws:comprehend:us-west-2:111122223333:entity-
    recognizer/business-recongizer-1/version/1",
    "LanguageCode": "en",
    "Status": "TRAINED",
    "SubmitTime": "2023-06-14T20:44:59.631000+00:00",
    "EndTime": "2023-06-14T20:59:19.532000+00:00",
    "TrainingStartTime": "2023-06-14T20:48:52.811000+00:00",
    "TrainingEndTime": "2023-06-14T20:58:11.473000+00:00",
    "InputDataConfig": {
      "DataFormat": "COMPREHEND_CSV",
      "EntityTypes": [
        {
          "Type": "BUSINESS"
        }
      ],
      "Documents": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/trainingdata/dataset/",
        "InputFormat": "ONE_DOC_PER_LINE"
      }
    }
  }
}
```

```
    },
    "EntityList": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/trainingdata/entity.csv"
    }
  },
  "RecognizerMetadata": {
    "NumberOfTrainedDocuments": 1814,
    "NumberOfTestDocuments": 486,
    "EvaluationMetrics": {
      "Precision": 100.0,
      "Recall": 100.0,
      "F1Score": 100.0
    },
  },
  "EntityTypes": [
    {
      "Type": "BUSINESS",
      "EvaluationMetrics": {
        "Precision": 100.0,
        "Recall": 100.0,
        "F1Score": 100.0
      },
      "NumberOfTrainMentions": 1520
    }
  ],
  "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/AmazonComprehendServiceRole-example-role",
  "VersionName": "1"
}
}
```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的[自定义实体识别](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeEntityRecognizer](#)中的。

## describe-events-detection-job

以下代码示例演示了如何使用 describe-events-detection-job。

### AWS CLI

描述事件检测作业。

以下 describe-events-detection-job 示例获取异步事件检测作业的属性。

```
aws comprehend describe-events-detection-job \  
--job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{  
  "EventsDetectionJobProperties": {  
    "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:events-detection-  
job/123456abcdeb0e11022f22a11EXAMPLE",  
    "JobName": "events_job_1",  
    "JobStatus": "IN_PROGRESS",  
    "SubmitTime": "2023-06-12T18:45:56.054000+00:00",  
    "InputDataConfig": {  
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/EventsData",  
      "InputFormat": "ONE_DOC_PER_LINE"  
    },  
    "OutputDataConfig": {  
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/111122223333-  
EVENTS-123456abcdeb0e11022f22a11EXAMPLE/output/"  
    },  
    "LanguageCode": "en",  
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/  
AmazonComprehendServiceRole-example-role",  
    "TargetEventTypes": [  
      "BANKRUPTCY",  
      "EMPLOYMENT",  
      "CORPORATE_ACQUISITION",  
      "CORPORATE_MERGER",  
      "INVESTMENT_GENERAL"  
    ]  
  }  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEventsDetectionJob](#)中的。

## describe-flywheel-iteration

以下代码示例演示了如何使用 describe-flywheel-iteration。

## AWS CLI

### 描述飞轮迭代

以下describe-flywheel-iteration示例获取飞轮迭代的属性。

```
aws comprehend describe-flywheel-iteration \  
  --flywheel-arn arn:aws:comprehend:us-west-2:111122223333:flywheel/example-  
flywheel \  
  --flywheel-iteration-id 20232222AEXAMPLE
```

输出：

```
{  
  "FlywheelIterationProperties": {  
    "FlywheelArn": "arn:aws:comprehend:us-west-2:111122223333:flywheel/flywheel-  
entity",  
    "FlywheelIterationId": "20232222AEXAMPLE",  
    "CreationTime": "2023-06-16T21:10:26.385000+00:00",  
    "EndTime": "2023-06-16T23:33:16.827000+00:00",  
    "Status": "COMPLETED",  
    "Message": "FULL_ITERATION: Flywheel iteration performed all functions  
successfully.",  
    "EvaluatedModelArn": "arn:aws:comprehend:us-west-2:111122223333:document-  
classifier/example-classifier/version/1",  
    "EvaluatedModelMetrics": {  
      "AverageF1Score": 0.7742663922375772,  
      "AveragePrecision": 0.8287636394041166,  
      "AverageRecall": 0.7427084833645399,  
      "AverageAccuracy": 0.8795394154118689  
    },  
    "TrainedModelArn": "arn:aws:comprehend:us-west-2:111122223333:document-  
classifier/example-classifier/version/Comprehend-Generated-v1-bb52d585",  
    "TrainedModelMetrics": {  
      "AverageF1Score": 0.9767700253081214,  
      "AveragePrecision": 0.9767700253081214,  
      "AverageRecall": 0.9767700253081214,  
      "AverageAccuracy": 0.9858281665190434  
    },  
    "EvaluationManifestS3Prefix": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/flywheel-  
entity/schemaVersion=1/20230616T200543Z/evaluation/20230616T211026Z/"  
  }  
}
```



有关更多信息，请参阅《亚马逊 Comprehend 开发者指南》中的 [Flywheel 概述](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeFlywheelIteration](#) 中的。

## describe-flywheel

以下代码示例演示了如何使用 describe-flywheel。

### AWS CLI

#### 描述飞轮

以下 describe-flywheel 示例获取飞轮的属性。在此示例中，与飞轮关联的模型是一个自定义分类器模型，该模型经过训练，可以将文档分类为垃圾邮件或非垃圾邮件或“ham”。

```
aws comprehend describe-flywheel \
  --flywheel-arn arn:aws:comprehend:us-west-2:111122223333:flywheel/example-
  flywheel
```

输出：

```
{
  "FlywheelProperties": {
    "FlywheelArn": "arn:aws:comprehend:us-west-2:111122223333:flywheel/example-
  flywheel",
    "ActiveModelArn": "arn:aws:comprehend:us-west-2:111122223333:document-
  classifier/example-model/version/1",
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
  AmazonComprehendServiceRole-example-role",
    "TaskConfig": {
      "LanguageCode": "en",
      "DocumentClassificationConfig": {
        "Mode": "MULTI_CLASS",
        "Labels": [
          "ham",
          "spam"
        ]
      }
    },
    "DataLakeS3Uri": "s3://DOC-EXAMPLE-BUCKET/example-flywheel/
  schemaVersion=1/20230616T200543Z/",
    "DataSecurityConfig": {},
    "Status": "ACTIVE",
```

```
    "ModelType": "DOCUMENT_CLASSIFIER",
    "CreationTime": "2023-06-16T20:05:43.242000+00:00",
    "LastModifiedTime": "2023-06-16T20:21:43.567000+00:00"
  }
}
```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的[飞轮概述](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeFlywheel](#)中的。

## describe-key-phrases-detection-job

以下代码示例演示了如何使用 `describe-key-phrases-detection-job`。

### AWS CLI

描述一项关键短语检测工作

以下 `describe-key-phrases-detection-job` 示例获取异步关键短语检测作业的属性。

```
aws comprehend describe-key-phrases-detection-job \
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{
  "KeyPhrasesDetectionJobProperties": {
    "JobId": "69aa080c00fc68934a6a98f10EXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:key-phrases-detection-job/69aa080c00fc68934a6a98f10EXAMPLE",
    "JobName": "example-key-phrases-detection-job",
    "JobStatus": "COMPLETED",
    "SubmitTime": 1686606439.177,
    "EndTime": 1686606806.157,
    "InputDataConfig": {
      "S3Uri": "s3://dereksbucket1001/EventsData/",
      "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
      "S3Uri": "s3://dereksbucket1002/testfolder/111122223333-KP-69aa080c00fc68934a6a98f10EXAMPLE/output/output.tar.gz"
    },
    "LanguageCode": "en",
  }
}
```

```
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
    AmazonComprehendServiceRole-testrole"
  }
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeKeyPhrasesDetectionJob](#)中的。

## describe-pii-entities-detection-job

以下代码示例演示了如何使用 describe-pii-entities-detection-job。

### AWS CLI

描述 PII 实体检测作业

以下describe-pii-entities-detection-job示例获取异步 pii 实体检测作业的属性。

```
aws comprehend describe-pii-entities-detection-job \
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{
  "PiiEntitiesDetectionJobProperties": {
    "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:pii-entities-detection-
job/123456abcdeb0e11022f22a11EXAMPLE",
    "JobName": "example-pii-entities-job",
    "JobStatus": "IN_PROGRESS",
    "SubmitTime": "2023-06-08T21:30:15.323000+00:00",
    "EndTime": "2023-06-08T21:40:23.509000+00:00",
    "InputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/AsyncBatchJobs/",
      "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/thefolder/111122223333-
NER-123456abcdeb0e11022f22a11EXAMPLE/output/output.tar.gz"
    },
    "LanguageCode": "en",
```

```
"DataAccessRoleArn": "arn:aws:iam::12345678012:role/service-role/
AmazonComprehendServiceRole-example-role"
  }
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePiiEntitiesDetectionJob](#)中的。

## describe-resource-policy

以下代码示例演示了如何使用 describe-resource-policy。

### AWS CLI

描述附加到模型的资源策略

以下describe-resource-policy示例获取了附加到模型的基于资源的策略的属性。

```
aws comprehend describe-resource-policy \
  --resource-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/
example-classifier/version/1
```

输出：

```
{
  "ResourcePolicy": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":
  \"Allow\",\"Principal\":{\"AWS\":\"arn:aws:iam::444455556666:root\"},\"Action\":
  \"comprehend:ImportModel\",\"Resource\":\"*\"}]}",
  "CreationTime": "2023-06-19T18:44:26.028000+00:00",
  "LastModifiedTime": "2023-06-19T18:53:02.002000+00:00",
  "PolicyRevisionId": "baa675d069d07afaa2aa3106ae280f61"
}
```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的在[AWS 账户之间复制自定义模型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeResourcePolicy](#)中的。

## describe-sentiment-detection-job

以下代码示例演示了如何使用 describe-sentiment-detection-job。

## AWS CLI

### 描述情绪检测作业

以下describe-sentiment-detection-job示例获取异步情绪检测作业的属性。

```
aws comprehend describe-sentiment-detection-job \  
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{  
  "SentimentDetectionJobProperties": {  
    "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:sentiment-detection-  
job/123456abcdeb0e11022f22a11EXAMPLE",  
    "JobName": "movie_review_analysis",  
    "JobStatus": "IN_PROGRESS",  
    "SubmitTime": "2023-06-09T23:16:15.956000+00:00",  
    "InputDataConfig": {  
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/MovieData",  
      "InputFormat": "ONE_DOC_PER_LINE"  
    },  
    "OutputDataConfig": {  
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/111122223333-  
TS-123456abcdeb0e11022f22a11EXAMPLE/output/output.tar.gz"  
    },  
    "LanguageCode": "en",  
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/  
AmazonComprehendServiceRole-servicerole"  
  }  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSentimentDetectionJob](#)中的。

### describe-targeted-sentiment-detection-job

以下代码示例演示了如何使用 describe-targeted-sentiment-detection-job。

## AWS CLI

描述有针对性的情绪检测工作

以下describe-targeted-sentiment-detection-job示例获取异步定向情绪检测作业的属性。

```
aws comprehend describe-targeted-sentiment-detection-job \  
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{  
  "TargetedSentimentDetectionJobProperties": {  
    "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:targeted-sentiment-  
detection-job/123456abcdeb0e11022f22a11EXAMPLE",  
    "JobName": "movie_review_analysis",  
    "JobStatus": "IN_PROGRESS",  
    "SubmitTime": "2023-06-09T23:16:15.956000+00:00",  
    "InputDataConfig": {  
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/MovieData",  
      "InputFormat": "ONE_DOC_PER_LINE"  
    },  
    "OutputDataConfig": {  
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/111122223333-  
TS-123456abcdeb0e11022f22a11EXAMPLE/output/output.tar.gz"  
    },  
    "LanguageCode": "en",  
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/  
AmazonComprehendServiceRole-servicerole"  
  }  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTargetedSentimentDetectionJob](#)中的。

## describe-topics-detection-job

以下代码示例演示了如何使用 describe-topics-detection-job。

### AWS CLI

#### 描述主题检测作业

以下 describe-topics-detection-job 示例获取异步主题检测作业的属性。

```
aws comprehend describe-topics-detection-job \  
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{  
  "TopicsDetectionJobProperties": {  
    "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-detection-  
job/123456abcdeb0e11022f22a11EXAMPLE",  
    "JobName": "example_topics_detection",  
    "JobStatus": "IN_PROGRESS",  
    "SubmitTime": "2023-06-09T18:44:43.414000+00:00",  
    "InputDataConfig": {  
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET",  
      "InputFormat": "ONE_DOC_PER_LINE"  
    },  
    "OutputDataConfig": {  
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/111122223333-  
TOPICS-123456abcdeb0e11022f22a11EXAMPLE/output/output.tar.gz"  
    },  
    "NumberOfTopics": 10,  
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/  
AmazonComprehendServiceRole-examplerole"  
  }  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTopicsDetectionJob](#)中的。

## detect-dominant-language

以下代码示例演示了如何使用 detect-dominant-language。

### AWS CLI

检测输入文本的主要语言

以下 detect-dominant-language 分析输入文本并识别主要语言。预训练模型的置信度分数也是输出。

```
aws comprehend detect-dominant-language \  
  --text "It is a beautiful day in Seattle."
```

输出：

```
{  
  "Languages": [  
    {  
      "LanguageCode": "en",  
      "Score": 0.9877256155014038  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[主要语言](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetectDominantLanguage](#)中的。

## detect-entities

以下代码示例演示了如何使用 detect-entities。

### AWS CLI

检测输入文本中的命名实体

以下 detect-entities 示例分析输入文本并返回命名实体。预训练模型的置信度分数也是每个预测的输出。

```
aws comprehend detect-entities \  
  --language-code en \  
  --text "It is a beautiful day in Seattle."
```



```
--text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
credit card \
  account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due by July
31st. Based on your autopay settings, \
  we will withdraw your payment on the due date from your bank account number
XXXXXX1111 with the routing number XXXXX0000. \
  Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments to
Alice at AnySpa@example.com."
```

输出：

```
{
  "Entities": [
    {
      "Score": 0.9994556307792664,
      "Type": "PERSON",
      "Text": "Zhang Wei",
      "BeginOffset": 6,
      "EndOffset": 15
    },
    {
      "Score": 0.9981022477149963,
      "Type": "PERSON",
      "Text": "John",
      "BeginOffset": 22,
      "EndOffset": 26
    },
    {
      "Score": 0.9986887574195862,
      "Type": "ORGANIZATION",
      "Text": "AnyCompany Financial Services, LLC",
      "BeginOffset": 33,
      "EndOffset": 67
    },
    {
      "Score": 0.9959119558334351,
      "Type": "OTHER",
      "Text": "1111-XXXX-1111-XXXX",
      "BeginOffset": 88,
      "EndOffset": 107
    },
    {
      "Score": 0.9708039164543152,
```

```
    "Type": "QUANTITY",
    "Text": ".53",
    "BeginOffset": 133,
    "EndOffset": 136
  },
  {
    "Score": 0.9987268447875977,
    "Type": "DATE",
    "Text": "July 31st",
    "BeginOffset": 152,
    "EndOffset": 161
  },
  {
    "Score": 0.9858865737915039,
    "Type": "OTHER",
    "Text": "XXXXXX1111",
    "BeginOffset": 271,
    "EndOffset": 281
  },
  {
    "Score": 0.9700471758842468,
    "Type": "OTHER",
    "Text": "XXXXX0000",
    "BeginOffset": 306,
    "EndOffset": 315
  },
  {
    "Score": 0.9591118693351746,
    "Type": "ORGANIZATION",
    "Text": "Sunshine Spa",
    "BeginOffset": 340,
    "EndOffset": 352
  },
  {
    "Score": 0.9797496795654297,
    "Type": "LOCATION",
    "Text": "123 Main St",
    "BeginOffset": 354,
    "EndOffset": 365
  },
  {
    "Score": 0.994929313659668,
    "Type": "PERSON",
    "Text": "Alice",
```

```
        "BeginOffset": 394,  
        "EndOffset": 399  
    },  
    {  
        "Score": 0.9949769377708435,  
        "Type": "OTHER",  
        "Text": "AnySpa@example.com",  
        "BeginOffset": 403,  
        "EndOffset": 418  
    }  
]  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[实体](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetectEntities](#)中的。

## detect-key-phrases

以下代码示例演示了如何使用 detect-key-phrases。

### AWS CLI

检测输入文本中的关键词

以下 detect-key-phrases 示例分析输入文本并识别关键名词短语。预训练模型的置信度分数也是每个预测的输出。

```
aws comprehend detect-key-phrases \  
  --language-code en \  
  --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC  
credit card \  
  account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due by  
July 31st. Based on your autopay settings, \  
  we will withdraw your payment on the due date from your bank account number  
XXXXXX1111 with the routing number XXXXX0000. \  
  Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments to  
Alice at AnySpa@example.com."
```

输出：

```
{  
  "KeyPhrases": [  
    {  
      "Text": "AnyCompany Financial Services, LLC",  
      "Score": 0.9949769377708435,  
      "Type": "OTHER",  
      "BeginOffset": 403,  
      "EndOffset": 418  
    }  
  ]  
}
```

```
{
  "Score": 0.8996376395225525,
  "Text": "Zhang Wei",
  "BeginOffset": 6,
  "EndOffset": 15
},
{
  "Score": 0.9992469549179077,
  "Text": "John",
  "BeginOffset": 22,
  "EndOffset": 26
},
{
  "Score": 0.988385021686554,
  "Text": "Your AnyCompany Financial Services",
  "BeginOffset": 28,
  "EndOffset": 62
},
{
  "Score": 0.8740853071212769,
  "Text": "LLC credit card account 1111-XXXX-1111-XXXX",
  "BeginOffset": 64,
  "EndOffset": 107
},
{
  "Score": 0.9999437928199768,
  "Text": "a minimum payment",
  "BeginOffset": 112,
  "EndOffset": 129
},
{
  "Score": 0.9998900890350342,
  "Text": ".53",
  "BeginOffset": 133,
  "EndOffset": 136
},
{
  "Score": 0.9979453086853027,
  "Text": "July 31st",
  "BeginOffset": 152,
  "EndOffset": 161
},
{
  "Score": 0.9983011484146118,
```

```
    "Text": "your autopay settings",
    "BeginOffset": 172,
    "EndOffset": 193
  },
  {
    "Score": 0.9996572136878967,
    "Text": "your payment",
    "BeginOffset": 211,
    "EndOffset": 223
  },
  {
    "Score": 0.9995037317276001,
    "Text": "the due date",
    "BeginOffset": 227,
    "EndOffset": 239
  },
  {
    "Score": 0.9702621698379517,
    "Text": "your bank account number XXXXXX1111",
    "BeginOffset": 245,
    "EndOffset": 280
  },
  {
    "Score": 0.9179925918579102,
    "Text": "the routing number XXXXX0000.Customer feedback",
    "BeginOffset": 286,
    "EndOffset": 332
  },
  {
    "Score": 0.9978160858154297,
    "Text": "Sunshine Spa",
    "BeginOffset": 337,
    "EndOffset": 349
  },
  {
    "Score": 0.9706913232803345,
    "Text": "123 Main St",
    "BeginOffset": 351,
    "EndOffset": 362
  },
  {
    "Score": 0.9941995143890381,
    "Text": "comments",
    "BeginOffset": 379,
```

```
        "EndOffset": 387
      },
      {
        "Score": 0.9759287238121033,
        "Text": "Alice",
        "BeginOffset": 391,
        "EndOffset": 396
      },
      {
        "Score": 0.8376792669296265,
        "Text": "AnySpa@example.com",
        "BeginOffset": 400,
        "EndOffset": 415
      }
    ]
  }
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[关键词](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetectKeyPhrases](#)中的。

## detect-pii-entities

以下代码示例演示了如何使用 detect-pii-entities。

### AWS CLI

检测输入文本中的 PII 实体

以下 detect-pii-entities 示例分析输入文本，并识别包含个人信息 (PII) 的实体。预训练模型的置信度分数也是每个预测的输出。

```
aws comprehend detect-pii-entities \
  --language-code en \
  --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
credit card \
  account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due by
July 31st. Based on your autopay settings, \
  we will withdraw your payment on the due date from your bank account number
XXXXXX1111 with the routing number XXXXX0000. \
  Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments to
Alice at AnySpa@example.com."
```

输出：

```
{
  "Entities": [
    {
      "Score": 0.9998322129249573,
      "Type": "NAME",
      "BeginOffset": 6,
      "EndOffset": 15
    },
    {
      "Score": 0.9998878240585327,
      "Type": "NAME",
      "BeginOffset": 22,
      "EndOffset": 26
    },
    {
      "Score": 0.9994089603424072,
      "Type": "CREDIT_DEBIT_NUMBER",
      "BeginOffset": 88,
      "EndOffset": 107
    },
    {
      "Score": 0.9999760985374451,
      "Type": "DATE_TIME",
      "BeginOffset": 152,
      "EndOffset": 161
    },
    {
      "Score": 0.9999449253082275,
      "Type": "BANK_ACCOUNT_NUMBER",
      "BeginOffset": 271,
      "EndOffset": 281
    },
    {
      "Score": 0.9999847412109375,
      "Type": "BANK_ROUTING",
      "BeginOffset": 306,
      "EndOffset": 315
    },
    {
      "Score": 0.999925434589386,
      "Type": "ADDRESS",
      "BeginOffset": 354,
```

```
        "EndOffset": 365
      },
      {
        "Score": 0.9989161491394043,
        "Type": "NAME",
        "BeginOffset": 394,
        "EndOffset": 399
      },
      {
        "Score": 0.9994171857833862,
        "Type": "EMAIL",
        "BeginOffset": 403,
        "EndOffset": 418
      }
    ]
  }
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[个人信息 \(PII\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetectPiiEntities](#)中的。

## detect-sentiment

以下代码示例演示了如何使用 detect-sentiment。

### AWS CLI

检测输入文本的情绪

以下 detect-sentiment 示例分析输入文本，并返回占主导地位的情绪 ( POSITIVE、NEUTRAL、MIXED 或NEGATIVE ) 的推断。

```
aws comprehend detect-sentiment \
  --language-code en \
  --text "It is a beautiful day in Seattle"
```

输出：

```
{
  "Sentiment": "POSITIVE",
  "SentimentScore": {
    "Positive": 0.9976957440376282,
    "Negative": 9.653854067437351e-05,
```



```
    "Neutral": 0.002169104292988777,  
    "Mixed": 3.857641786453314e-05  
  }  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[情绪](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetectSentiment](#)中的。

## detect-syntax

以下代码示例演示了如何使用 detect-syntax。

### AWS CLI

检测输入文本中的语音部分

以下 detect-syntax 示例分析输入文本的语法并返回语音的不同部分。预训练模型的置信度分数也是每个预测的输出。

```
aws comprehend detect-syntax \  
  --language-code en \  
  --text "It is a beautiful day in Seattle."
```

输出：

```
{  
  "SyntaxTokens": [  
    {  
      "TokenId": 1,  
      "Text": "It",  
      "BeginOffset": 0,  
      "EndOffset": 2,  
      "PartOfSpeech": {  
        "Tag": "PRON",  
        "Score": 0.9999740719795227  
      }  
    },  
    {  
      "TokenId": 2,  
      "Text": "is",  
      "BeginOffset": 3,  
      "EndOffset": 5,
```

```
    "PartOfSpeech": {
      "Tag": "VERB",
      "Score": 0.999901294708252
    }
  },
  {
    "TokenId": 3,
    "Text": "a",
    "BeginOffset": 6,
    "EndOffset": 7,
    "PartOfSpeech": {
      "Tag": "DET",
      "Score": 0.9999938607215881
    }
  },
  {
    "TokenId": 4,
    "Text": "beautiful",
    "BeginOffset": 8,
    "EndOffset": 17,
    "PartOfSpeech": {
      "Tag": "ADJ",
      "Score": 0.9987351894378662
    }
  },
  {
    "TokenId": 5,
    "Text": "day",
    "BeginOffset": 18,
    "EndOffset": 21,
    "PartOfSpeech": {
      "Tag": "NOUN",
      "Score": 0.9999796748161316
    }
  },
  {
    "TokenId": 6,
    "Text": "in",
    "BeginOffset": 22,
    "EndOffset": 24,
    "PartOfSpeech": {
      "Tag": "ADP",
      "Score": 0.9998047947883606
    }
  }
}
```

```
    },
    {
      "TokenId": 7,
      "Text": "Seattle",
      "BeginOffset": 25,
      "EndOffset": 32,
      "PartOfSpeech": {
        "Tag": "PROPN",
        "Score": 0.9940530061721802
      }
    }
  ]
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[语法分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetectSyntax](#)中的。

## detect-targeted-sentiment

以下代码示例演示了如何使用 detect-targeted-sentiment。

### AWS CLI

检测输入文本中命名实体的目标情绪

以下detect-targeted-sentiment示例分析输入文本，并返回命名实体以及与每个实体关联的目标情绪。还会输出每个预测的预训练模型置信度分数。

```
aws comprehend detect-targeted-sentiment \
  --language-code en \
  --text "I do not enjoy January because it is too cold but August is the perfect
  temperature"
```

输出：

```
{
  "Entities": [
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
```

```
        {
          "Score": 0.9999979734420776,
          "GroupScore": 1.0,
          "Text": "I",
          "Type": "PERSON",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Positive": 0.0,
              "Negative": 0.0,
              "Neutral": 1.0,
              "Mixed": 0.0
            }
          },
          "BeginOffset": 0,
          "EndOffset": 1
        }
      ],
    },
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "Score": 0.9638869762420654,
          "GroupScore": 1.0,
          "Text": "January",
          "Type": "DATE",
          "MentionSentiment": {
            "Sentiment": "NEGATIVE",
            "SentimentScore": {
              "Positive": 0.0031610000878572464,
              "Negative": 0.9967250227928162,
              "Neutral": 0.00011100000119768083,
              "Mixed": 1.9999999949504854e-06
            }
          },
          "BeginOffset": 15,
          "EndOffset": 22
        }
      ]
    }
  ],
}
```

```
    "DescriptiveMentionIndex": [
      0
    ],
    "Mentions": [
      {
        {
          "Score": 0.9664419889450073,
          "GroupScore": 1.0,
          "Text": "August",
          "Type": "DATE",
          "MentionSentiment": {
            "Sentiment": "POSITIVE",
            "SentimentScore": {
              "Positive": 0.9999549984931946,
              "Negative": 3.999999989900971e-06,
              "Neutral": 4.0999999932805076e-05,
              "Mixed": 0.0
            }
          }
        },
        "BeginOffset": 50,
        "EndOffset": 56
      }
    ]
  },
  {
    "DescriptiveMentionIndex": [
      0
    ],
    "Mentions": [
      {
        {
          "Score": 0.9803199768066406,
          "GroupScore": 1.0,
          "Text": "temperature",
          "Type": "ATTRIBUTE",
          "MentionSentiment": {
            "Sentiment": "POSITIVE",
            "SentimentScore": {
              "Positive": 1.0,
              "Negative": 0.0,
              "Neutral": 0.0,
              "Mixed": 0.0
            }
          }
        },
        "BeginOffset": 77,
```

```

    "EndOffset": 88
  }
]
}

```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的[定向情绪](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DetectTargetedSentiment](#)中的。

## import-model

以下代码示例演示了如何使用 import-model。

### AWS CLI

#### 要导入模型

以下 import-model 示例从不同的 AWS 账户导入模型。账户中的文档分类器模型 444455556666 具有基于资源的策略，允许账户 111122223333 导入模型。

```

aws comprehend import-model \
  --source-model-arn arn:aws:comprehend:us-west-2:444455556666:document-
  classifier/example-classifier

```

#### 输出：

```

{
  "ModelArn": "arn:aws:comprehend:us-west-2:111122223333:document-classifier/
  example-classifier"
}

```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的在[AWS 账户之间复制自定义模型](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ImportModel](#)中的。

## list-datasets

以下代码示例演示了如何使用 list-datasets。

## AWS CLI

### 列出所有飞轮数据集

以下`list-datasets`示例列出了与飞轮关联的所有数据集。

```
aws comprehend list-datasets \
  --flywheel-arn arn:aws:comprehend:us-west-2:111122223333:flywheel/flywheel-
  entity
```

输出：

```
{
  "DatasetPropertiesList": [
    {
      "DatasetArn": "arn:aws:comprehend:us-west-2:111122223333:flywheel/
flywheel-entity/dataset/example-dataset-1",
      "DatasetName": "example-dataset-1",
      "DatasetType": "TRAIN",
      "DatasetS3Uri": "s3://DOC-EXAMPLE-BUCKET/flywheel-entity/
schemaVersion=1/20230616T200543Z/datasets/example-dataset-1/20230616T203710Z/",
      "Status": "CREATING",
      "CreationTime": "2023-06-16T20:37:10.400000+00:00"
    },
    {
      "DatasetArn": "arn:aws:comprehend:us-west-2:111122223333:flywheel/
flywheel-entity/dataset/example-dataset-2",
      "DatasetName": "example-dataset-2",
      "DatasetType": "TRAIN",
      "DatasetS3Uri": "s3://DOC-EXAMPLE-BUCKET/flywheel-entity/
schemaVersion=1/20230616T200543Z/datasets/example-dataset-2/20230616T200607Z/",
      "Description": "TRAIN Dataset created by Flywheel creation.",
      "Status": "COMPLETED",
      "NumberOfDocuments": 5572,
      "CreationTime": "2023-06-16T20:06:07.722000+00:00"
    }
  ]
}
```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的[飞轮概述](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListDatasets](#)中的。

## list-document-classification-jobs

以下代码示例演示了如何使用 `list-document-classification-jobs`。

### AWS CLI

列出所有文档分类作业

以下 `list-document-classification-jobs` 示例列出所有文档分类作业。

```
aws comprehend list-document-classification-jobs
```

输出：

```
{
  "DocumentClassificationJobPropertiesList": [
    {
      "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
      "JobArn": "arn:aws:comprehend:us-west-2:1234567890101:document-
classification-job/123456abcdeb0e11022f22a11EXAMPLE",
      "JobName": "exampleclassificationjob",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2023-06-14T17:09:51.788000+00:00",
      "EndTime": "2023-06-14T17:15:58.582000+00:00",
      "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:1234567890101:document-classifier/mymodel/version/12",
      "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/jobdata/",
        "InputFormat": "ONE_DOC_PER_LINE"
      },
      "OutputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/1234567890101-CLN-e758dd56b824aa717ceab551f11749fb/output/output.tar.gz"
      },
      "DataAccessRoleArn": "arn:aws:iam::1234567890101:role/service-role/
AmazonComprehendServiceRole-example-role"
    },
    {
      "JobId": "123456abcdeb0e11022f22a11EXAMPLE2",
      "JobArn": "arn:aws:comprehend:us-west-2:1234567890101:document-
classification-job/123456abcdeb0e11022f22a11EXAMPLE2",
      "JobName": "exampleclassificationjob2",
      "JobStatus": "COMPLETED",
    }
  ]
}
```



```

    "SubmitTime": "2023-06-14T17:22:39.829000+00:00",
    "EndTime": "2023-06-14T17:28:46.107000+00:00",
    "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:1234567890101:document-classifier/mymodel/version/12",
    "InputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/jobdata/",
      "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/1234567890101-CLN-123456abcdeb0e11022f22a1EXAMPLE2/output/output.tar.gz"
    },
    "DataAccessRoleArn": "arn:aws:iam::1234567890101:role/service-role/
AmazonComprehendServiceRole-example-role"
  }
]
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[自定义分类](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDocumentClassificationJobs](#)中的。

## list-document-classifier-summaries

以下代码示例演示了如何使用 `list-document-classifier-summaries`。

### AWS CLI

列出所有已创建的文档分类器的摘要

以下`list-document-classifier-summaries`示例列出了所有创建的文档分类器摘要。

```
aws comprehend list-document-classifier-summaries
```

输出：

```

{
  "DocumentClassifierSummariesList": [
    {
      "DocumentClassifierName": "example-classifier-1",
      "NumberOfVersions": 1,
      "LatestVersionCreatedAt": "2023-06-13T22:07:59.825000+00:00",

```

```

        "LatestVersionName": "1",
        "LatestVersionStatus": "TRAINED"
    },
    {
        "DocumentClassifierName": "example-classifier-2",
        "NumberOfVersions": 2,
        "LatestVersionCreatedAt": "2023-06-13T21:54:59.589000+00:00",
        "LatestVersionName": "2",
        "LatestVersionStatus": "TRAINED"
    }
]
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[创建和管理自定义模型](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDocumentClassifierSummaries](#) 中的。

## list-document-classifiers

以下代码示例演示了如何使用 `list-document-classifiers`。

### AWS CLI

列出所有文档分类器

以下 `list-document-classifiers` 示例列出所有经过训练和正在训练的文档分类器模型。

```
aws comprehend list-document-classifiers
```

输出：

```

{
  "DocumentClassifierPropertiesList": [
    {
      "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/exampleclassifier1",
      "LanguageCode": "en",
      "Status": "TRAINED",
      "SubmitTime": "2023-06-13T19:04:15.735000+00:00",
      "EndTime": "2023-06-13T19:42:31.752000+00:00",
      "TrainingStartTime": "2023-06-13T19:08:20.114000+00:00",
      "TrainingEndTime": "2023-06-13T19:41:35.080000+00:00",
      "InputDataConfig": {

```

```

        "DataFormat": "COMPREHEND_CSV",
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/trainingdata"
    },
    "OutputDataConfig": {},
    "ClassifierMetadata": {
        "NumberOfLabels": 3,
        "NumberOfTrainedDocuments": 5016,
        "NumberOfTestDocuments": 557,
        "EvaluationMetrics": {
            "Accuracy": 0.9856,
            "Precision": 0.9919,
            "Recall": 0.9459,
            "F1Score": 0.9673,
            "MicroPrecision": 0.9856,
            "MicroRecall": 0.9856,
            "MicroF1Score": 0.9856,
            "HammingLoss": 0.0144
        }
    },
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/AmazonComprehendServiceRole-testorle",
    "Mode": "MULTI_CLASS"
},
{
    "DocumentClassifierArn": "arn:aws:comprehend:us-west-2:111122223333:document-classifier/exampleclassifier2",
    "LanguageCode": "en",
    "Status": "TRAINING",
    "SubmitTime": "2023-06-13T21:20:28.690000+00:00",
    "InputDataConfig": {
        "DataFormat": "COMPREHEND_CSV",
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/trainingdata"
    },
    "OutputDataConfig": {},
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/AmazonComprehendServiceRole-testorle",
    "Mode": "MULTI_CLASS"
}
]
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[创建和管理自定义模型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDocumentClassifiers](#)中的。

## list-dominant-language-detection-jobs

以下代码示例演示了如何使用 `list-dominant-language-detection-jobs`。

### AWS CLI

列出所有占主导地位的语言检测作业

以下 `list-dominant-language-detection-jobs` 示例列出了所有正在进行和已完成的异步主导语言检测作业。

```
aws comprehend list-dominant-language-detection-jobs
```

输出：

```
{
  "DominantLanguageDetectionJobPropertiesList": [
    {
      "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
      "JobArn": "arn:aws:comprehend:us-west-2:111122223333:dominant-language-detection-job/123456abcdeb0e11022f22a11EXAMPLE",
      "JobName": "languageanalysis1",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2023-06-09T18:10:38.037000+00:00",
      "EndTime": "2023-06-09T18:18:45.498000+00:00",
      "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET",
        "InputFormat": "ONE_DOC_PER_LINE"
      },
      "OutputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/111122223333-LANGUAGE-123456abcdeb0e11022f22a11EXAMPLE/output/output.tar.gz"
      },
      "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/AmazonComprehendServiceRole-example-role"
    },
    {
      "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
      "JobArn": "arn:aws:comprehend:us-west-2:111122223333:dominant-language-detection-job/123456abcdeb0e11022f22a11EXAMPLE",
      "JobName": "languageanalysis2",
      "JobStatus": "STOPPED",
    }
  ]
}
```

```

    "SubmitTime": "2023-06-09T18:16:33.690000+00:00",
    "EndTime": "2023-06-09T18:24:40.608000+00:00",
    "InputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET",
      "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
testfolder/111122223333-LANGUAGE-123456abcdeb0e11022f22a11EXAMPLE/output/
output.tar.gz"
    },
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
  }
]
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDominantLanguageDetectionJobs](#) 中的。

## list-endpoints

以下代码示例演示了如何使用 `list-endpoints`。

### AWS CLI

列出所有端点

以下 `list-endpoints` 示例列出了所有特定于模型的活动端点。

```
aws comprehend list-endpoints
```

输出：

```

{
  "EndpointPropertiesList": [
    {
      "EndpointArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier-endpoint/ExampleClassifierEndpoint",
      "Status": "IN_SERVICE",

```

```

        "ModelArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/exampleclassifier1",
        "DesiredModelArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/exampleclassifier1",
        "DesiredInferenceUnits": 1,
        "CurrentInferenceUnits": 1,
        "CreationTime": "2023-06-13T20:32:54.526000+00:00",
        "LastModifiedTime": "2023-06-13T20:32:54.526000+00:00"
    },
    {
        "EndpointArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier-endpoint/ExampleClassifierEndpoint2",
        "Status": "IN_SERVICE",
        "ModelArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/exampleclassifier2",
        "DesiredModelArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/exampleclassifier2",
        "DesiredInferenceUnits": 1,
        "CurrentInferenceUnits": 1,
        "CreationTime": "2023-06-13T20:32:54.526000+00:00",
        "LastModifiedTime": "2023-06-13T20:32:54.526000+00:00"
    }
]
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[管理 Amazon Comprehend 端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListEndpoints](#)中的。

## list-entities-detection-jobs

以下代码示例演示了如何使用 list-entities-detection-jobs。

### AWS CLI

列出所有实体检测任务

以下 list-entities-detection-jobs 示例列出了所有异步实体检测作业。

```
aws comprehend list-entities-detection-jobs
```

输出：

```
{
  "EntitiesDetectionJobPropertiesList": [
    {
      "JobId": "468af39c28ab45b83eb0c4ab9EXAMPLE",
      "JobArn": "arn:aws:comprehend:us-west-2:111122223333:entities-detection-
job/468af39c28ab45b83eb0c4ab9EXAMPLE",
      "JobName": "example-entities-detection",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2023-06-08T20:57:46.476000+00:00",
      "EndTime": "2023-06-08T21:05:53.718000+00:00",
      "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/AsyncBatchJobs/",
        "InputFormat": "ONE_DOC_PER_LINE"
      },
      "OutputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/111122223333-NER-468af39c28ab45b83eb0c4ab9EXAMPLE/output/output.tar.gz"
      },
      "LanguageCode": "en",
      "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
    },
    {
      "JobId": "809691caeaab0e71406f80a28EXAMPLE",
      "JobArn": "arn:aws:comprehend:us-west-2:111122223333:entities-detection-
job/809691caeaab0e71406f80a28EXAMPLE",
      "JobName": "example-entities-detection-2",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2023-06-08T21:30:15.323000+00:00",
      "EndTime": "2023-06-08T21:40:23.509000+00:00",
      "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/AsyncBatchJobs/",
        "InputFormat": "ONE_DOC_PER_LINE"
      },
      "OutputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/111122223333-NER-809691caeaab0e71406f80a28EXAMPLE/output/output.tar.gz"
      },
      "LanguageCode": "en",
      "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
    },
    {
```

```

    "JobId": "e00597c36b448b91d70dea165EXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:entities-detection-
job/e00597c36b448b91d70dea165EXAMPLE",
    "JobName": "example-entities-detection-3",
    "JobStatus": "STOPPED",
    "SubmitTime": "2023-06-08T22:19:28.528000+00:00",
    "EndTime": "2023-06-08T22:27:33.991000+00:00",
    "InputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/AsyncBatchJobs/",
      "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/111122223333-NER-e00597c36b448b91d70dea165EXAMPLE/output/output.tar.gz"
    },
    "LanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
  }
]
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[实体](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListEntitiesDetectionJobs](#)中的。

## list-entity-recognizer-summaries

以下代码示例演示了如何使用 list-entity-recognizer-summaries。

### AWS CLI

查看所有已创建的实体识别器的摘要列表

以下list-entity-recognizer-summaries示例列出了所有实体识别器摘要。

```
aws comprehend list-entity-recognizer-summaries
```

输出：

```
{
  "EntityRecognizerSummariesList": [
    {
```



```
    "RecognizerName": "entity-recognizer-3",
    "NumberOfVersions": 2,
    "LatestVersionCreatedAt": "2023-06-15T23:15:07.621000+00:00",
    "LatestVersionName": "2",
    "LatestVersionStatus": "STOP_REQUESTED"
  },
  {
    "RecognizerName": "entity-recognizer-2",
    "NumberOfVersions": 1,
    "LatestVersionCreatedAt": "2023-06-14T22:55:27.805000+00:00",
    "LatestVersionName": "2"
    "LatestVersionStatus": "TRAINED"
  },
  {
    "RecognizerName": "entity-recognizer-1",
    "NumberOfVersions": 1,
    "LatestVersionCreatedAt": "2023-06-14T20:44:59.631000+00:00",
    "LatestVersionName": "1",
    "LatestVersionStatus": "TRAINED"
  }
]
}
```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的[自定义实体识别](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListEntityRecognizerSummaries](#)中的。

## list-entity-recognizers

以下代码示例演示了如何使用 list-entity-recognizers。

### AWS CLI

列出所有自定义实体识别器

以下list-entity-recognizers示例列出了所有创建的自定义实体识别器。

```
aws comprehend list-entity-recognizers
```

输出：

```
{
  "EntityRecognizerPropertiesList": [
```

```
{
  "EntityRecognizerArn": "arn:aws:comprehend:us-
west-2:111122223333:entity-recognizer/EntityRecognizer/version/1",
  "LanguageCode": "en",
  "Status": "TRAINED",
  "SubmitTime": "2023-06-14T20:44:59.631000+00:00",
  "EndTime": "2023-06-14T20:59:19.532000+00:00",
  "TrainingStartTime": "2023-06-14T20:48:52.811000+00:00",
  "TrainingEndTime": "2023-06-14T20:58:11.473000+00:00",
  "InputDataConfig": {
    "DataFormat": "COMPREHEND_CSV",
    "EntityTypes": [
      {
        "Type": "BUSINESS"
      }
    ],
    "Documents": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/trainingdata/dataset/",
      "InputFormat": "ONE_DOC_PER_LINE"
    },
    "EntityList": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/trainingdata/entity.csv"
    }
  },
  "RecognizerMetadata": {
    "NumberOfTrainedDocuments": 1814,
    "NumberOfTestDocuments": 486,
    "EvaluationMetrics": {
      "Precision": 100.0,
      "Recall": 100.0,
      "F1Score": 100.0
    },
    "EntityTypes": [
      {
        "Type": "BUSINESS",
        "EvaluationMetrics": {
          "Precision": 100.0,
          "Recall": 100.0,
          "F1Score": 100.0
        }
      },
      "NumberOfTrainMentions": 1520
    ]
  }
},
```

```
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-servicerole",
    "VersionName": "1"
  },
  {
    "EntityRecognizerArn": "arn:aws:comprehend:us-
west-2:111122223333:entity-recognizer/entityrecognizer3",
    "LanguageCode": "en",
    "Status": "TRAINED",
    "SubmitTime": "2023-06-14T22:57:51.056000+00:00",
    "EndTime": "2023-06-14T23:14:13.894000+00:00",
    "TrainingStartTime": "2023-06-14T23:01:33.984000+00:00",
    "TrainingEndTime": "2023-06-14T23:13:02.984000+00:00",
    "InputDataConfig": {
      "DataFormat": "COMPREHEND_CSV",
      "EntityTypes": [
        {
          "Type": "DEVICE"
        }
      ],
      "Documents": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/trainingdata/raw_txt.csv",
        "InputFormat": "ONE_DOC_PER_LINE"
      },
      "EntityList": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/trainingdata/entity_list.csv"
      }
    },
    "RecognizerMetadata": {
      "NumberOfTrainedDocuments": 4616,
      "NumberOfTestDocuments": 3489,
      "EvaluationMetrics": {
        "Precision": 98.54227405247813,
        "Recall": 100.0,
        "F1Score": 99.26578560939794
      }
    },
    "EntityTypes": [
      {
        "Type": "DEVICE",
        "EvaluationMetrics": {
          "Precision": 98.54227405247813,
          "Recall": 100.0,
          "F1Score": 99.26578560939794
        }
      }
    ]
  },
}
```

```

        "NumberOfTrainMentions": 2764
      }
    ]
  },
  "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-servicerole"
}
]
}

```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的[自定义实体识别](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListEntityRecognizers](#)中的。

## list-events-detection-jobs

以下代码示例演示了如何使用 list-events-detection-jobs。

### AWS CLI

列出所有事件检测作业

以下 list-events-detection-jobs 示例列出了所有异步事件检测作业。

```
aws comprehend list-events-detection-jobs
```

输出：

```

{
  "EventsDetectionJobPropertiesList": [
    {
      "JobId": "aa9593f9203e84f3ef032ce18EXAMPLE",
      "JobArn": "arn:aws:comprehend:us-west-2:111122223333:events-detection-
job/aa9593f9203e84f3ef032ce18EXAMPLE",
      "JobName": "events_job_1",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2023-06-12T19:14:57.751000+00:00",
      "EndTime": "2023-06-12T19:21:04.962000+00:00",
      "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-SOURCE-BUCKET/EventsData/",
        "InputFormat": "ONE_DOC_PER_LINE"
      },
      "OutputDataConfig": {

```

```

        "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
testfolder/1111222233333-EVENTS-aa9593f9203e84f3ef032ce18EXAMPLE/output/"
    },
    "LanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::1111222233333:role/service-role/
AmazonComprehendServiceRole-example-role",
    "TargetEventTypes": [
        "BANKRUPTCY",
        "EMPLOYMENT",
        "CORPORATE_ACQUISITION",
        "CORPORATE_MERGER",
        "INVESTMENT_GENERAL"
    ]
},
{
    "JobId": "4a990a2f7e82adfca6e171135EXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:1111222233333:events-detection-
job/4a990a2f7e82adfca6e171135EXAMPLE",
    "JobName": "events_job_2",
    "JobStatus": "COMPLETED",
    "SubmitTime": "2023-06-12T19:55:43.702000+00:00",
    "EndTime": "2023-06-12T20:03:49.893000+00:00",
    "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-SOURCE-BUCKET/EventsData/",
        "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
testfolder/1111222233333-EVENTS-4a990a2f7e82adfca6e171135EXAMPLE/output/"
    },
    "LanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::1111222233333:role/service-role/
AmazonComprehendServiceRole-example-role",
    "TargetEventTypes": [
        "BANKRUPTCY",
        "EMPLOYMENT",
        "CORPORATE_ACQUISITION",
        "CORPORATE_MERGER",
        "INVESTMENT_GENERAL"
    ]
}
]
}
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListEventsDetectionJobs](#) 中的。

## list-flywheel-iteration-history

以下代码示例演示了如何使用 list-flywheel-iteration-history。

### AWS CLI

列出所有飞轮迭代历史记录

以下 list-flywheel-iteration-history 示例列出了飞轮的所有迭代。

```
aws comprehend list-flywheel-iteration-history
  --flywheel-arn arn:aws:comprehend:us-west-2:111122223333:flywheel/example-
  flywheel
```

输出：

```
{
  "FlywheelIterationPropertiesList": [
    {
      "FlywheelArn": "arn:aws:comprehend:us-west-2:111122223333:flywheel/
example-flywheel",
      "FlywheelIterationId": "20230619EXAMPLE",
      "CreationTime": "2023-06-19T04:00:32.594000+00:00",
      "EndTime": "2023-06-19T04:00:49.248000+00:00",
      "Status": "COMPLETED",
      "Message": "FULL_ITERATION: Flywheel iteration performed all functions
successfully.",
      "EvaluatedModelArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/example-classifier/version/1",
      "EvaluatedModelMetrics": {
        "AverageF1Score": 0.7742663922375772,
        "AverageF1Score": 0.9876464664646313,
        "AveragePrecision": 0.9800000253081214,
        "AverageRecall": 0.9445600253081214,
        "AverageAccuracy": 0.9997281665190434
      },
      "EvaluationManifestS3Prefix": "s3://DOC-EXAMPLE-BUCKET/example-flywheel/
schemaVersion=1/20230619EXAMPLE/evaluation/20230619EXAMPLE/"
    }
  ]
}
```

```
    },
    {
      "FlywheelArn": "arn:aws:comprehend:us-west-2:111122223333:flywheel/
example-flywheel-2",
      "FlywheelIterationId": "20230616EXAMPLE",
      "CreationTime": "2023-06-16T21:10:26.385000+00:00",
      "EndTime": "2023-06-16T23:33:16.827000+00:00",
      "Status": "COMPLETED",
      "Message": "FULL_ITERATION: Flywheel iteration performed all functions
successfully.",
      "EvaluatedModelArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/spamvshamclassify/version/1",
      "EvaluatedModelMetrics": {
        "AverageF1Score": 0.7742663922375772,
        "AverageF1Score": 0.9767700253081214,
        "AveragePrecision": 0.9767700253081214,
        "AverageRecall": 0.9767700253081214,
        "AverageAccuracy": 0.9858281665190434
      },
      "EvaluationManifestS3Prefix": "s3://DOC-EXAMPLE-BUCKET/example-
flywheel-2/schemaVersion=1/20230616EXAMPLE/evaluation/20230616EXAMPLE/"
    }
  ]
}
```

有关更多信息，请参阅《亚马逊 Comprehend 开发者指南》中的 [Flywheel 概述](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListFlywheelIterationHistory](#) 中的。

## list-flywheels

以下代码示例演示了如何使用 list-flywheels。

### AWS CLI

列出所有飞轮

以下 list-flywheels 示例列出了所有创建的飞轮。

```
aws comprehend list-flywheels
```

输出：

```
{
  "FlywheelSummaryList": [
    {
      "FlywheelArn": "arn:aws:comprehend:us-west-2:111122223333:flywheel/
example-flywheel-1",
      "ActiveModelArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/exampleclassifier/version/1",
      "DataLakeS3Uri": "s3://DOC-EXAMPLE-BUCKET/example-flywheel-1/
schemaVersion=1/20230616T200543Z/",
      "Status": "ACTIVE",
      "ModelType": "DOCUMENT_CLASSIFIER",
      "CreationTime": "2023-06-16T20:05:43.242000+00:00",
      "LastModifiedTime": "2023-06-19T04:00:43.027000+00:00",
      "LatestFlywheelIteration": "20230619T040032Z"
    },
    {
      "FlywheelArn": "arn:aws:comprehend:us-west-2:111122223333:flywheel/
example-flywheel-2",
      "ActiveModelArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/exampleclassifier2/version/1",
      "DataLakeS3Uri": "s3://DOC-EXAMPLE-BUCKET/example-flywheel-2/
schemaVersion=1/20220616T200543Z/",
      "Status": "ACTIVE",
      "ModelType": "DOCUMENT_CLASSIFIER",
      "CreationTime": "2022-06-16T20:05:43.242000+00:00",
      "LastModifiedTime": "2022-06-19T04:00:43.027000+00:00",
      "LatestFlywheelIteration": "20220619T040032Z"
    }
  ]
}
```

有关更多信息，请参阅《亚马逊 Comprehend 开发者指南》中的 [Flywheel 概述](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListFlywheels](#) 中的。

## list-key-phrases-detection-jobs

以下代码示例演示了如何使用 list-key-phrases-detection-jobs。

### AWS CLI

列出所有关键短语检测作业



以下 `list-key-phrases-detection-jobs` 示例列出了所有正在进行和已完成的异步关键词检测作业。

```
aws comprehend list-key-phrases-detection-jobs
```

输出：

```
{
  "KeyPhrasesDetectionJobPropertiesList": [
    {
      "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
      "JobArn": "arn:aws:comprehend:us-west-2:111122223333:key-phrases-
detection-job/123456abcdeb0e11022f22a11EXAMPLE",
      "JobName": "keyphrasesanalysis1",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2023-06-08T22:31:43.767000+00:00",
      "EndTime": "2023-06-08T22:39:52.565000+00:00",
      "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-SOURCE-BUCKET/AsyncBatchJobs/",
        "InputFormat": "ONE_DOC_PER_LINE"
      },
      "OutputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
testfolder/111122223333-KP-123456abcdeb0e11022f22a11EXAMPLE/output/output.tar.gz"
      },
      "LanguageCode": "en",
      "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
    },
    {
      "JobId": "123456abcdeb0e11022f22a33EXAMPLE",
      "JobArn": "arn:aws:comprehend:us-west-2:111122223333:key-phrases-
detection-job/123456abcdeb0e11022f22a33EXAMPLE",
      "JobName": "keyphrasesanalysis2",
      "JobStatus": "STOPPED",
      "SubmitTime": "2023-06-08T22:57:52.154000+00:00",
      "EndTime": "2023-06-08T23:05:48.385000+00:00",
      "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/AsyncBatchJobs/",
        "InputFormat": "ONE_DOC_PER_LINE"
      },
      "OutputDataConfig": {
```

```

        "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
testfolder/111122223333-KP-123456abcdeb0e11022f22a33EXAMPLE/output/output.tar.gz"
    },
    "LanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
  },
  {
    "JobId": "123456abcdeb0e11022f22a44EXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:key-phrases-
detection-job/123456abcdeb0e11022f22a44EXAMPLE",
    "JobName": "keyphrasesanalysis3",
    "JobStatus": "FAILED",
    "Message": "NO_READ_ACCESS_TO_INPUT: The provided data access role does
not have proper access to the input data.",
    "SubmitTime": "2023-06-09T16:47:04.029000+00:00",
    "EndTime": "2023-06-09T16:47:18.413000+00:00",
    "InputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET",
      "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
testfolder/111122223333-KP-123456abcdeb0e11022f22a44EXAMPLE/output/output.tar.gz"
    },
    "LanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
  }
]
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListKeyPhrasesDetectionJobs](#) 中的。

## list-pii-entities-detection-jobs

以下代码示例演示了如何使用 `list-pii-entities-detection-jobs`。

## AWS CLI

### 列出所有 pii 实体检测作业

以下 `list-pii-entities-detection-jobs` 示例列出了所有正在进行和已完成的异步 pii 检测作业。

```
aws comprehend list-pii-entities-detection-jobs
```

输出：

```
{
  "PiiEntitiesDetectionJobPropertiesList": [
    {
      "JobId": "6f9db0c42d0c810e814670ee4EXAMPLE",
      "JobArn": "arn:aws:comprehend:us-west-2:111122223333:pii-entities-detection-job/6f9db0c42d0c810e814670ee4EXAMPLE",
      "JobName": "example-pii-detection-job",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2023-06-09T21:02:46.241000+00:00",
      "EndTime": "2023-06-09T21:12:52.602000+00:00",
      "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/AsyncBatchJobs/",
        "InputFormat": "ONE_DOC_PER_LINE"
      },
      "OutputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-SOURCE-BUCKET/111122223333-PII-6f9db0c42d0c810e814670ee4EXAMPLE/output/"
      },
      "LanguageCode": "en",
      "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/AmazonComprehendServiceRole-example-role",
      "Mode": "ONLY_OFFSETS"
    },
    {
      "JobId": "d927562638cfa739331a99b3cEXAMPLE",
      "JobArn": "arn:aws:comprehend:us-west-2:111122223333:pii-entities-detection-job/d927562638cfa739331a99b3cEXAMPLE",
      "JobName": "example-pii-detection-job-2",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2023-06-09T21:20:58.211000+00:00",
      "EndTime": "2023-06-09T21:31:06.027000+00:00",
      "InputDataConfig": {
```

```

        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/AsyncBatchJobs/",
        "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/111122223333-PII-d927562638cfa739331a99b3cEXAMPLE/output/"
    },
    "LanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role",
    "Mode": "ONLY_OFFSETS"
    }
]
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPiiEntitiesDetectionJobs](#)中的。

## list-sentiment-detection-jobs

以下代码示例演示了如何使用 list-sentiment-detection-jobs。

### AWS CLI

列出所有情绪检测作业

以下list-sentiment-detection-jobs示例列出了所有正在进行和已完成的异步情绪检测作业。

```
aws comprehend list-sentiment-detection-jobs
```

输出：

```

{
  "SentimentDetectionJobPropertiesList": [
    {
      "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
      "JobArn": "arn:aws:comprehend:us-west-2:111122223333:sentiment-
detection-job/123456abcdeb0e11022f22a11EXAMPLE",

```

```

    "JobName": "example-sentiment-detection-job",
    "JobStatus": "IN_PROGRESS",
    "SubmitTime": "2023-06-09T22:42:20.545000+00:00",
    "EndTime": "2023-06-09T22:52:27.416000+00:00",
    "InputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/MovieData",
      "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
testfolder/111122223333-TS-123456abcdeb0e11022f22a11EXAMPLE/output/output.tar.gz"
    },
    "LanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
  },
  {
    "JobId": "123456abcdeb0e11022f22a11EXAMPLE2",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:sentiment-
detection-job/123456abcdeb0e11022f22a11EXAMPLE2",
    "JobName": "example-sentiment-detection-job-2",
    "JobStatus": "COMPLETED",
    "SubmitTime": "2023-06-09T23:16:15.956000+00:00",
    "EndTime": "2023-06-09T23:26:00.168000+00:00",
    "InputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/MovieData2",
      "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
testfolder/111122223333-TS-123456abcdeb0e11022f22a11EXAMPLE2/output/output.tar.gz"
    },
    "LanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
  }
]
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSentimentDetectionJobs](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出资源的标签

以下 `list-tags-for-resource` 示例列出了 Amazon Comprehend 资源的标签。

```
aws comprehend list-tags-for-resource \  
  --resource-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/  
example-classifier/version/1
```

输出：

```
{  
  "ResourceArn": "arn:aws:comprehend:us-west-2:111122223333:document-classifier/  
example-classifier/version/1",  
  "Tags": [  
    {  
      "Key": "Department",  
      "Value": "Finance"  
    },  
    {  
      "Key": "location",  
      "Value": "Seattle"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon Comprehend [开发者指南中的为资源添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## list-targeted-sentiment-detection-jobs

以下代码示例演示了如何使用 `list-targeted-sentiment-detection-jobs`。

### AWS CLI

列出所有有针对性的情绪检测作业

以下`list-targeted-sentiment-detection-jobs`示例列出了所有正在进行和已完成的异步定向情绪检测作业。

```
aws comprehend list-targeted-sentiment-detection-jobs
```

输出：

```
{
  "TargetedSentimentDetectionJobPropertiesList": [
    {
      "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
      "JobArn": "arn:aws:comprehend:us-west-2:111122223333:targeted-sentiment-detection-job/123456abcdeb0e11022f22a11EXAMPLE",
      "JobName": "example-targeted-sentiment-detection-job",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2023-06-09T22:42:20.545000+00:00",
      "EndTime": "2023-06-09T22:52:27.416000+00:00",
      "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/MovieData",
        "InputFormat": "ONE_DOC_PER_LINE"
      },
      "OutputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/111122223333-TS-123456abcdeb0e11022f22a11EXAMPLE/output/output.tar.gz"
      },
      "LanguageCode": "en",
      "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/AmazonComprehendServiceRole-I0role"
    },
    {
      "JobId": "123456abcdeb0e11022f22a11EXAMPLE2",
      "JobArn": "arn:aws:comprehend:us-west-2:111122223333:targeted-sentiment-detection-job/123456abcdeb0e11022f22a11EXAMPLE2",
      "JobName": "example-targeted-sentiment-detection-job-2",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2023-06-09T23:16:15.956000+00:00",
      "EndTime": "2023-06-09T23:26:00.168000+00:00",
      "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/MovieData2",
        "InputFormat": "ONE_DOC_PER_LINE"
      },
      "OutputDataConfig": {
```

```

        "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
testfolder/111122223333-TS-123456abcdeb0e11022f22a1EXAMPLE2/output/output.tar.gz"
    },
    "LanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
    }
]
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTargetedSentimentDetectionJobs](#) 中的。

## list-topics-detection-jobs

以下代码示例演示了如何使用 list-topics-detection-jobs。

### AWS CLI

列出所有主题检测作业

以下 list-topics-detection-jobs 示例列出所有正在进行和已完成的异步主题检测作业。

```
aws comprehend list-topics-detection-jobs
```

输出：

```

{
  "TopicsDetectionJobPropertiesList": [
    {
      "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
      "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-detection-
job/123456abcdeb0e11022f22a11EXAMPLE",
      "JobName": "topic-analysis-1"
      "JobStatus": "IN_PROGRESS",
      "SubmitTime": "2023-06-09T18:40:35.384000+00:00",
      "EndTime": "2023-06-09T18:46:41.936000+00:00",
      "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET",
        "InputFormat": "ONE_DOC_PER_LINE"
      }
    },
  ],
}

```



```
    "OutputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a1EXAMPLE/output/output.tar.gz"
    },
    "NumberOfTopics": 10,
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
  },
  {
    "JobId": "123456abcdeb0e11022f22a1EXAMPLE2",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-detection-
job/123456abcdeb0e11022f22a1EXAMPLE2",
    "JobName": "topic-analysis-2",
    "JobStatus": "COMPLETED",
    "SubmitTime": "2023-06-09T18:44:43.414000+00:00",
    "EndTime": "2023-06-09T18:50:50.872000+00:00",
    "InputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET",
      "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a1EXAMPLE2/output/output.tar.gz"
    },
    "NumberOfTopics": 10,
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
  },
  {
    "JobId": "123456abcdeb0e11022f22a1EXAMPLE3",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-detection-
job/123456abcdeb0e11022f22a1EXAMPLE3",
    "JobName": "topic-analysis-2",
    "JobStatus": "IN_PROGRESS",
    "SubmitTime": "2023-06-09T18:50:56.737000+00:00",
    "InputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET",
      "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a1EXAMPLE3/output/output.tar.gz"
    },
    "NumberOfTopics": 10,
```

```

        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
    }
]
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTopicsDetectionJobs](#)中的。

## put-resource-policy

以下代码示例演示了如何使用 put-resource-policy。

### AWS CLI

#### 附加基于资源的策略

以下put-resource-policy示例将基于资源的策略附加到模型，以便其他 AWS 账户可以导入该模型。该策略附加到账户中的模型，111122223333并允许账户444455556666导入模型。

```

aws comprehend put-resource-policy \
  --resource-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/
example-classifier/version/1 \
  --resource-policy '{"Version":"2012-10-17","Statement":
[{"Effect":"Allow","Action":"comprehend:ImportModel","Resource":"*","Principal":
{"AWS":["arn:aws:iam::444455556666:root"]}]}]'

```

输出：

```

{
  "PolicyRevisionId": "aaa111d069d07afaa2aa3106aEXAMPLE"
}

```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的在[AWS 账户之间复制自定义模型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutResourcePolicy](#)中的。

## start-document-classification-job

以下代码示例演示了如何使用 start-document-classification-job。

## AWS CLI

### 列出文档分类作业

以下 `start-document-classification-job` 示例以自定义模型启动文档分类作业，该作业对 `--input-data-config` 标签所指定地址处的所有文件都使用自定义模型。在此示例中，输入 S3 存储桶包含 `SampleSMStext1.txt`、`SampleSMStext2.txt`、和 `SampleSMStext3.txt`。该模型之前曾接受过关于垃圾邮件和非垃圾邮件，或“ham”、短信的文档分类训练。作业完成后，`output.tar.gz` 将放置在 `--output-data-config` 标签指定的位置。`output.tar.gz` 包含 `predictions.jsonl`，其中列出了每个文档的分类。Json 输出在每个文件的一行上打印，但是为了便于阅读，此处设置了格式。

```
aws comprehend start-document-classification-job \  
  --job-name exampleclassificationjob \  
  --input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET-INPUT/jobdata/" \  
  --output-data-config "S3Uri=s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/" \  
  --data-access-role-arn arn:aws:iam::111122223333:role/service-role/  
AmazonComprehendServiceRole-example-role \  
  --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-  
classifier/mymodel/version/12
```

`SampleSMStext1.txt` 的内容：

```
"CONGRATULATIONS! TXT 2155550100 to win $5000"
```

`SampleSMStext2.txt` 的内容：

```
"Hi, when do you want me to pick you up from practice?"
```

`SampleSMStext3.txt` 的内容：

```
"Plz send bank account # to 2155550100 to claim prize!!"
```

输出：

```
{  
  "JobId": "e758dd56b824aa717ceab551fEXAMPLE",  
  "JobArn": "arn:aws:comprehend:us-west-2:111122223333:document-classification-  
job/e758dd56b824aa717ceab551fEXAMPLE",
```

```
"JobStatus": "SUBMITTED"
}
```

predictions.jsonl 的内容：

```
{"File": "SampleSMSText1.txt", "Line": "0", "Classes": [{"Name": "spam", "Score": 0.9999}, {"Name": "ham", "Score": 0.0001}]}
{"File": "SampleSMStext2.txt", "Line": "0", "Classes": [{"Name": "ham", "Score": 0.9994}, {"Name": "spam", "Score": 0.0006}]}
{"File": "SampleSMSText3.txt", "Line": "0", "Classes": [{"Name": "spam", "Score": 0.9999}, {"Name": "ham", "Score": 0.0001}]}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[自定义分类](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartDocumentClassificationJob](#)中的。

## start-dominant-language-detection-job

以下代码示例演示了如何使用 start-dominant-language-detection-job。

### AWS CLI

#### 启动异步语言检测作业

以下start-dominant-language-detection-job示例为位于--input-data-config标签指定地址的所有文件启动异步语言检测作业。本示例中的 S3 存储桶包含Sampletext1.txt。任务完成后output，文件夹将放置在--output-data-config标签指定的位置。output.txt该文件夹包含每个文本文件的主要语言以及每个预测的预训练模型的置信度分数。

```
aws comprehend start-dominant-language-detection-job \
  --job-name example_language_analysis_job \
  --language-code en \
  --input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET/" \
  --output-data-config "S3Uri=s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/" \
  --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role \
  --language-code en
```

Sampletext1.txt 的内容：

```
"Physics is the natural science that involves the study of matter and its motion and behavior through space and time, along with related concepts such as energy and force."
```

输出：

```
{
  "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
  "JobArn": "arn:aws:comprehend:us-west-2:111122223333:dominant-language-detection-job/123456abcdeb0e11022f22a11EXAMPLE",
  "JobStatus": "SUBMITTED"
}
```

output.txt 的内容：

```
{"File": "Sampletext1.txt", "Languages": [{"LanguageCode": "en", "Score": 0.9913753867149353}], "Line": 0}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartDominantLanguageDetectionJob](#) 中的。

## start-entities-detection-job

以下代码示例演示了如何使用 start-entities-detection-job。

### AWS CLI

示例 1：使用预训练模型启动标准实体检测作业

以下 start-entities-detection-job 示例为位于 --input-data-config 标签指定地址的所有文件启动异步实体检测作业。本示例中的 S3 存储桶包含 Sampletext1.txt、Sampletext2.txt、和 Sampletext3.txt。任务完成后，output 文件夹将放置在 --output-data-config 标签指定的位置。output.txt 该文件夹中列出了在每个文本文件中检测到的所有命名实体，以及预训练模型对每个预测的置信度得分。Json 输出在每个输入文件中打印一行，但为了便于阅读，在此处进行了格式化。

```
aws comprehend start-entities-detection-job \
  --job-name entitiestest \
  --language-code en \
```

```
--input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET/" \  
--output-data-config "S3Uri=s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/" \  
--data-access-role-arn arn:aws:iam::111122223333:role/service-role/  
AmazonComprehendServiceRole-example-role \  
--language-code en
```

Sampletext1.txt 的内容 :

```
"Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC credit card  
account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due by July  
31st."
```

Sampletext2.txt 的内容 :

```
"Dear Max, based on your autopay settings for your account example1.org account, we  
will withdraw your payment on the due date from your bank account number XXXXXX1111  
with the routing number XXXXX0000. "
```

Sampletext3.txt 的内容 :

```
"Jane, please submit any customer feedback from this weekend to AnySpa, 123 Main St,  
Anywhere and send comments to Alice at AnySpa@example.com."
```

输出 :

```
{  
  "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
  "JobArn": "arn:aws:comprehend:us-west-2:111122223333:entities-detection-  
job/123456abcdeb0e11022f22a11EXAMPLE",  
  "JobStatus": "SUBMITTED"  
}
```

为了便于阅读output.txt , 带有行缩进的内容 :

```
{  
  "Entities": [  
    {  
      "BeginOffset": 6,  
      "EndOffset": 15,  
      "Score": 0.9994006636420306,  
      "Text": "Zhang Wei",
```

```
"Type": "PERSON"
},
{
  "BeginOffset": 22,
  "EndOffset": 26,
  "Score": 0.9976647915128143,
  "Text": "John",
  "Type": "PERSON"
},
{
  "BeginOffset": 33,
  "EndOffset": 67,
  "Score": 0.9984608700836206,
  "Text": "AnyCompany Financial Services, LLC",
  "Type": "ORGANIZATION"
},
{
  "BeginOffset": 88,
  "EndOffset": 107,
  "Score": 0.9868521019555556,
  "Text": "1111-XXXX-1111-XXXX",
  "Type": "OTHER"
},
{
  "BeginOffset": 133,
  "EndOffset": 139,
  "Score": 0.998242565709204,
  "Text": "$24.53",
  "Type": "QUANTITY"
},
{
  "BeginOffset": 155,
  "EndOffset": 164,
  "Score": 0.9993039263159287,
  "Text": "July 31st",
  "Type": "DATE"
}
],
"File": "SampleText1.txt",
"Line": 0
}
{
  "Entities": [
    {
```

```
"BeginOffset": 5,
"EndOffset": 8,
"Score": 0.9866232147545232,
"Text": "Max",
"Type": "PERSON"
},
{
  "BeginOffset": 156,
  "EndOffset": 166,
  "Score": 0.9797723450933329,
  "Text": "XXXXXX1111",
  "Type": "OTHER"
},
{
  "BeginOffset": 191,
  "EndOffset": 200,
  "Score": 0.9247838572396843,
  "Text": "XXXXX0000",
  "Type": "OTHER"
}
],
"File": "SampleText2.txt",
"Line": 0
}
{
  "Entities": [
    {
      "Score": 0.9990532994270325,
      "Type": "PERSON",
      "Text": "Jane",
      "BeginOffset": 0,
      "EndOffset": 4
    },
    {
      "Score": 0.9519651532173157,
      "Type": "DATE",
      "Text": "this weekend",
      "BeginOffset": 47,
      "EndOffset": 59
    },
    {
      "Score": 0.5566426515579224,
      "Type": "ORGANIZATION",
      "Text": "AnySpa",
```



```
    "BeginOffset": 63,
    "EndOffset": 69
  },
  {
    "Score": 0.8059805631637573,
    "Type": "LOCATION",
    "Text": "123 Main St, Anywhere",
    "BeginOffset": 71,
    "EndOffset": 92
  },
  {
    "Score": 0.998830258846283,
    "Type": "PERSON",
    "Text": "Alice",
    "BeginOffset": 114,
    "EndOffset": 119
  },
  {
    "Score": 0.997818112373352,
    "Type": "OTHER",
    "Text": "AnySpa@example.com",
    "BeginOffset": 123,
    "EndOffset": 138
  }
],
"File": "SampleText3.txt",
"Line": 0
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

## 示例 2：启动自定义实体检测作业

以下 `start-entities-detection-job` 示例为位于 `--input-data-config` 标签指定地址的所有文件启动异步自定义实体检测作业。在此示例中，此示例中的 S3 存储桶包含 `SampleFeedback1.txt`、`SampleFeedback2.txt`、和 `SampleFeedback3.txt`。实体识别器模型经过客户支持反馈的训练，可以识别设备名称。任务完成后，文件夹 `output`，将放置在 `--output-data-config` 标签指定的位置。该文件夹包含 `output.txt`，其中列出了在每个文本文件中检测到的所有命名实体，以及预训练模型对每个预测的置信度得分。Json 输出在每个文件的一行上打印，但是为了便于阅读，此处设置了格式。

```
aws comprehend start-entities-detection-job \  
  --job-name customentitiestest \  
  --entity-recognizer-arn "arn:aws:comprehend:us-west-2:111122223333:entity-recognizer/entityrecognizer" \  
  --language-code en \  
  --input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET/jobdata/" \  
  --output-data-config "S3Uri=s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/" \  
  --data-access-role-arn "arn:aws:iam::111122223333:role/service-role/AmazonComprehendServiceRole-I0role"
```

SampleFeedback1.txt 的内容 :

```
"I've been on the AnyPhone app have had issues for 24 hours when trying to pay bill. Cannot make payment. Sigh. | Oh man! Lets get that app up and running. DM me, and we can get to work!"
```

SampleFeedback2.txt 的内容 :

```
"Hi, I have a discrepancy with my new bill. Could we get it sorted out? A rep added stuff I didnt sign up for when I did my AnyPhone 10 upgrade. | We can absolutely get this sorted!"
```

SampleFeedback3.txt 的内容 :

```
"Is the by 1 get 1 free AnySmartPhone promo still going on? | Hi Christian! It ended yesterday, send us a DM if you have any questions and we can take a look at your options!"
```

输出 :

```
{  
  "JobId": "019ea9edac758806850fa8a79ff83021",  
  "JobArn": "arn:aws:comprehend:us-west-2:111122223333:entities-detection-job/019ea9edac758806850fa8a79ff83021",  
  "JobStatus": "SUBMITTED"  
}
```

为了便于阅读output.txt , 带有行缩进的内容 :

```
{
```

```
"Entities": [  
  {  
    "BeginOffset": 17,  
    "EndOffset": 25,  
    "Score": 0.9999728210205924,  
    "Text": "AnyPhone",  
    "Type": "DEVICE"  
  }  
],  
"File": "SampleFeedback1.txt",  
"Line": 0  
}  
{  
"Entities": [  
  {  
    "BeginOffset": 123,  
    "EndOffset": 133,  
    "Score": 0.9999892116761524,  
    "Text": "AnyPhone 10",  
    "Type": "DEVICE"  
  }  
],  
"File": "SampleFeedback2.txt",  
"Line": 0  
}  
{  
"Entities": [  
  {  
    "BeginOffset": 23,  
    "EndOffset": 35,  
    "Score": 0.9999971389852362,  
    "Text": "AnySmartPhone",  
    "Type": "DEVICE"  
  }  
],  
"File": "SampleFeedback3.txt",  
"Line": 0  
}
```

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的 [自定义实体识别](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartEntitiesDetectionJob](#) 中的。

## start-events-detection-job

以下代码示例演示了如何使用 start-events-detection-job。

### AWS CLI

#### 启动异步事件检测作业

以下 start-events-detection-job 示例为位于 --input-data-config 标签指定地址的所有文件启动异步事件检测作业。可能的目标事件类型包括 BANKRUPTCYEMPLOYMENT、CORPORATE\_ACQUISITION、INVESTMENT\_GENERAL、CORPORATE\_MERGER 和 STOCK\_SPLIT。本示例中的 S3 存储桶包含 SampleText1.txt、SampleText2.txt 和 SampleText3.txt。任务完成后 output，文件夹将放置在 --output-data-config 标签指定的位置。该文件夹包含 SampleText1.txt.out、SampleText2.txt.out 和 SampleText3.txt.out。JSON 输出按每个文件一行打印，但为了便于阅读，在此处进行了格式化。

```
aws comprehend start-events-detection-job \
  --job-name events-detection-1 \
  --input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET/EventsData" \
  --output-data-config "S3Uri=s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/" \
  --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-servicerole \
  --language-code en \
  --target-event-types "BANKRUPTCY" "EMPLOYMENT" "CORPORATE_ACQUISITION"
"CORPORATE_MERGER" "INVESTMENT_GENERAL"
```

SampleText1.txt 的内容：

```
"Company AnyCompany grew by increasing sales and through acquisitions. After
purchasing competing firms in 2020, AnyBusiness, a part of the AnyBusinessGroup,
gave Jane Does firm a going rate of one cent a gallon or forty-two cents a barrel."
```

SampleText2.txt 的内容：

```
"In 2021, AnyCompany officially purchased AnyBusiness for 100 billion dollars,
surprising and exciting the shareholders."
```

SampleText3.txt 的内容：

```
"In 2022, AnyCompany stock crashed 50. Eventually later that year they filed for bankruptcy."
```

输出：

```
{
  "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
  "JobArn": "arn:aws:comprehend:us-west-2:111122223333:events-detection-job/123456abcdeb0e11022f22a11EXAMPLE",
  "JobStatus": "SUBMITTED"
}
```

为了便于阅读SampleText1.txt.out，带有行缩进的内容：

```
{
  "Entities": [
    {
      "Mentions": [
        {
          "BeginOffset": 8,
          "EndOffset": 18,
          "Score": 0.99977,
          "Text": "AnyCompany",
          "Type": "ORGANIZATION",
          "GroupScore": 1
        },
        {
          "BeginOffset": 112,
          "EndOffset": 123,
          "Score": 0.999747,
          "Text": "AnyBusiness",
          "Type": "ORGANIZATION",
          "GroupScore": 0.979826
        },
        {
          "BeginOffset": 171,
          "EndOffset": 175,
          "Score": 0.999615,
          "Text": "firm",
          "Type": "ORGANIZATION",
          "GroupScore": 0.871647
        }
      ]
    }
  ]
}
```

```
]
},
{
  "Mentions": [
    {
      "BeginOffset": 97,
      "EndOffset": 102,
      "Score": 0.987687,
      "Text": "firms",
      "Type": "ORGANIZATION",
      "GroupScore": 1
    }
  ]
},
{
  "Mentions": [
    {
      "BeginOffset": 103,
      "EndOffset": 110,
      "Score": 0.999458,
      "Text": "in 2020",
      "Type": "DATE",
      "GroupScore": 1
    }
  ]
},
{
  "Mentions": [
    {
      "BeginOffset": 160,
      "EndOffset": 168,
      "Score": 0.999649,
      "Text": "John Doe",
      "Type": "PERSON",
      "GroupScore": 1
    }
  ]
}
],
"Events": [
  {
    "Type": "CORPORATE_ACQUISITION",
    "Arguments": [
      {
```

```
    "EntityIndex": 0,  
    "Role": "INVESTOR",  
    "Score": 0.99977  
  }  
],  
"Triggers": [  
  {  
    "BeginOffset": 56,  
    "EndOffset": 68,  
    "Score": 0.999967,  
    "Text": "acquisitions",  
    "Type": "CORPORATE_ACQUISITION",  
    "GroupScore": 1  
  }  
]  
},  
{  
  "Type": "CORPORATE_ACQUISITION",  
  "Arguments": [  
    {  
      "EntityIndex": 1,  
      "Role": "INVESTEES",  
      "Score": 0.987687  
    },  
    {  
      "EntityIndex": 2,  
      "Role": "DATE",  
      "Score": 0.999458  
    },  
    {  
      "EntityIndex": 3,  
      "Role": "INVESTOR",  
      "Score": 0.999649  
    }  
  ],  
  "Triggers": [  
    {  
      "BeginOffset": 76,  
      "EndOffset": 86,  
      "Score": 0.999973,  
      "Text": "purchasing",  
      "Type": "CORPORATE_ACQUISITION",  
      "GroupScore": 1  
    }  
  ]  
}
```

```
    ]
  }
],
"File": "SampleText1.txt",
"Line": 0
}
```

SampleText2.txt.out 的内容：

```
{
  "Entities": [
    {
      "Mentions": [
        {
          "BeginOffset": 0,
          "EndOffset": 7,
          "Score": 0.999473,
          "Text": "In 2021",
          "Type": "DATE",
          "GroupScore": 1
        }
      ]
    },
    {
      "Mentions": [
        {
          "BeginOffset": 9,
          "EndOffset": 19,
          "Score": 0.999636,
          "Text": "AnyCompany",
          "Type": "ORGANIZATION",
          "GroupScore": 1
        }
      ]
    },
    {
      "Mentions": [
        {
          "BeginOffset": 45,
          "EndOffset": 56,
          "Score": 0.999712,
          "Text": "AnyBusiness",
          "Type": "ORGANIZATION",

```



```
        "GroupScore": 1
      }
    ]
  },
  {
    "Mentions": [
      {
        "BeginOffset": 61,
        "EndOffset": 80,
        "Score": 0.998886,
        "Text": "100 billion dollars",
        "Type": "MONETARY_VALUE",
        "GroupScore": 1
      }
    ]
  }
],
"Events": [
  {
    "Type": "CORPORATE_ACQUISITION",
    "Arguments": [
      {
        "EntityIndex": 3,
        "Role": "AMOUNT",
        "Score": 0.998886
      },
      {
        "EntityIndex": 2,
        "Role": "INVESTEES",
        "Score": 0.999712
      },
      {
        "EntityIndex": 0,
        "Role": "DATE",
        "Score": 0.999473
      },
      {
        "EntityIndex": 1,
        "Role": "INVESTOR",
        "Score": 0.999636
      }
    ]
  },
  {
    "Triggers": [
      {
```

```
        "BeginOffset": 31,
        "EndOffset": 40,
        "Score": 0.99995,
        "Text": "purchased",
        "Type": "CORPORATE_ACQUISITION",
        "GroupScore": 1
      }
    ]
  }
],
"File": "SampleText2.txt",
"Line": 0
}
```

SampleText3.txt.out 的内容：

```
{
  "Entities": [
    {
      "Mentions": [
        {
          "BeginOffset": 9,
          "EndOffset": 19,
          "Score": 0.999774,
          "Text": "AnyCompany",
          "Type": "ORGANIZATION",
          "GroupScore": 1
        },
        {
          "BeginOffset": 66,
          "EndOffset": 70,
          "Score": 0.995717,
          "Text": "they",
          "Type": "ORGANIZATION",
          "GroupScore": 0.997626
        }
      ]
    },
    {
      "Mentions": [
        {
          "BeginOffset": 50,
          "EndOffset": 65,
```

```
        "Score": 0.999656,
        "Text": "later that year",
        "Type": "DATE",
        "GroupScore": 1
      }
    ]
  }
],
"Events": [
  {
    "Type": "BANKRUPTCY",
    "Arguments": [
      {
        "EntityIndex": 1,
        "Role": "DATE",
        "Score": 0.999656
      },
      {
        "EntityIndex": 0,
        "Role": "FILER",
        "Score": 0.995717
      }
    ],
    "Triggers": [
      {
        "BeginOffset": 81,
        "EndOffset": 91,
        "Score": 0.999936,
        "Text": "bankruptcy",
        "Type": "BANKRUPTCY",
        "GroupScore": 1
      }
    ]
  }
],
"File": "SampleText3.txt",
"Line": 0
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartEventsDetectionJob](#) 中的。

## start-flywheel-iteration

以下代码示例演示了如何使用 `start-flywheel-iteration`。

### AWS CLI

#### 开始飞轮迭代

以下 `start-flywheel-iteration` 示例开始飞轮迭代。此操作使用飞轮中的任何新数据集来训练新的模型版本。

```
aws comprehend start-flywheel-iteration \  
  --flywheel-arn arn:aws:comprehend:us-west-2:111122223333:flywheel/example-  
flywheel
```

#### 输出：

```
{  
  "FlywheelArn": "arn:aws:comprehend:us-west-2:111122223333:flywheel/example-  
flywheel",  
  "FlywheelIterationId": "12345123TEXAMPLE"  
}
```

有关更多信息，请参阅《亚马逊 Comprehend 开发者指南》中的 [Flywheel 概述](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartFlywheelIteration](#) 中的。

## start-key-phrases-detection-job

以下代码示例演示了如何使用 `start-key-phrases-detection-job`。

### AWS CLI

#### 开始关键短语检测作业

以下 `start-key-phrases-detection-job` 示例为位于 `--input-data-config` 标签指定地址的所有文件启动异步关键短语检测作业。本示例中的 S3 存储桶包含 `Sampletext1.txt`、`Sampletext2.txt`、和 `Sampletext3.txt`。作业完成后 `output`，文件夹将放置在 `--output-data-config` 标签指定的位置。该文件夹包含文件，`output.txt` 其中包含在每个文本文件中检测到的所有关键短语以及预训练模型对每个预测的置信度得分。Json 输出在每个文件的一行上打印，但是为了便于阅读，此处设置了格式。

```
aws comprehend start-key-phrases-detection-job \  
  --job-name keyphrasesanalysisitest1 \  
  --language-code en \  
  --input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET/" \  
  --output-data-config "S3Uri=s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/" \  
  --data-access-role-arn "arn:aws:iam::111122223333:role/service-role/  
AmazonComprehendServiceRole-example-role" \  
  --language-code en
```

Sampletext1.txt 的内容：

```
"Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC credit card  
account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due by July  
31st."
```

Sampletext2.txt 的内容：

```
"Dear Max, based on your autopay settings for your account Internet.org account, we  
will withdraw your payment on the due date from your bank account number XXXXXX1111  
with the routing number XXXXX0000. "
```

Sampletext3.txt 的内容：

```
"Jane, please submit any customer feedback from this weekend to Sunshine Spa, 123  
Main St, Anywhere and send comments to Alice at AnySpa@example.com."
```

输出：

```
{  
  "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
  "JobArn": "arn:aws:comprehend:us-west-2:111122223333:key-phrases-detection-  
job/123456abcdeb0e11022f22a11EXAMPLE",  
  "JobStatus": "SUBMITTED"  
}
```

为了便于output.txt阅读，带有行缩进的内容：

```
{  
  "File": "SampleText1.txt",  
  "KeyPhrases": [  
    {
```

```
"BeginOffset": 6,
"EndOffset": 15,
"Score": 0.9748965572679326,
"Text": "Zhang Wei"
},
{
"BeginOffset": 22,
"EndOffset": 26,
"Score": 0.9997344722354619,
"Text": "John"
},
{
"BeginOffset": 28,
"EndOffset": 62,
"Score": 0.9843791074032948,
"Text": "Your AnyCompany Financial Services"
},
{
"BeginOffset": 64,
"EndOffset": 107,
"Score": 0.8976122401721824,
"Text": "LLC credit card account 1111-XXXX-1111-XXXX"
},
{
"BeginOffset": 112,
"EndOffset": 129,
"Score": 0.9999612982629748,
"Text": "a minimum payment"
},
{
"BeginOffset": 133,
"EndOffset": 139,
"Score": 0.99975728947036,
"Text": "$24.53"
},
{
"BeginOffset": 155,
"EndOffset": 164,
"Score": 0.9940866241449973,
"Text": "July 31st"
}
],
"Line": 0
}
```

```
{
  "File": "SampleText2.txt",
  "KeyPhrases": [
    {
      "BeginOffset": 0,
      "EndOffset": 8,
      "Score": 0.9974021100118472,
      "Text": "Dear Max"
    },
    {
      "BeginOffset": 19,
      "EndOffset": 40,
      "Score": 0.9961120519515884,
      "Text": "your autopay settings"
    },
    {
      "BeginOffset": 45,
      "EndOffset": 78,
      "Score": 0.9980620070116009,
      "Text": "your account Internet.org account"
    },
    {
      "BeginOffset": 97,
      "EndOffset": 109,
      "Score": 0.999919660140754,
      "Text": "your payment"
    },
    {
      "BeginOffset": 113,
      "EndOffset": 125,
      "Score": 0.9998370719754205,
      "Text": "the due date"
    },
    {
      "BeginOffset": 131,
      "EndOffset": 166,
      "Score": 0.9955068678502509,
      "Text": "your bank account number XXXXXX1111"
    },
    {
      "BeginOffset": 172,
      "EndOffset": 200,
      "Score": 0.8653433315829526,
      "Text": "the routing number XXXXX0000"
    }
  ]
}
```

```
    }
  ],
  "Line": 0
}
{
  "File": "SampleText3.txt",
  "KeyPhrases": [
    {
      "BeginOffset": 0,
      "EndOffset": 4,
      "Score": 0.9142947833681668,
      "Text": "Jane"
    },
    {
      "BeginOffset": 20,
      "EndOffset": 41,
      "Score": 0.9984325676596763,
      "Text": "any customer feedback"
    },
    {
      "BeginOffset": 47,
      "EndOffset": 59,
      "Score": 0.9998782448150636,
      "Text": "this weekend"
    },
    {
      "BeginOffset": 63,
      "EndOffset": 75,
      "Score": 0.99866741830757,
      "Text": "Sunshine Spa"
    },
    {
      "BeginOffset": 77,
      "EndOffset": 88,
      "Score": 0.9695803485466054,
      "Text": "123 Main St"
    },
    {
      "BeginOffset": 108,
      "EndOffset": 116,
      "Score": 0.9997065928550928,
      "Text": "comments"
    }
  ]
}
```



```
    "BeginOffset": 120,
    "EndOffset": 125,
    "Score": 0.9993466833825161,
    "Text": "Alice"
  },
  {
    "BeginOffset": 129,
    "EndOffset": 144,
    "Score": 0.9654563612885667,
    "Text": "AnySpa@example.com"
  }
],
"Line": 0
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartKeyPhrasesDetectionJob](#) 中的。

## start-pii-entities-detection-job

以下代码示例演示了如何使用 `start-pii-entities-detection-job`。

### AWS CLI

#### 启动异步 PII 检测作业

以下 `start-pii-entities-detection-job` 示例为位于 `--input-data-config` 标签指定地址的所有文件启动异步个人信息 (PII) 实体检测作业。本示例中的 S3 存储桶包含 `Sampletext1.txt`、`Sampletext2.txt`、和 `Sampletext3.txt`。任务完成后 `output`，文件夹将放置在 `--output-data-config` 标签指定的位置。该文件夹包含 `SampleText1.txt.out`、`SampleText2.txt.out`、和 `SampleText3.txt.out` 它们列出了每个文本文件中的命名实体。Json 输出在每个文件的一行上打印，但是为了便于阅读，此处设置了格式。

```
aws comprehend start-pii-entities-detection-job \
  --job-name entities_test \
  --language-code en \
  --input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET/" \
  --output-data-config "S3Uri=s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/" \
```

```
--data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role \
--language-code en \
--mode ONLY_OFFSETS
```

Sampletext1.txt 的内容 :

```
"Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC credit card
account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due by July
31st."
```

Sampletext2.txt 的内容 :

```
"Dear Max, based on your autopay settings for your account Internet.org account, we
will withdraw your payment on the due date from your bank account number XXXXXX1111
with the routing number XXXXX0000. "
```

Sampletext3.txt 的内容 :

```
"Jane, please submit any customer feedback from this weekend to Sunshine Spa, 123
Main St, Anywhere and send comments to Alice at AnySpa@example.com."
```

输出 :

```
{
  "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
  "JobArn": "arn:aws:comprehend:us-west-2:111122223333:pii-entities-detection-
job/123456abcdeb0e11022f22a11EXAMPLE",
  "JobStatus": "SUBMITTED"
}
```

为了便于阅读SampleText1.txt.out , 带有行缩进的内容 :

```
{
  "Entities": [
    {
      "BeginOffset": 6,
      "EndOffset": 15,
      "Type": "NAME",
      "Score": 0.9998490510222595
    }
  ]
}
```

```
    },
    {
      "BeginOffset": 22,
      "EndOffset": 26,
      "Type": "NAME",
      "Score": 0.9998937958019426
    },
    {
      "BeginOffset": 88,
      "EndOffset": 107,
      "Type": "CREDIT_DEBIT_NUMBER",
      "Score": 0.9554297245278491
    },
    {
      "BeginOffset": 155,
      "EndOffset": 164,
      "Type": "DATE_TIME",
      "Score": 0.9999720462925257
    }
  ],
  "File": "SampleText1.txt",
  "Line": 0
}
```

为了便于阅读SampleText2.txt.out，带有行缩进的内容：

```
{
  "Entities": [
    {
      "BeginOffset": 5,
      "EndOffset": 8,
      "Type": "NAME",
      "Score": 0.9994390774924007
    },
    {
      "BeginOffset": 58,
      "EndOffset": 70,
      "Type": "URL",
      "Score": 0.9999958276922101
    },
    {
      "BeginOffset": 156,
      "EndOffset": 166,
```

```
    "Type": "BANK_ACCOUNT_NUMBER",
    "Score": 0.9999721058045592
  },
  {
    "BeginOffset": 191,
    "EndOffset": 200,
    "Type": "BANK_ROUTING",
    "Score": 0.9998968945989909
  }
],
"File": "SampleText2.txt",
"Line": 0
}
```

为了便于阅读SampleText3.txt.out，带有行缩进的内容：

```
{
  "Entities": [
    {
      "BeginOffset": 0,
      "EndOffset": 4,
      "Type": "NAME",
      "Score": 0.999949934606805
    },
    {
      "BeginOffset": 77,
      "EndOffset": 88,
      "Type": "ADDRESS",
      "Score": 0.9999035300466904
    },
    {
      "BeginOffset": 120,
      "EndOffset": 125,
      "Type": "NAME",
      "Score": 0.9998203838716296
    },
    {
      "BeginOffset": 129,
      "EndOffset": 144,
      "Type": "EMAIL",
      "Score": 0.9998313473105228
    }
  ],
}
```

```
"File": "SampleText3.txt",  
"Line": 0  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartPiiEntitiesDetectionJob](#)中的。

## start-sentiment-detection-job

以下代码示例演示了如何使用 start-sentiment-detection-job。

### AWS CLI

#### 启动异步情绪分析作业

以下start-sentiment-detection-job示例为位于--input-data-config标签指定地址的所有文件启动异步情绪分析检测作业。本示例中的 S3 存储桶文件夹包含SampleMovieReview1.txtSampleMovieReview2.txt、和SampleMovieReview3.txt。任务完成后output，文件夹将放置在--output-data-config标签指定的位置。该文件夹包含该文件output.txt，其中包含每个文本文件的主要情绪以及预训练模型对每个预测的置信度得分。Json 输出在每个文件的一行上打印，但是为了便于阅读，此处设置了格式。

```
aws comprehend start-sentiment-detection-job \  
  --job-name example-sentiment-detection-job \  
  --language-code en \  
  --input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET/MovieData" \  
  --output-data-config "S3Uri=s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/" \  
  --data-access-role-arn arn:aws:iam::111122223333:role/service-role/  
AmazonComprehendServiceRole-example-role
```

SampleMovieReview1.txt 的内容：

```
"The film, AnyMovie2, is fairly predictable and just okay."
```

SampleMovieReview2.txt 的内容：

```
"AnyMovie2 is the essential sci-fi film that I grew up watching when I was a kid. I  
highly recommend this movie."
```

### SampleMovieReview3.txt 的内容：

```
"Don't get fooled by the 'awards' for AnyMovie2. All parts of the film were poorly
stolen from other modern directors."
```

### 输出：

```
{
  "JobId": "0b5001e25f62ebb40631a9a1a7fde7b3",
  "JobArn": "arn:aws:comprehend:us-west-2:111122223333:sentiment-detection-
job/0b5001e25f62ebb40631a9a1a7fde7b3",
  "JobStatus": "SUBMITTED"
}
```

### 为了便于阅读output.txt，带有缩进行的内容：

```
{
  "File": "SampleMovieReview1.txt",
  "Line": 0,
  "Sentiment": "MIXED",
  "SentimentScore": {
    "Mixed": 0.6591159105300903,
    "Negative": 0.26492202281951904,
    "Neutral": 0.035430654883384705,
    "Positive": 0.04053137078881264
  }
}
{
  "File": "SampleMovieReview2.txt",
  "Line": 0,
  "Sentiment": "POSITIVE",
  "SentimentScore": {
    "Mixed": 0.000008718466233403888,
    "Negative": 0.00006134175055194646,
    "Neutral": 0.0002941041602753103,
    "Positive": 0.9996358156204224
  }
}
{
  "File": "SampleMovieReview3.txt",
  "Line": 0,
  "Sentiment": "NEGATIVE",
```

```

    "SentimentScore": {
      "Mixed": 0.004146667663007975,
      "Negative": 0.9645107984542847,
      "Neutral": 0.016559595242142677,
      "Positive": 0.014782938174903393
    }
  }
}

```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartSentimentDetectionJob](#) 中的。

## start-targeted-sentiment-detection-job

以下代码示例演示了如何使用 start-targeted-sentiment-detection-job。

### AWS CLI

#### 启动异步定向情绪分析作业

以下 start-targeted-sentiment-detection-job 示例为位于 --input-data-config 标签指定地址的所有文件启动异步定向情绪分析检测作业。本示例中的 S3 存储桶文件夹包含 SampleMovieReview1.txt、SampleMovieReview2.txt、和 SampleMovieReview3.txt。任务完成后 output.tar.gz，将放置在 --output-data-config 标签指定的位置。output.tar.gz 包含文件 SampleMovieReview1.txt.out、SampleMovieReview2.txt.out、SampleMovieReview3.txt.out 和，每个文件都包含单个输入文本文件的所有命名实体和关联情绪。

```

aws comprehend start-targeted-sentiment-detection-job \
  --job-name targeted_movie_review_analysis1 \
  --language-code en \
  --input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET/MovieData" \
  --output-data-config "S3Uri=s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/" \
  --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role

```

SampleMovieReview1.txt 的内容：

```
"The film, AnyMovie, is fairly predictable and just okay."
```

SampleMovieReview2.txt 的内容：

```
"AnyMovie is the essential sci-fi film that I grew up watching when I was a kid. I highly recommend this movie."
```

SampleMovieReview3.txt 的内容：

```
"Don't get fooled by the 'awards' for AnyMovie. All parts of the film were poorly stolen from other modern directors."
```

输出：

```
{
  "JobId": "0b5001e25f62ebb40631a9a1a7fde7b3",
  "JobArn": "arn:aws:comprehend:us-west-2:111122223333:targeted-sentiment-detection-job/0b5001e25f62ebb40631a9a1a7fde7b3",
  "JobStatus": "SUBMITTED"
}
```

为了便于阅读SampleMovieReview1.txt.out，带有行缩进的内容：

```
{
  "Entities": [
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 4,
          "EndOffset": 8,
          "Score": 0.994972,
          "GroupScore": 1,
          "Text": "film",
          "Type": "MOVIE",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Mixed": 0,
              "Negative": 0,
              "Neutral": 1,
              "Positive": 0
            }
          }
        }
      ]
    }
  ]
}
```



```

    }
  }
}
],
{
  "DescriptiveMentionIndex": [
    0
  ],
  "Mentions": [
    {
      "BeginOffset": 10,
      "EndOffset": 18,
      "Score": 0.631368,
      "GroupScore": 1,
      "Text": "AnyMovie",
      "Type": "ORGANIZATION",
      "MentionSentiment": {
        "Sentiment": "POSITIVE",
        "SentimentScore": {
          "Mixed": 0.001729,
          "Negative": 0.000001,
          "Neutral": 0.000318,
          "Positive": 0.997952
        }
      }
    }
  ]
}
],
"File": "SampleMovieReview1.txt",
"Line": 0
}

```

为了便于阅读，SampleMovieReview2.txt.out行缩进的内容：

```

{
  "Entities": [
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [

```

```
{
  "BeginOffset": 0,
  "EndOffset": 8,
  "Score": 0.854024,
  "GroupScore": 1,
  "Text": "AnyMovie",
  "Type": "MOVIE",
  "MentionSentiment": {
    "Sentiment": "POSITIVE",
    "SentimentScore": {
      "Mixed": 0,
      "Negative": 0,
      "Neutral": 0.000007,
      "Positive": 0.999993
    }
  }
},
{
  "BeginOffset": 104,
  "EndOffset": 109,
  "Score": 0.999129,
  "GroupScore": 0.502937,
  "Text": "movie",
  "Type": "MOVIE",
  "MentionSentiment": {
    "Sentiment": "POSITIVE",
    "SentimentScore": {
      "Mixed": 0,
      "Negative": 0,
      "Neutral": 0,
      "Positive": 1
    }
  }
},
{
  "BeginOffset": 33,
  "EndOffset": 37,
  "Score": 0.999823,
  "GroupScore": 0.999252,
  "Text": "film",
  "Type": "MOVIE",
  "MentionSentiment": {
    "Sentiment": "POSITIVE",
    "SentimentScore": {
```

```
        "Mixed": 0,
        "Negative": 0,
        "Neutral": 0.000001,
        "Positive": 0.999999
      }
    }
  ],
},
{
  "DescriptiveMentionIndex": [
    0,
    1,
    2
  ],
  "Mentions": [
    {
      "BeginOffset": 43,
      "EndOffset": 44,
      "Score": 0.999997,
      "GroupScore": 1,
      "Text": "I",
      "Type": "PERSON",
      "MentionSentiment": {
        "Sentiment": "NEUTRAL",
        "SentimentScore": {
          "Mixed": 0,
          "Negative": 0,
          "Neutral": 1,
          "Positive": 0
        }
      }
    }
  ],
  {
    "BeginOffset": 80,
    "EndOffset": 81,
    "Score": 0.999996,
    "GroupScore": 0.52523,
    "Text": "I",
    "Type": "PERSON",
    "MentionSentiment": {
      "Sentiment": "NEUTRAL",
      "SentimentScore": {
        "Mixed": 0,
```

```
        "Negative": 0,
        "Neutral": 1,
        "Positive": 0
      }
    }
  },
  {
    "BeginOffset": 67,
    "EndOffset": 68,
    "Score": 0.999994,
    "GroupScore": 0.999499,
    "Text": "I",
    "Type": "PERSON",
    "MentionSentiment": {
      "Sentiment": "NEUTRAL",
      "SentimentScore": {
        "Mixed": 0,
        "Negative": 0,
        "Neutral": 1,
        "Positive": 0
      }
    }
  }
]
},
{
  "DescriptiveMentionIndex": [
    0
  ],
  "Mentions": [
    {
      "BeginOffset": 75,
      "EndOffset": 78,
      "Score": 0.999978,
      "GroupScore": 1,
      "Text": "kid",
      "Type": "PERSON",
      "MentionSentiment": {
        "Sentiment": "NEUTRAL",
        "SentimentScore": {
          "Mixed": 0,
          "Negative": 0,
          "Neutral": 1,
          "Positive": 0
        }
      }
    }
  ]
}
```

```
        }
      }
    }
  ]
}
],
"File": "SampleMovieReview2.txt",
"Line": 0
}
```

为了便于SampleMovieReview3.txt.out阅读，带有行缩进的内容：

```
{
  "Entities": [
    {
      "DescriptiveMentionIndex": [
        1
      ],
      "Mentions": [
        {
          "BeginOffset": 64,
          "EndOffset": 68,
          "Score": 0.992953,
          "GroupScore": 0.999814,
          "Text": "film",
          "Type": "MOVIE",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Mixed": 0.000004,
              "Negative": 0.010425,
              "Neutral": 0.989543,
              "Positive": 0.000027
            }
          }
        }
      ],
    },
    {
      "BeginOffset": 37,
      "EndOffset": 45,
      "Score": 0.999782,
      "GroupScore": 1,
      "Text": "AnyMovie",
      "Type": "ORGANIZATION",
    }
  ]
}
```

```
    "MentionSentiment": {
      "Sentiment": "POSITIVE",
      "SentimentScore": {
        "Mixed": 0.000095,
        "Negative": 0.039847,
        "Neutral": 0.000673,
        "Positive": 0.959384
      }
    }
  },
  {
    "DescriptiveMentionIndex": [
      0
    ],
    "Mentions": [
      {
        "BeginOffset": 47,
        "EndOffset": 50,
        "Score": 0.999991,
        "GroupScore": 1,
        "Text": "All",
        "Type": "QUANTITY",
        "MentionSentiment": {
          "Sentiment": "NEUTRAL",
          "SentimentScore": {
            "Mixed": 0.000001,
            "Negative": 0.000001,
            "Neutral": 0.999998,
            "Positive": 0
          }
        }
      }
    ]
  },
  {
    "DescriptiveMentionIndex": [
      0
    ],
    "Mentions": [
      {
        "BeginOffset": 106,
        "EndOffset": 115,
```

```
        "Score": 0.542083,
        "GroupScore": 1,
        "Text": "directors",
        "Type": "PERSON",
        "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
                "Mixed": 0,
                "Negative": 0,
                "Neutral": 1,
                "Positive": 0
            }
        }
    ]
}
],
"File": "SampleMovieReview3.txt",
"Line": 0
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartTargetedSentimentDetectionJob](#) 中的。

## start-topics-detection-job

以下代码示例演示了如何使用 start-topics-detection-job。

### AWS CLI

#### 启动主题检测分析作业

以下 start-topics-detection-job 示例为位于 --input-data-config 标签指定地址的所有文件启动异步主题检测作业。作业完成后，文件夹 output 将放置在 --output-data-config 标签指定的位置。output 包含 topic-terms.csv 和 doc-topics.csv。第一个输出文件 topic-terms.csv 是集合中的主题列表。对于每个主题，默认情况下，该列表按权重排列主题列出根据其的热门术语。第二个文件 doc-topics.csv 列出了与主题相关的文档以及与该主题相关的文档比例。

```
aws comprehend start-topics-detection-job \
```

```
--job-name example_topics_detection_job \  
--language-code en \  
--input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET/" \  
--output-data-config "S3Uri=s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/" \  
--data-access-role-arn arn:aws:iam::111122223333:role/service-role/  
AmazonComprehendServiceRole-example-role \  
--language-code en
```

输出：

```
{  
  "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
  "JobArn": "arn:aws:comprehend:us-west-2:111122223333:key-phrases-detection-  
job/123456abcdeb0e11022f22a11EXAMPLE",  
  "JobStatus": "SUBMITTED"  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[主题建模](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartTopicsDetectionJob](#)中的。

## stop-dominant-language-detection-job

以下代码示例演示了如何使用 stop-dominant-language-detection-job。

### AWS CLI

停止异步主导语言检测作业

以下stop-dominant-language-detection-job示例停止正在进行的异步主导语言检测作业。如果当前作业状态为IN\_PROGRESS，则该作业被标记为终止并进入STOP\_REQUESTED状态。如果任务在可以停止之前完成，则会将其置于COMPLETED状态。

```
aws comprehend stop-dominant-language-detection-job \  
--job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{  
  "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
  "JobStatus": "STOP_REQUESTED"
```



```
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopDominantLanguageDetectionJob](#)中的。

## stop-entities-detection-job

以下代码示例演示了如何使用 stop-entities-detection-job。

### AWS CLI

#### 停止异步实体检测作业

以下stop-entities-detection-job示例停止正在进行的异步实体检测作业。如果当前作业状态为IN\_PROGRESS，则该作业被标记为终止并进入STOP\_REQUESTED状态。如果作业在可以停止之前就完成了，则会将其置于COMPLETED状态。

```
aws comprehend stop-entities-detection-job \  
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{  
  "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
  "JobStatus": "STOP_REQUESTED"  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopEntitiesDetectionJob](#)中的。

## stop-events-detection-job

以下代码示例演示了如何使用 stop-events-detection-job。

### AWS CLI

#### 停止异步事件检测作业

以下`stop-events-detection-job`示例停止正在进行的异步事件检测作业。如果当前作业状态为`IN_PROGRESS`，则该作业被标记为终止并进入`STOP_REQUESTED`状态。如果作业在可以停止之前就完成了，则会将其置于`COMPLETED`状态。

```
aws comprehend stop-events-detection-job \  
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{  
  "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
  "JobStatus": "STOP_REQUESTED"  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopEventsDetectionJob](#)中的。

## stop-key-phrases-detection-job

以下代码示例演示了如何使用 `stop-key-phrases-detection-job`。

### AWS CLI

停止异步关键短语检测作业

以下`stop-key-phrases-detection-job`示例停止正在进行的异步关键短语检测作业。如果当前作业状态为`IN_PROGRESS`，则该作业被标记为终止并进入`STOP_REQUESTED`状态。如果作业在可以停止之前就完成了，则会将其置于`COMPLETED`状态。

```
aws comprehend stop-key-phrases-detection-job \  
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{  
  "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
  "JobStatus": "STOP_REQUESTED"  
}
```

```
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopKeyPhrasesDetectionJob](#)中的。

## stop-pii-entities-detection-job

以下代码示例演示了如何使用 stop-pii-entities-detection-job。

### AWS CLI

停止异步 pii 实体检测作业

以下stop-pii-entities-detection-job示例停止正在进行的异步 pii 实体检测作业。如果当前作业状态为IN\_PROGRESS，则该作业被标记为终止并进入STOP\_REQUESTED状态。如果作业在可以停止之前就完成了，则会将其置于COMPLETED状态。

```
aws comprehend stop-pii-entities-detection-job \  
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{  
  "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
  "JobStatus": "STOP_REQUESTED"  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopPiiEntitiesDetectionJob](#)中的。

## stop-sentiment-detection-job

以下代码示例演示了如何使用 stop-sentiment-detection-job。

### AWS CLI

停止异步情绪检测作业

以下`stop-sentiment-detection-job`示例停止正在进行的异步情绪检测作业。如果当前作业状态为`IN_PROGRESS`，则该作业被标记为终止并进入`STOP_REQUESTED`状态。如果作业在可以停止之前就完成了，则会将其置于`COMPLETED`状态。

```
aws comprehend stop-sentiment-detection-job \  
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{  
  "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
  "JobStatus": "STOP_REQUESTED"  
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopSentimentDetectionJob](#)中的。

## stop-targeted-sentiment-detection-job

以下代码示例演示了如何使用 `stop-targeted-sentiment-detection-job`。

### AWS CLI

停止异步定向情绪检测作业

以下`stop-targeted-sentiment-detection-job`示例停止正在进行的异步定向情绪检测作业。如果当前作业状态为`IN_PROGRESS`，则该作业被标记为终止并进入`STOP_REQUESTED`状态。如果作业在可以停止之前就完成了，则会将其置于`COMPLETED`状态。

```
aws comprehend stop-targeted-sentiment-detection-job \  
  --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

输出：

```
{  
  "JobId": "123456abcdeb0e11022f22a11EXAMPLE",  
  "JobStatus": "STOP_REQUESTED"  
}
```

```
}
```

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的 [Amazon Comprehend 洞察的异步分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopTargetedSentimentDetectionJob](#)中的。

## stop-training-document-classifier

以下代码示例演示了如何使用 stop-training-document-classifier。

### AWS CLI

停止训练文档分类器模型

以下stop-training-document-classifier示例在文档分类器模型的训练过程中停止训练。

```
aws comprehend stop-training-document-classifier
  --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-
  classifier/example-classifier
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[创建和管理自定义模型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopTrainingDocumentClassifier](#)中的。

## stop-training-entity-recognizer

以下代码示例演示了如何使用 stop-training-entity-recognizer。

### AWS CLI

停止训练实体识别器模型

以下stop-training-entity-recognizer示例在实体识别器模型的训练过程中停止训练。

```
aws comprehend stop-training-entity-recognizer
  --entity-recognizer-arn "arn:aws:comprehend:us-west-2:111122223333:entity-
  recognizer/examplerrecognizer1"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[创建和管理自定义模型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopTrainingEntityRecognizer](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

示例 1：为资源添加标签

以下tag-resource示例向 Amazon Comprehend 资源添加了一个标签。

```
aws comprehend tag-resource \  
  --resource-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/  
example-classifier/version/1 \  
  --tags Key=Location,Value=Seattle
```

此命令没有输出。

有关更多信息，请参阅 Amazon Comprehend [开发者指南中的为资源添加标签](#)。

示例 2：为资源添加多个标签

以下tag-resource示例向 Amazon Comprehend 资源添加了多个标签。

```
aws comprehend tag-resource \  
  --resource-arn "arn:aws:comprehend:us-west-2:111122223333:document-classifier/  
example-classifier/version/1" \  
  --tags Key=location,Value=Seattle Key=Department,Value=Finance
```

此命令没有输出。

有关更多信息，请参阅 Amazon Comprehend [开发者指南中的为资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

## AWS CLI

### 示例 1：从资源中移除单个标签

以下 `untag-resource` 示例从 Amazon Comprehend 资源中移除单个标签。

```
aws comprehend untag-resource \  
  --resource-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/  
example-classifier/version/1  
  --tag-keys Location
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Comprehend [开发者指南中的为资源添加标签](#)。

### 示例 2：从资源中移除多个标签

以下 `untag-resource` 示例从 Amazon Comprehend 资源中移除多个标签。

```
aws comprehend untag-resource \  
  --resource-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/  
example-classifier/version/1  
  --tag-keys Location Department
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Comprehend [开发者指南中的为资源添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-endpoint

以下代码示例演示了如何使用 `update-endpoint`。

## AWS CLI

### 示例 1：更新终端节点的推理单元

以下 `update-endpoint` 示例更新了有关终端节点的信息。在此示例中，推理单元的数量增加了。

```
aws comprehend update-endpoint \  
  --resource-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/  
example-classifier/version/1  
  --tag-keys Location Department
```

```
--endpoint-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier-  
endpoint/example-classifier-endpoint  
--desired-inference-units 2
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[管理 Amazon Comprehend 端点](#)。

### 示例 2：更新端点的活动模型

以下update-endpoint示例更新了有关终端节点的信息。在此示例中，活动模型已更改。

```
aws comprehend update-endpoint \  
  --endpoint-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier-  
endpoint/example-classifier-endpoint  
  --active-model-arn arn:aws:comprehend:us-west-2:111122223333:document-  
classifier/example-classifier-new
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Comprehend 开发人员指南》中的[管理 Amazon Comprehend 端点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateEndpoint](#)中的。

## update-flywheel

以下代码示例演示了如何使用 update-flywheel。

### AWS CLI

#### 更新飞轮配置

以下update-flywheel示例更新了飞轮配置。在此示例中，飞轮的活动模型已更新。

```
aws comprehend update-flywheel \  
  --flywheel-arn arn:aws:comprehend:us-west-2:111122223333:flywheel/example-  
flywheel-1 \  
  --active-model-arn arn:aws:comprehend:us-west-2:111122223333:document-  
classifier/example-classifier/version/new-example-classifier-model
```



输出：

```
{
  "FlywheelProperties": {
    "FlywheelArn": "arn:aws:comprehend:us-west-2:111122223333:flywheel/flywheel-
entity",
    "ActiveModelArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/example-classifier/version/new-example-classifier-model",
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role",
    "TaskConfig": {
      "LanguageCode": "en",
      "DocumentClassificationConfig": {
        "Mode": "MULTI_CLASS"
      }
    },
    "DataLakeS3Uri": "s3://DOC-EXAMPLE-BUCKET/flywheel-entity/
schemaVersion=1/20230616T200543Z/",
    "DataSecurityConfig": {},
    "Status": "ACTIVE",
    "ModelType": "DOCUMENT_CLASSIFIER",
    "CreationTime": "2023-06-16T20:05:43.242000+00:00",
    "LastModifiedTime": "2023-06-19T04:00:43.027000+00:00",
    "LatestFlywheelIteration": "20230619T040032Z"
  }
}
```

有关更多信息，请参阅《亚马逊 Comprehend 开发者指南》中的 [Flywheel 概述](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateFlywheel](#) 中的。

## 使用 Amazon Comprehend Medical 的示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon Comprehend Medical 一起使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### describe-entities-detection-v2-job

以下代码示例演示了如何使用 describe-entities-detection-v2-job。

#### AWS CLI

##### 描述实体检测作业

以下describe-entities-detection-v2-job示例显示了与异步实体检测作业关联的属性。

```
aws comprehendmedical describe-entities-detection-v2-job \
  --job-id "ab9887877365fe70299089371c043b96"
```

##### 输出：

```
{
  "ComprehendMedicalAsyncJobProperties": {
    "JobId": "ab9887877365fe70299089371c043b96",
    "JobStatus": "COMPLETED",
    "SubmitTime": "2020-03-18T21:20:15.614000+00:00",
    "EndTime": "2020-03-18T21:27:07.350000+00:00",
    "ExpirationTime": "2020-07-16T21:20:15+00:00",
    "InputDataConfig": {
      "S3Bucket": "comp-med-input",
      "S3Key": ""
    },
    "OutputDataConfig": {
      "S3Bucket": "comp-med-output",
      "S3Key": "867139942017-EntitiesDetection-
ab9887877365fe70299089371c043b96/"
    },
    "LanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::867139942017:role/
ComprehendMedicalBatchProcessingRole",
    "ModelVersion": "DetectEntitiesModelV20190930"
  }
}
```

```
}
```

有关更多信息，请参阅亚马逊 Comprehend Medical 开发者指南中的[批处理 API](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [DescribeEntitiesDetectionV2Job](#)。

## describe-icd10-cm-inference-job

以下代码示例演示了如何使用 describe-icd10-cm-inference-job。

### AWS CLI

#### 描述 ICD-10-CM 推理作业

以下 describe-icd10-cm-inference-job 示例描述了具有指定作业 ID 的请求推理作业的属性。

```
aws comprehendmedical describe-icd10-cm-inference-job \
  --job-id "5780034166536cdb52ffa3295a1b00a7"
```

输出：

```
{
  "ComprehendMedicalAsyncJobProperties": {
    "JobId": "5780034166536cdb52ffa3295a1b00a7",
    "JobStatus": "COMPLETED",
    "SubmitTime": "2020-05-18T21:20:15.614000+00:00",
    "EndTime": "2020-05-18T21:27:07.350000+00:00",
    "ExpirationTime": "2020-09-16T21:20:15+00:00",
    "InputDataConfig": {
      "S3Bucket": "comp-med-input",
      "S3Key": "AKIAIOSFODNN7EXAMPLE"
    },
    "OutputDataConfig": {
      "S3Bucket": "comp-med-output",
      "S3Key": "AKIAIOSFODNN7EXAMPLE"
    },
    "LanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::867139942017:role/ComprehendMedicalBatchProcessingRole",
    "ModelVersion": "0.1.0"
  }
}
```

```
}
```

有关更多信息，请参阅 Amazon Comprehend Medical [开发者指南中的本体链接批量分析](#)。

- 有关 API 的详细信息，请参阅 CmlInferenceJob 《AWS CLI 命令参考》中的 [DescribeCdl10](#)。

## describe-phi-detection-job

以下代码示例演示了如何使用 describe-phi-detection-job。

### AWS CLI

#### 描述 PHI 检测作业

以下 describe-phi-detection-job 示例显示了与异步受保护的健康信息 (PHI) 检测作业关联的属性。

```
aws comprehendmedical describe-phi-detection-job \  
  --job-id "4750034166536cdb52ffa3295a1b00a3"
```

输出：

```
{  
  "ComprehendMedicalAsyncJobProperties": {  
    "JobId": "4750034166536cdb52ffa3295a1b00a3",  
    "JobStatus": "COMPLETED",  
    "SubmitTime": "2020-03-19T20:38:37.594000+00:00",  
    "EndTime": "2020-03-19T20:45:07.894000+00:00",  
    "ExpirationTime": "2020-07-17T20:38:37+00:00",  
    "InputDataConfig": {  
      "S3Bucket": "comp-med-input",  
      "S3Key": ""  
    },  
    "OutputDataConfig": {  
      "S3Bucket": "comp-med-output",  
      "S3Key": "867139942017-PHIDetection-4750034166536cdb52ffa3295a1b00a3/"  
    },  
    "LanguageCode": "en",  
    "DataAccessRoleArn": "arn:aws:iam::867139942017:role/  
ComprehendMedicalBatchProcessingRole",  
    "ModelVersion": "PHIModelV20190903"  
  }  
}
```

```
}
```

有关更多信息，请参阅亚马逊 Comprehend Medical 开发者指南中的[批处理 API](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePhiDetectionJob](#)中的。

## describe-rx-norm-inference-job

以下代码示例演示了如何使用 describe-rx-norm-inference-job。

### AWS CLI

描述 RxNorm 推理工作

以下describe-rx-norm-inference-job示例描述了具有指定作业 ID 的请求推理作业的属性。

```
aws comprehendmedical describe-rx-norm-inference-job \
  --job-id "eg8199877365fc70299089371c043b96"
```

输出：

```
{
  "ComprehendMedicalAsyncJobProperties": {
    "JobId": "g8199877365fc70299089371c043b96",
    "JobStatus": "COMPLETED",
    "SubmitTime": "2020-05-18T21:20:15.614000+00:00",
    "EndTime": "2020-05-18T21:27:07.350000+00:00",
    "ExpirationTime": "2020-09-16T21:20:15+00:00",
    "InputDataConfig": {
      "S3Bucket": "comp-med-input",
      "S3Key": "AKIAIOSFODNN7EXAMPLE"
    },
    "OutputDataConfig": {
      "S3Bucket": "comp-med-output",
      "S3Key": "AKIAIOSFODNN7EXAMPLE"
    },
    "LanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::867139942017:role/ComprehendMedicalBatchProcessingRole",
    "ModelVersion": "0.0.0"
  }
}
```

```
}
```

有关更多信息，请参阅 Amazon Comprehend Medical [开发者指南中的本体链接批量分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeRxNormInferenceJob](#) 中的。

## describe-snomedct-inference-job

以下代码示例演示了如何使用 `describe-snomedct-inference-job`。

### AWS CLI

#### 描述 SNOMED CT 推理作业

以下 `describe-snomedct-inference-job` 示例描述了具有指定作业 ID 的请求推理作业的属性。

```
aws comprehendmedical describe-snomedct-inference-job \  
  --job-id "2630034166536cdb52ffa3295a1b00a7"
```

输出：

```
{  
  "ComprehendMedicalAsyncJobProperties": {  
    "JobId": "2630034166536cdb52ffa3295a1b00a7",  
    "JobStatus": "COMPLETED",  
    "SubmitTime": "2021-12-18T21:20:15.614000+00:00",  
    "EndTime": "2021-12-18T21:27:07.350000+00:00",  
    "ExpirationTime": "2022-05-16T21:20:15+00:00",  
    "InputDataConfig": {  
      "S3Bucket": "comp-med-input",  
      "S3Key": "AKIAIOSFODNN7EXAMPLE"  
    },  
    "OutputDataConfig": {  
      "S3Bucket": "comp-med-output",  
      "S3Key": "AKIAIOSFODNN7EXAMPLE"  
    },  
    "LanguageCode": "en",  
    "DataAccessRoleArn": "arn:aws:iam::867139942017:role/  
ComprehendMedicalBatchProcessingRole",  
    "ModelVersion": "0.1.0"  
  }  
}
```

```
}
```

有关更多信息，请参阅 Amazon Comprehend Medical [开发者指南中的本体链接批量分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeSnomedctInferenceJob](#) 中的。

## detect-entities-v2

以下代码示例演示了如何使用 detect-entities-v2。

### AWS CLI

示例 1：直接从文本中检测实体

以下 detect-entities-v2 示例显示了检测到的实体，并直接从输入文本中根据类型对其进行标记。

```
aws comprehendmedical detect-entities-v2 \  
  --text "Sleeping trouble on present dosage of Clonidine. Severe rash on face and  
  leg, slightly itchy."
```

输出：

```
{  
  "Id": 0,  
  "BeginOffset": 38,  
  "EndOffset": 47,  
  "Score": 0.9942955374717712,  
  "Text": "Clonidine",  
  "Category": "MEDICATION",  
  "Type": "GENERIC_NAME",  
  "Traits": []  
}
```

有关更多信息，请参阅亚马逊 Comprehend Medical 开发者指南中的 [检测实体版本 2](#)。

示例 2：检测文件路径中的实体

以下 detect-entities-v2 示例显示了检测到的实体，并根据文件路径中的类型对其进行标记。

```
aws comprehendmedical detect-entities-v2 \  
  --text file://medical_entities.txt
```

medical\_entities.txt 的内容：

```
{
  "Sleeping trouble on present dosage of Clonidine. Severe rash on face and leg,
  slightly itchy."
}
```

输出：

```
{
  "Id": 0,
  "BeginOffset": 38,
  "EndOffset": 47,
  "Score": 0.9942955374717712,
  "Text": "Clonidine",
  "Category": "MEDICATION",
  "Type": "GENERIC_NAME",
  "Traits": []
}
```

有关更多信息，请参阅亚马逊 Comprehend Medical 开发者指南中的[检测实体版本 2](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [DetectEntitiesV2](#)。

## detect-phi

以下代码示例演示了如何使用 detect-phi。

### AWS CLI

示例 1：直接从文本中检测受保护的健康信息 (PHI)

以下 detect-phi 示例直接从输入文本中显示检测到的受保护健康信息 (PHI) 实体。

```
aws comprehendmedical detect-phi \
  --text "Patient Carlos Salazar presented with rash on his upper extremities and
  dry cough. He lives at 100 Main Street, Anytown, USA where he works from his home
  as a carpenter."
```

输出：

```
{
```



```
"Entities": [  
  {  
    "Id": 0,  
    "BeginOffset": 8,  
    "EndOffset": 21,  
    "Score": 0.9914507269859314,  
    "Text": "Carlos Salazar",  
    "Category": "PROTECTED_HEALTH_INFORMATION",  
    "Type": "NAME",  
    "Traits": []  
  },  
  {  
    "Id": 1,  
    "BeginOffset": 94,  
    "EndOffset": 109,  
    "Score": 0.871849775314331,  
    "Text": "100 Main Street, Anytown, USA",  
    "Category": "PROTECTED_HEALTH_INFORMATION",  
    "Type": "ADDRESS",  
    "Traits": []  
  },  
  {  
    "Id": 2,  
    "BeginOffset": 145,  
    "EndOffset": 154,  
    "Score": 0.8302185535430908,  
    "Text": "carpenter",  
    "Category": "PROTECTED_HEALTH_INFORMATION",  
    "Type": "PROFESSION",  
    "Traits": []  
  }  
],  
"ModelVersion": "0.0.0"  
}
```

有关更多信息，请参阅亚马逊 [Comprehend Medical 开发者指南中的检测 PHI](#)。

示例 2：直接从文件路径检测保护健康信息 (PHI)

以下 detect-phi 示例显示了从文件路径中检测到的受保护健康信息 (PHI) 实体。

```
aws comprehendmedical detect-phi \  
  --text file://phi.txt
```

phi.txt 的内容：

```
"Patient Carlos Salazar presented with a rash on his upper extremities and a dry cough. He lives at 100 Main Street, Anytown, USA, where he works from his home as a carpenter."
```

输出：

```
{
  "Entities": [
    {
      "Id": 0,
      "BeginOffset": 8,
      "EndOffset": 21,
      "Score": 0.9914507269859314,
      "Text": "Carlos Salazar",
      "Category": "PROTECTED_HEALTH_INFORMATION",
      "Type": "NAME",
      "Traits": []
    },
    {
      "Id": 1,
      "BeginOffset": 94,
      "EndOffset": 109,
      "Score": 0.871849775314331,
      "Text": "100 Main Street, Anytown, USA",
      "Category": "PROTECTED_HEALTH_INFORMATION",
      "Type": "ADDRESS",
      "Traits": []
    },
    {
      "Id": 2,
      "BeginOffset": 145,
      "EndOffset": 154,
      "Score": 0.8302185535430908,
      "Text": "carpenter",
      "Category": "PROTECTED_HEALTH_INFORMATION",
      "Type": "PROFESSION",
      "Traits": []
    }
  ],
  "ModelVersion": "0.0.0"
}
```

有关更多信息，请参阅亚马逊 [Comprehend Medical 开发者指南中的检测 PHI](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DetectPhi](#) 中的。

## infer-icd10-cm

以下代码示例演示了如何使用 infer-icd10-cm。

### AWS CLI

示例 1：检测医疗状况实体并直接从文本链接到 ICD-10-CM 本体论

以下 infer-icd10-cm 示例标记了检测到的医疗状况实体，并将这些实体与 2019 年版《国际疾病分类临床修改》(ICD-10-CM) 中的代码关联起来。

```
aws comprehendmedical infer-icd10-cm \  
  --text "The patient complains of abdominal pain, has a long-standing history of  
  diabetes treated with Micronase daily."
```

输出：

```
{  
  "Entities": [  
    {  
      "Id": 0,  
      "Text": "abdominal pain",  
      "Category": "MEDICAL_CONDITION",  
      "Type": "DX_NAME",  
      "Score": 0.9475538730621338,  
      "BeginOffset": 28,  
      "EndOffset": 42,  
      "Attributes": [],  
      "Traits": [  
        {  
          "Name": "SYMPTOM",  
          "Score": 0.6724207401275635  
        }  
      ],  
      "ICD10CMConcepts": [  
        {  
          "Description": "Unspecified abdominal pain",  
          "Code": "R10.9",  
          "Score": 0.6904221177101135  
        }  
      ]  
    }  
  ]  
}
```

```
    },
    {
      "Description": "Epigastric pain",
      "Code": "R10.13",
      "Score": 0.1364113688468933
    },
    {
      "Description": "Generalized abdominal pain",
      "Code": "R10.84",
      "Score": 0.12508003413677216
    },
    {
      "Description": "Left lower quadrant pain",
      "Code": "R10.32",
      "Score": 0.10063883662223816
    },
    {
      "Description": "Lower abdominal pain, unspecified",
      "Code": "R10.30",
      "Score": 0.09933677315711975
    }
  ]
},
{
  "Id": 1,
  "Text": "diabetes",
  "Category": "MEDICAL_CONDITION",
  "Type": "DX_NAME",
  "Score": 0.9899052977561951,
  "BeginOffset": 75,
  "EndOffset": 83,
  "Attributes": [],
  "Traits": [
    {
      "Name": "DIAGNOSIS",
      "Score": 0.9258432388305664
    }
  ],
  "ICD10CMConcepts": [
    {
      "Description": "Type 2 diabetes mellitus without complications",
      "Code": "E11.9",
      "Score": 0.7158446311950684
    }
  ],
}
```

```

        {
            "Description": "Family history of diabetes mellitus",
            "Code": "Z83.3",
            "Score": 0.5704703330993652
        },
        {
            "Description": "Family history of other endocrine, nutritional
and metabolic diseases",
            "Code": "Z83.49",
            "Score": 0.19856023788452148
        },
        {
            "Description": "Type 1 diabetes mellitus with ketoacidosis
without coma",
            "Code": "E10.10",
            "Score": 0.13285516202449799
        },
        {
            "Description": "Type 2 diabetes mellitus with hyperglycemia",
            "Code": "E11.65",
            "Score": 0.0993388369679451
        }
    ]
}
    ],
    "ModelVersion": "0.1.0"
}

```

有关更多信息，请参阅《亚马逊 Comprehend Medical 开发者指南》中的 [Infer ICD10-CM](#)。

示例 2：检测医疗状况实体并通过文件路径链接到 ICD-10-CM 本体论

以下 infer-icd-10-cm 示例标记了检测到的医疗状况实体，并将这些实体与 2019 年版《国际疾病分类临床修改》(ICD-10-CM) 中的代码关联起来。

```
aws comprehendmedical infer-icd10-cm \
  --text file://icd10cm.txt
```

icd10cm.txt 的内容：

```
{
  "The patient complains of abdominal pain, has a long-standing history of
diabetes treated with Micronase daily."
```

```
}
```

输出：

```
{
  "Entities": [
    {
      "Id": 0,
      "Text": "abdominal pain",
      "Category": "MEDICAL_CONDITION",
      "Type": "DX_NAME",
      "Score": 0.9475538730621338,
      "BeginOffset": 28,
      "EndOffset": 42,
      "Attributes": [],
      "Traits": [
        {
          "Name": "SYMPTOM",
          "Score": 0.6724207401275635
        }
      ],
      "ICD10CMConcepts": [
        {
          "Description": "Unspecified abdominal pain",
          "Code": "R10.9",
          "Score": 0.6904221177101135
        },
        {
          "Description": "Epigastric pain",
          "Code": "R10.13",
          "Score": 0.1364113688468933
        },
        {
          "Description": "Generalized abdominal pain",
          "Code": "R10.84",
          "Score": 0.12508003413677216
        },
        {
          "Description": "Left lower quadrant pain",
          "Code": "R10.32",
          "Score": 0.10063883662223816
        },
        {
```

```

        "Description": "Lower abdominal pain, unspecified",
        "Code": "R10.30",
        "Score": 0.09933677315711975
      }
    ]
  },
  {
    "Id": 1,
    "Text": "diabetes",
    "Category": "MEDICAL_CONDITION",
    "Type": "DX_NAME",
    "Score": 0.9899052977561951,
    "BeginOffset": 75,
    "EndOffset": 83,
    "Attributes": [],
    "Traits": [
      {
        "Name": "DIAGNOSIS",
        "Score": 0.9258432388305664
      }
    ],
    "ICD10CMConcepts": [
      {
        "Description": "Type 2 diabetes mellitus without complications",
        "Code": "E11.9",
        "Score": 0.7158446311950684
      },
      {
        "Description": "Family history of diabetes mellitus",
        "Code": "Z83.3",
        "Score": 0.5704703330993652
      },
      {
        "Description": "Family history of other endocrine, nutritional
and metabolic diseases",
        "Code": "Z83.49",
        "Score": 0.19856023788452148
      },
      {
        "Description": "Type 1 diabetes mellitus with ketoacidosis
without coma",
        "Code": "E10.10",
        "Score": 0.13285516202449799
      },
    ]
  },

```

```

        {
            "Description": "Type 2 diabetes mellitus with hyperglycemia",
            "Code": "E11.65",
            "Score": 0.0993388369679451
        }
    ]
}
],
"ModelVersion": "0.1.0"
}

```

欲了解更多信息，请参阅亚马逊 [Comprehend Medical 开发者指南中的 infer-icd10-cm](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [InferIcd10Cm](#)。

## infer-rx-norm

以下代码示例演示了如何使用 infer-rx-norm。

### AWS CLI

示例 1：检测药物实体并 RxNorm 直接从文本链接到

以下 infer-rx-norm 示例显示并标记了检测到的药物实体，并将这些实体链接到美国国家医学图书馆 RxNorm 数据库中的概念标识符 (rxCUI)。

```

aws comprehendmedical infer-rx-norm \
  --text "Patient reports taking Levothyroxine 125 micrograms p.o. once daily, but
denies taking Synthroid."

```

输出：

```

{
  "Entities": [
    {
      "Id": 0,
      "Text": "Levothyroxine",
      "Category": "MEDICATION",
      "Type": "GENERIC_NAME",
      "Score": 0.9996285438537598,
      "BeginOffset": 23,
      "EndOffset": 36,
      "Attributes": [

```



```
{
  "Type": "DOSAGE",
  "Score": 0.9892290830612183,
  "RelationshipScore": 0.9997978806495667,
  "Id": 1,
  "BeginOffset": 37,
  "EndOffset": 51,
  "Text": "125 micrograms",
  "Traits": []
},
{
  "Type": "ROUTE_OR_MODE",
  "Score": 0.9988924860954285,
  "RelationshipScore": 0.998291552066803,
  "Id": 2,
  "BeginOffset": 52,
  "EndOffset": 56,
  "Text": "p.o.",
  "Traits": []
},
{
  "Type": "FREQUENCY",
  "Score": 0.9953463673591614,
  "RelationshipScore": 0.9999889135360718,
  "Id": 3,
  "BeginOffset": 57,
  "EndOffset": 67,
  "Text": "once daily",
  "Traits": []
}
],
"Traits": [],
"RxNormConcepts": [
  {
    "Description": "Levothyroxine Sodium 0.125 MG Oral Tablet",
    "Code": "966224",
    "Score": 0.9912070631980896
  },
  {
    "Description": "Levothyroxine Sodium 0.125 MG Oral Capsule",
    "Code": "966405",
    "Score": 0.8698278665542603
  },
  {
```

```

        "Description": "Levothyroxine Sodium 0.125 MG Oral Tablet
[Synthroid]",
        "Code": "966191",
        "Score": 0.7448257803916931
    },
    {
        "Description": "levothyroxine",
        "Code": "10582",
        "Score": 0.7050482630729675
    },
    {
        "Description": "Levothyroxine Sodium 0.125 MG Oral Tablet
[Levoxy1]",
        "Code": "966190",
        "Score": 0.6921631693840027
    }
]
},
{
    "Id": 4,
    "Text": "Synthroid",
    "Category": "MEDICATION",
    "Type": "BRAND_NAME",
    "Score": 0.9946461319923401,
    "BeginOffset": 86,
    "EndOffset": 95,
    "Attributes": [],
    "Traits": [
        {
            "Name": "NEGATION",
            "Score": 0.5167351961135864
        }
    ],
    "RxNormConcepts": [
        {
            "Description": "Synthroid",
            "Code": "224920",
            "Score": 0.9462039470672607
        },
        {
            "Description": "Levothyroxine Sodium 0.088 MG Oral Tablet
[Synthroid]",
            "Code": "966282",
            "Score": 0.8309829235076904
        }
    ]
}

```

```

    },
    {
      "Description": "Levothyroxine Sodium 0.125 MG Oral Tablet
[Synthroid]",
      "Code": "966191",
      "Score": 0.4945160448551178
    },
    {
      "Description": "Levothyroxine Sodium 0.05 MG Oral Tablet
[Synthroid]",
      "Code": "966247",
      "Score": 0.3674522042274475
    },
    {
      "Description": "Levothyroxine Sodium 0.025 MG Oral Tablet
[Synthroid]",
      "Code": "966158",
      "Score": 0.2588822841644287
    }
  ]
}
],
"ModelVersion": "0.0.0"
}

```

有关更多信息，请参阅亚马逊 Comprehend Medical [开发者指南 RxNorm 中的推断](#)。

示例 2：检测药物实体并 RxNorm 从文件路径链接到。

以下 `infer-rx-norm` 示例显示并标记了检测到的药物实体，并将这些实体链接到美国国家医学图书馆 RxNorm 数据库中的概念标识符 (rxCUI)。

```
aws comprehendmedical infer-rx-norm \
  --text file://rxnorm.txt
```

`rxnorm.txt` 的内容：

```
{
  "Patient reports taking Levothyroxine 125 micrograms p.o. once daily, but denies
  taking Synthroid."
}
```

输出：

```
{
  "Entities": [
    {
      "Id": 0,
      "Text": "Levothyroxine",
      "Category": "MEDICATION",
      "Type": "GENERIC_NAME",
      "Score": 0.9996285438537598,
      "BeginOffset": 23,
      "EndOffset": 36,
      "Attributes": [
        {
          "Type": "DOSAGE",
          "Score": 0.9892290830612183,
          "RelationshipScore": 0.9997978806495667,
          "Id": 1,
          "BeginOffset": 37,
          "EndOffset": 51,
          "Text": "125 micrograms",
          "Traits": []
        },
        {
          "Type": "ROUTE_OR_MODE",
          "Score": 0.9988924860954285,
          "RelationshipScore": 0.998291552066803,
          "Id": 2,
          "BeginOffset": 52,
          "EndOffset": 56,
          "Text": "p.o.",
          "Traits": []
        },
        {
          "Type": "FREQUENCY",
          "Score": 0.9953463673591614,
          "RelationshipScore": 0.9999889135360718,
          "Id": 3,
          "BeginOffset": 57,
          "EndOffset": 67,
          "Text": "once daily",
          "Traits": []
        }
      ],
      "Traits": [],
    }
  ]
}
```

```

    "RxNormConcepts": [
      {
        "Description": "Levothyroxine Sodium 0.125 MG Oral Tablet",
        "Code": "966224",
        "Score": 0.9912070631980896
      },
      {
        "Description": "Levothyroxine Sodium 0.125 MG Oral Capsule",
        "Code": "966405",
        "Score": 0.8698278665542603
      },
      {
        "Description": "Levothyroxine Sodium 0.125 MG Oral Tablet
[Synthroid]",
        "Code": "966191",
        "Score": 0.7448257803916931
      },
      {
        "Description": "levothyroxine",
        "Code": "10582",
        "Score": 0.7050482630729675
      },
      {
        "Description": "Levothyroxine Sodium 0.125 MG Oral Tablet
[Levoxy1]",
        "Code": "966190",
        "Score": 0.6921631693840027
      }
    ]
  },
  {
    "Id": 4,
    "Text": "Synthroid",
    "Category": "MEDICATION",
    "Type": "BRAND_NAME",
    "Score": 0.9946461319923401,
    "BeginOffset": 86,
    "EndOffset": 95,
    "Attributes": [],
    "Traits": [
      {
        "Name": "NEGATION",
        "Score": 0.5167351961135864
      }
    ]
  }

```

```

    ],
    "RxNormConcepts": [
      {
        "Description": "Synthroid",
        "Code": "224920",
        "Score": 0.9462039470672607
      },
      {
        "Description": "Levothyroxine Sodium 0.088 MG Oral Tablet
[Synthroid]",
        "Code": "966282",
        "Score": 0.8309829235076904
      },
      {
        "Description": "Levothyroxine Sodium 0.125 MG Oral Tablet
[Synthroid]",
        "Code": "966191",
        "Score": 0.4945160448551178
      },
      {
        "Description": "Levothyroxine Sodium 0.05 MG Oral Tablet
[Synthroid]",
        "Code": "966247",
        "Score": 0.3674522042274475
      },
      {
        "Description": "Levothyroxine Sodium 0.025 MG Oral Tablet
[Synthroid]",
        "Code": "966158",
        "Score": 0.2588822841644287
      }
    ]
  }
],
  "ModelVersion": "0.0.0"
}

```

有关更多信息，请参阅亚马逊 Comprehend Medical [开发者指南 RxNorm](#) 中的 [推断](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [InferRxNorm](#) 中的。

## infer-snomedct

以下代码示例演示了如何使用 infer-snomedct。

### AWS CLI

示例：检测实体并直接从文本链接到 SNOMED CT 本体论

以下infer-snomedct示例说明如何检测医疗实体并将其与2021-03年版的系统化医学命名法，临床术语 ( SNOMED CT ) 中的概念关联起来。

```
aws comprehendmedical infer-snomedct \  
  --text "The patient complains of abdominal pain, has a long-standing history of  
  diabetes treated with Micronase daily."
```

输出：

```
{  
  "Entities": [  
    {  
      "Id": 3,  
      "BeginOffset": 26,  
      "EndOffset": 40,  
      "Score": 0.9598260521888733,  
      "Text": "abdominal pain",  
      "Category": "MEDICAL_CONDITION",  
      "Type": "DX_NAME",  
      "Traits": [  
        {  
          "Name": "SYMPTOM",  
          "Score": 0.6819021701812744  
        }  
      ]  
    },  
    {  
      "Id": 4,  
      "BeginOffset": 73,  
      "EndOffset": 81,  
      "Score": 0.9905840158462524,  
      "Text": "diabetes",  
      "Category": "MEDICAL_CONDITION",  
      "Type": "DX_NAME",  
      "Traits": [  

```

```

        {
            "Name": "DIAGNOSIS",
            "Score": 0.9255214333534241
        }
    ],
    {
        "Id": 1,
        "BeginOffset": 95,
        "EndOffset": 104,
        "Score": 0.6371926665306091,
        "Text": "Micronase",
        "Category": "MEDICATION",
        "Type": "BRAND_NAME",
        "Traits": [],
        "Attributes": [
            {
                "Type": "FREQUENCY",
                "Score": 0.9761165380477905,
                "RelationshipScore": 0.9984188079833984,
                "RelationshipType": "FREQUENCY",
                "Id": 2,
                "BeginOffset": 105,
                "EndOffset": 110,
                "Text": "daily",
                "Category": "MEDICATION",
                "Traits": []
            }
        ]
    }
],
"UnmappedAttributes": [],
"ModelVersion": "1.0.0"
}

```

欲了解更多信息，请参阅亚马逊 C [omprehend Medical 开发者指南中的 InfnomedCT](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[InferSnomedct](#)中的。

## list-entities-detection-v2-jobs

以下代码示例演示了如何使用 list-entities-detection-v2-jobs。



## AWS CLI

### 列出实体检测任务

以下 `list-entities-detection-v2-jobs` 示例列出了当前的异步检测作业。

```
aws comprehendmedical list-entities-detection-v2-jobs
```

输出：

```
{
  "ComprehendMedicalAsyncJobPropertiesList": [
    {
      "JobId": "ab9887877365fe70299089371c043b96",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2020-03-19T20:38:37.594000+00:00",
      "EndTime": "2020-03-19T20:45:07.894000+00:00",
      "ExpirationTime": "2020-07-17T20:38:37+00:00",
      "InputDataConfig": {
        "S3Bucket": "comp-med-input",
        "S3Key": ""
      },
      "OutputDataConfig": {
        "S3Bucket": "comp-med-output",
        "S3Key": "867139942017-EntitiesDetection-ab9887877365fe70299089371c043b96/"
      },
      "LanguageCode": "en",
      "DataAccessRoleArn": "arn:aws:iam::867139942017:role/ComprehendMedicalBatchProcessingRole",
      "ModelVersion": "DetectEntitiesModelV20190930"
    }
  ]
}
```

有关更多信息，请参阅亚马逊 Comprehend Medical 开发者指南中的 [批处理 API](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [ListEntitiesDetectionV2Jobs](#)。

### list-icd10-cm-inference-jobs

以下代码示例演示了如何使用 `list-icd10-cm-inference-jobs`。

## AWS CLI

列出所有当前的 ICD-10-CM 推理任务

以下示例显示了该 `list-icd10-cm-inference-jobs` 操作如何返回当前异步 ICD-10-CM 批量推理作业列表。

```
aws comprehendmedical list-icd10-cm-inference-jobs
```

输出：

```
{
  "ComprehendMedicalAsyncJobPropertiesList": [
    {
      "JobId": "5780034166536cdb52ffa3295a1b00a7",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2020-05-19T20:38:37.594000+00:00",
      "EndTime": "2020-05-19T20:45:07.894000+00:00",
      "ExpirationTime": "2020-09-17T20:38:37+00:00",
      "InputDataConfig": {
        "S3Bucket": "comp-med-input",
        "S3Key": "AKIAIOSFODNN7EXAMPLE"
      },
      "OutputDataConfig": {
        "S3Bucket": "comp-med-output",
        "S3Key": "AKIAIOSFODNN7EXAMPLE"
      },
      "LanguageCode": "en",
      "DataAccessRoleArn": "arn:aws:iam::867139942017:role/ComprehendMedicalBatchProcessingRole",
      "ModelVersion": "0.1.0"
    }
  ]
}
```

有关更多信息，请参阅 Amazon Comprehend Medical [开发者指南中的本体链接批量分析](#)。

- 有关 API 的详细信息，请参阅 `CmlInferenceJobs` 《AWS CLI 命令参考》中的 [ListIcd10](#)。

### **list-phi-detection-jobs**

以下代码示例演示了如何使用 `list-phi-detection-jobs`。

## AWS CLI

列出受保护的健康信息 (PHI) 检测作业

以下 `list-phi-detection-jobs` 示例列出了当前的受保护健康信息 (PHI) 检测作业

```
aws comprehendmedical list-phi-detection-jobs
```

输出：

```
{
  "ComprehendMedicalAsyncJobPropertiesList": [
    {
      "JobId": "4750034166536cdb52ffa3295a1b00a3",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2020-03-19T20:38:37.594000+00:00",
      "EndTime": "2020-03-19T20:45:07.894000+00:00",
      "ExpirationTime": "2020-07-17T20:38:37+00:00",
      "InputDataConfig": {
        "S3Bucket": "comp-med-input",
        "S3Key": ""
      },
      "OutputDataConfig": {
        "S3Bucket": "comp-med-output",
        "S3Key": "867139942017-
PHIDetection-4750034166536cdb52ffa3295a1b00a3/"
      },
      "LanguageCode": "en",
      "DataAccessRoleArn": "arn:aws:iam::867139942017:role/
ComprehendMedicalBatchProcessingRole",
      "ModelVersion": "PHIModelV20190903"
    }
  ]
}
```

有关更多信息，请参阅亚马逊 Comprehend Medical 开发者指南中的[批处理 API](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPhiDetectionJobs](#)中的。

### **list-rx-norm-inference-jobs**

以下代码示例演示了如何使用 `list-rx-norm-inference-jobs`。

## AWS CLI

列出所有当前的 Rx-Norm 推理作业

以下示例显示如何 `list-rx-norm-inference-jobs` 返回当前异步 Rx-Norm 批量推理作业的列表。

```
aws comprehendmedical list-rx-norm-inference-jobs
```

输出：

```
{
  "ComprehendMedicalAsyncJobPropertiesList": [
    {
      "JobId": "4980034166536cfb52gga3295a1b00a3",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2020-05-19T20:38:37.594000+00:00",
      "EndTime": "2020-05-19T20:45:07.894000+00:00",
      "ExpirationTime": "2020-09-17T20:38:37+00:00",
      "InputDataConfig": {
        "S3Bucket": "comp-med-input",
        "S3Key": "AKIAIOSFODNN7EXAMPLE"
      },
      "OutputDataConfig": {
        "S3Bucket": "comp-med-output",
        "S3Key": "AKIAIOSFODNN7EXAMPLE"
      },
      "LanguageCode": "en",
      "DataAccessRoleArn": "arn:aws:iam::867139942017:role/ComprehendMedicalBatchProcessingRole",
      "ModelVersion": "0.0.0"
    }
  ]
}
```

有关更多信息，请参阅 Amazon Comprehend Medical [开发者指南中的本体链接批量分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListRxNormInferenceJobs](#) 中的。

### **list-snomedct-inference-jobs**

以下代码示例演示了如何使用 `list-snomedct-inference-jobs`。

## AWS CLI

### 列出所有 SNOMED CT 推理作业

以下示例显示该`list-snomedct-inference-jobs`操作如何返回当前异步 SNOMED CT 批量推理作业的列表。

```
aws comprehendmedical list-snomedct-inference-jobs
```

输出：

```
{
  "ComprehendMedicalAsyncJobPropertiesList": [
    {
      "JobId": "5780034166536cdb52ffa3295a1b00a7",
      "JobStatus": "COMPLETED",
      "SubmitTime": "2020-05-19T20:38:37.594000+00:00",
      "EndTime": "2020-05-19T20:45:07.894000+00:00",
      "ExpirationTime": "2020-09-17T20:38:37+00:00",
      "InputDataConfig": {
        "S3Bucket": "comp-med-input",
        "S3Key": "AKIAIOSFODNN7EXAMPLE"
      },
      "OutputDataConfig": {
        "S3Bucket": "comp-med-output",
        "S3Key": "AKIAIOSFODNN7EXAMPLE"
      },
      "LanguageCode": "en",
      "DataAccessRoleArn": "arn:aws:iam::867139942017:role/ComprehendMedicalBatchProcessingRole",
      "ModelVersion": "0.1.0"
    }
  ]
}
```

有关更多信息，请参阅 Amazon Comprehend Medical [开发者指南中的本体链接批量分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListSnomedctInferenceJobs](#) 中的。

### **start-entities-detection-v2-job**

以下代码示例演示了如何使用 `start-entities-detection-v2-job`。

## AWS CLI

### 启动实体检测作业

以下 `start-entities-detection-v2-job` 示例启动异步实体检测作业。

```
aws comprehendmedical start-entities-detection-v2-job \  
  --input-data-config "S3Bucket=comp-med-input" \  
  --output-data-config "S3Bucket=comp-med-output" \  
  --data-access-role-arn arn:aws:iam::867139942017:role/  
ComprehendMedicalBatchProcessingRole \  
  --language-code en
```

输出：

```
{  
  "JobId": "ab9887877365fe70299089371c043b96"  
}
```

有关更多信息，请参阅亚马逊 Comprehend Medical 开发者指南中的 [批处理 API](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [StartEntitiesDetectionV2Job](#)。

### **start-icd10-cm-inference-job**

以下代码示例演示了如何使用 `start-icd10-cm-inference-job`。

## AWS CLI

### 启动 ICD-10-CM 推理作业

以下 `start-icd10-cm-inference-job` 示例启动 ICD-10-CM 推理批量分析作业。

```
aws comprehendmedical start-icd10-cm-inference-job \  
  --input-data-config "S3Bucket=comp-med-input" \  
  --output-data-config "S3Bucket=comp-med-output" \  
  --data-access-role-arn arn:aws:iam::867139942017:role/  
ComprehendMedicalBatchProcessingRole \  
  --language-code en
```

输出：

```
{
  "JobId": "ef7289877365fc70299089371c043b96"
}
```

有关更多信息，请参阅 Amazon Comprehend Medical [开发者指南中的本体链接批量分析](#)。

- 有关 API 的详细信息，请参阅 CmlInferenceJob 《AWS CLI 命令参考》中的 [StartIcd10](#)。

## start-phi-detection-job

以下代码示例演示了如何使用 start-phi-detection-job。

### AWS CLI

启动 PHI 检测作业

以下 start-phi-detection-job 示例启动异步 PHI 实体检测作业。

```
aws comprehendmedical start-phi-detection-job \
  --input-data-config "S3Bucket=comp-med-input" \
  --output-data-config "S3Bucket=comp-med-output" \
  --data-access-role-arn arn:aws:iam::867139942017:role/
ComprehendMedicalBatchProcessingRole \
  --language-code en
```

输出：

```
{
  "JobId": "ab9887877365fe70299089371c043b96"
}
```

有关更多信息，请参阅亚马逊 Comprehend Medical 开发者指南中的 [批处理 API](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartPhiDetectionJob](#) 中的。

## start-rx-norm-inference-job

以下代码示例演示了如何使用 start-rx-norm-inference-job。

### AWS CLI

开始 RxNorm 推理作业

以下 `start-rx-norm-inference-job` 示例启动 RxNorm 推理批量分析作业。

```
aws comprehendmedical start-rx-norm-inference-job \  
  --input-data-config "S3Bucket=comp-med-input" \  
  --output-data-config "S3Bucket=comp-med-output" \  
  --data-access-role-arn arn:aws:iam::867139942017:role/  
ComprehendMedicalBatchProcessingRole \  
  --language-code en
```

输出：

```
{  
  "JobId": "eg8199877365fc70299089371c043b96"  
}
```

有关更多信息，请参阅 Amazon Comprehend Medical [开发者指南中的本体链接批量分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartRxNormInferenceJob](#) 中的。

## start-snomedct-inference-job

以下代码示例演示了如何使用 `start-snomedct-inference-job`。

### AWS CLI

启动 SNOMED CT 推理作业

以下 `start-snomedct-inference-job` 示例启动 SNOMED CT 推理批量分析作业。

```
aws comprehendmedical start-snomedct-inference-job \  
  --input-data-config "S3Bucket=comp-med-input" \  
  --output-data-config "S3Bucket=comp-med-output" \  
  --data-access-role-arn arn:aws:iam::867139942017:role/  
ComprehendMedicalBatchProcessingRole \  
  --language-code en
```

输出：

```
{  
  "JobId": "dg7289877365fc70299089371c043b96"  
}
```



有关更多信息，请参阅 Amazon Comprehend Medical [开发者指南中的本体链接批量分析](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartSnomedctInferenceJob](#) 中的。

## stop-entities-detection-v2-job

以下代码示例演示了如何使用 stop-entities-detection-v2-job。

### AWS CLI

停止实体检测作业

以下 stop-entities-detection-v2-job 示例停止异步实体检测作业。

```
aws comprehendmedical stop-entities-detection-v2-job \  
  --job-id "ab9887877365fe70299089371c043b96"
```

输出：

```
{  
  "JobId": "ab9887877365fe70299089371c043b96"  
}
```

有关更多信息，请参阅亚马逊 Comprehend Medical 开发者指南中的 [批处理 API](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [StopEntitiesDetectionV2Job](#)。

## stop-icd10-cm-inference-job

以下代码示例演示了如何使用 stop-icd10-cm-inference-job。

### AWS CLI

停止 ICD-10-CM 推理作业

以下 stop-icd10-cm-inference-job 示例停止 ICD-10-CM 推理批量分析作业。

```
aws comprehendmedical stop-icd10-cm-inference-job \  
  --job-id "4750034166536cdb52ffa3295a1b00a3"
```

输出：

```
{
  "JobId": "ef7289877365fc70299089371c043b96",
}
```

有关更多信息，请参阅 Amazon Comprehend Medical [开发者指南中的本体链接批量分析](#)。

- 有关 API 的详细信息，请参阅 CmlInferenceJob 《AWS CLI 命令参考》中的 [StopIcd10](#)。

## stop-phi-detection-job

以下代码示例演示了如何使用 stop-phi-detection-job。

### AWS CLI

停止受保护的健康信息 (PHI) 检测作业

以下 stop-phi-detection-job 示例停止异步受保护的健康信息 (PHI) 检测作业。

```
aws comprehendmedical stop-phi-detection-job \
  --job-id "4750034166536cdb52ffa3295a1b00a3"
```

输出：

```
{
  "JobId": "ab9887877365fe70299089371c043b96"
}
```

有关更多信息，请参阅亚马逊 Comprehend Medical 开发者指南中的 [批处理 API](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StopPhiDetectionJob](#) 中的。

## stop-rx-norm-inference-job

以下代码示例演示了如何使用 stop-rx-norm-inference-job。

### AWS CLI

停止 RxNorm 推理作业

以下 stop-rx-norm-inference-job 示例停止 ICD-10-CM 推理批量分析作业。

```
aws comprehendmedical stop-rx-norm-inference-job \
```

```
--job-id "eg8199877365fc70299089371c043b96"
```

输出：

```
{
  "JobId": "eg8199877365fc70299089371c043b96",
}
```

有关更多信息，请参阅 Amazon Comprehend Medical [开发者指南中的本体链接批量分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopRxNormInferenceJob](#)中的。

## stop-snomedct-inference-job

以下代码示例演示了如何使用 stop-snomedct-inference-job。

### AWS CLI

停止 SNOMED CT 推理作业

以下stop-snomedct-inference-job示例停止 SNOMED CT 推理批量分析作业。

```
aws comprehendmedical stop-snomedct-inference-job \
  --job-id "8750034166436cdb52ffa3295a1b00a1"
```

输出：

```
{
  "JobId": "8750034166436cdb52ffa3295a1b00a1",
}
```

有关更多信息，请参阅 Amazon Comprehend Medical [开发者指南中的本体链接批量分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopSnomedctInferenceJob](#)中的。

## AWS Config 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Config。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## **delete-config-rule**

以下代码示例演示了如何使用 delete-config-rule。

AWS CLI

删除 C AWS onfig 规则

以下命令删除名为的 AWS Config 规则MyConfigRule：

```
aws configservice delete-config-rule --config-rule-name MyConfigRule
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteConfigRule](#)中的。

## **delete-delivery-channel**

以下代码示例演示了如何使用 delete-delivery-channel。

AWS CLI

删除配送渠道

以下命令删除默认的传送渠道：

```
aws configservice delete-delivery-channel --delivery-channel-name default
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDeliveryChannel](#)中的。

## **delete-evaluation-results**

以下代码示例演示了如何使用 delete-evaluation-results。

## AWS CLI

### 手动删除评估结果

以下命令删除 AWS 托管规则 s3-的当前评估结果bucket-versioning-enabled :

```
aws configservice delete-evaluation-results --config-rule-name s3-bucket-versioning-enabled
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteEvaluationResults](#)中的。

## deliver-config-snapshot

以下代码示例演示了如何使用 deliver-config-snapshot。

### AWS CLI

#### 提供配置快照

以下命令将配置快照传送到属于默认传输渠道的 Amazon S3 存储桶：

```
aws configservice deliver-config-snapshot --delivery-channel-name default
```

输出：

```
{
  "configSnapshotId": "d0333b00-a683-44af-921e-examplefb794"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeliverConfigSnapshot](#)中的。

## describe-compliance-by-config-rule

以下代码示例演示了如何使用 describe-compliance-by-config-rule。

### AWS CLI

获取您的 AWS Config 规则的合规性信息

以下命令返回一个或多个 AWS 资源违反的每个 AWS Config 规则的合规性信息：

```
aws configservice describe-compliance-by-config-rule --compliance-types
NON_COMPLIANT
```

在输出中，每个CappedCount属性的值表示有多少资源不符合相关规则。例如，以下输出表明 3 个资源不符合名为的规则InstanceTypesAreT2micro。

输出：

```
{
  "ComplianceByConfigRules": [
    {
      "Compliance": {
        "ComplianceContributorCount": {
          "CappedCount": 3,
          "CapExceeded": false
        },
        "ComplianceType": "NON_COMPLIANT"
      },
      "ConfigRuleName": "InstanceTypesAreT2micro"
    },
    {
      "Compliance": {
        "ComplianceContributorCount": {
          "CappedCount": 10,
          "CapExceeded": false
        },
        "ComplianceType": "NON_COMPLIANT"
      },
      "ConfigRuleName": "RequiredTagsForVolumes"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeComplianceByConfigRule](#)中的。

## describe-compliance-by-resource

以下代码示例演示了如何使用 describe-compliance-by-resource。

### AWS CLI

获取 AWS 资源的合规性信息

以下命令返回由 AWS Config 记录且违反一条或多条规则的每个 EC2 实例的合规性信息：

```
aws configservice describe-compliance-by-resource --resource-type AWS::EC2::Instance
--compliance-types NON_COMPLIANT
```

在输出中，每个CappedCount属性的值表示资源违反了多少规则。例如，以下输出表明该实例i-1a2b3c4d违反了 2 条规则。

输出：

```
{
  "ComplianceByResources": [
    {
      "ResourceType": "AWS::EC2::Instance",
      "ResourceId": "i-1a2b3c4d",
      "Compliance": {
        "ComplianceContributorCount": {
          "CappedCount": 2,
          "CapExceeded": false
        },
        "ComplianceType": "NON_COMPLIANT"
      }
    },
    {
      "ResourceType": "AWS::EC2::Instance",
      "ResourceId": "i-2a2b3c4d ",
      "Compliance": {
        "ComplianceContributorCount": {
          "CappedCount": 3,
          "CapExceeded": false
        },
        "ComplianceType": "NON_COMPLIANT"
      }
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeComplianceByResource](#)中的。

## describe-config-rule-evaluation-status

以下代码示例演示了如何使用 describe-config-rule-evaluation-status。

## AWS CLI

获取 AWS Config 规则的状态信息

以下命令返回名为的 AWS Config 规则的状态信息MyConfigRule :

```
aws configservice describe-config-rule-evaluation-status --config-rule-names
MyConfigRule
```

输出 :

```
{
  "ConfigRulesEvaluationStatus": [
    {
      "ConfigRuleArn": "arn:aws:config:us-east-1:123456789012:config-rule/
config-rule-abcdef",
      "FirstActivatedTime": 1450311703.844,
      "ConfigRuleId": "config-rule-abcdef",
      "LastSuccessfulInvocationTime": 1450314643.156,
      "ConfigRuleName": "MyConfigRule"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeConfigRuleEvaluationStatus](#)中的。

## describe-config-rules

以下代码示例演示了如何使用 describe-config-rules。

### AWS CLI

获取 AWS Config 规则的详细信息

以下命令返回名为的 AWS Config 规则的详细信息InstanceTypesAreT2micro :

```
aws configservice describe-config-rules --config-rule-names InstanceTypesAreT2micro
```

输出 :

```
{
```



```
"ConfigRules": [
  {
    "ConfigRuleState": "ACTIVE",
    "Description": "Evaluates whether EC2 instances are the t2.micro type.",
    "ConfigRuleName": "InstanceTypesAreT2micro",
    "ConfigRuleArn": "arn:aws:config:us-east-1:123456789012:config-rule/
config-rule-abcdef",
    "Source": {
      "Owner": "CUSTOM_LAMBDA",
      "SourceIdentifier": "arn:aws:lambda:us-
east-1:123456789012:function:InstanceTypeCheck",
      "SourceDetails": [
        {
          "EventSource": "aws.config",
          "MessageType": "ConfigurationItemChangeNotification"
        }
      ]
    },
    "InputParameters": "{\"desiredInstanceType\":\"t2.micro\"}",
    "Scope": {
      "ComplianceResourceTypes": [
        "AWS::EC2::Instance"
      ]
    },
    "ConfigRuleId": "config-rule-abcdef"
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeConfigRules](#)中的。

## describe-configuration-recorder-status

以下代码示例演示了如何使用 describe-configuration-recorder-status。

### AWS CLI

获取配置记录器的状态信息

以下命令返回默认配置记录器的状态：

```
aws configservice describe-configuration-recorder-status
```

输出：

```
{
  "ConfigurationRecordersStatus": [
    {
      "name": "default",
      "lastStatus": "SUCCESS",
      "recording": true,
      "lastStatusChangeTime": 1452193834.344,
      "lastStartTime": 1441039997.819,
      "lastStopTime": 1441039992.835
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeConfigurationRecorderStatus](#)中的。

## describe-configuration-recorders

以下代码示例演示了如何使用 describe-configuration-recorders。

### AWS CLI

获取有关配置记录器的详细信息

以下命令返回有关默认配置记录器的详细信息：

```
aws configservice describe-configuration-recorders
```

输出：

```
{
  "ConfigurationRecorders": [
    {
      "recordingGroup": {
        "allSupported": true,
        "resourceTypes": [],
        "includeGlobalResourceTypes": true
      },
      "roleARN": "arn:aws:iam::123456789012:role/config-ConfigRole-A1B2C3D4E5F6",
      "name": "default"
    }
  ]
}
```

```
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeConfigurationRecorders](#)中的。

## describe-delivery-channel-status

以下代码示例演示了如何使用 `describe-delivery-channel-status`。

### AWS CLI

获取配送渠道的状态信息

以下命令返回传送渠道的状态：

```
aws configservice describe-delivery-channel-status
```

输出：

```
{  
  "DeliveryChannelsStatus": [  
    {  
      "configStreamDeliveryInfo": {  
        "lastStatusChangeTime": 1452193834.381,  
        "lastStatus": "SUCCESS"  
      },  
      "configHistoryDeliveryInfo": {  
        "lastSuccessfulTime": 1450317838.412,  
        "lastStatus": "SUCCESS",  
        "lastAttemptTime": 1450317838.412  
      },  
      "configSnapshotDeliveryInfo": {  
        "lastSuccessfulTime": 1452185597.094,  
        "lastStatus": "SUCCESS",  
        "lastAttemptTime": 1452185597.094  
      },  
      "name": "default"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDeliveryChannelStatus](#)中的。

## describe-delivery-channels

以下代码示例演示了如何使用 describe-delivery-channels。

### AWS CLI

获取有关配送渠道的详细信息

以下命令返回有关交付渠道的详细信息：

```
aws configservice describe-delivery-channels
```

输出：

```
{
  "DeliveryChannels": [
    {
      "snsTopicARN": "arn:aws:sns:us-east-1:123456789012:config-topic",
      "name": "default",
      "s3BucketName": "config-bucket-123456789012"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDeliveryChannels](#)中的。

## get-compliance-details-by-config-rule

以下代码示例演示了如何使用 get-compliance-details-by-config-rule。

### AWS CLI

获取 AWS Config 规则的评估结果

以下命令返回所有不符合名为的 AWS Config 规则的资源的评估结果InstanceTypesAreT2micro：

```
aws configservice get-compliance-details-by-config-rule --config-rule-name
InstanceTypesAreT2micro --compliance-types NON_COMPLIANT
```

输出：

```
{
  "EvaluationResults": [
    {
      "EvaluationResultIdentifier": {
        "OrderingTimestamp": 1450314635.065,
        "EvaluationResultQualifier": {
          "ResourceType": "AWS::EC2::Instance",
          "ResourceId": "i-1a2b3c4d",
          "ConfigRuleName": "InstanceTypesAreT2micro"
        }
      },
      "ResultRecordedTime": 1450314645.261,
      "ConfigRuleInvokedTime": 1450314642.948,
      "ComplianceType": "NON_COMPLIANT"
    },
    {
      "EvaluationResultIdentifier": {
        "OrderingTimestamp": 1450314635.065,
        "EvaluationResultQualifier": {
          "ResourceType": "AWS::EC2::Instance",
          "ResourceId": "i-2a2b3c4d",
          "ConfigRuleName": "InstanceTypesAreT2micro"
        }
      },
      "ResultRecordedTime": 1450314645.18,
      "ConfigRuleInvokedTime": 1450314642.902,
      "ComplianceType": "NON_COMPLIANT"
    },
    {
      "EvaluationResultIdentifier": {
        "OrderingTimestamp": 1450314635.065,
        "EvaluationResultQualifier": {
          "ResourceType": "AWS::EC2::Instance",
          "ResourceId": "i-3a2b3c4d",
          "ConfigRuleName": "InstanceTypesAreT2micro"
        }
      },
      "ResultRecordedTime": 1450314643.346,
      "ConfigRuleInvokedTime": 1450314643.124,
      "ComplianceType": "NON_COMPLIANT"
    }
  ]
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetComplianceDetailsByConfigRule](#)中的。

## get-compliance-details-by-resource

以下代码示例演示了如何使用 `get-compliance-details-by-resource`。

### AWS CLI

获取 AWS 资源的评估结果

以下命令返回 EC2 实例 `i-1a2b3c4d` 不符合的每条规则的评估结果：

```
aws configservice get-compliance-details-by-resource --resource-type
AWS::EC2::Instance --resource-id i-1a2b3c4d --compliance-types NON_COMPLIANT
```

输出：

```
{
  "EvaluationResults": [
    {
      "EvaluationResultIdentifier": {
        "OrderingTimestamp": 1450314635.065,
        "EvaluationResultQualifier": {
          "ResourceType": "AWS::EC2::Instance",
          "ResourceId": "i-1a2b3c4d",
          "ConfigRuleName": "InstanceTypesAreT2micro"
        }
      },
      "ResultRecordedTime": 1450314643.288,
      "ConfigRuleInvokedTime": 1450314643.034,
      "ComplianceType": "NON_COMPLIANT"
    },
    {
      "EvaluationResultIdentifier": {
        "OrderingTimestamp": 1450314635.065,
        "EvaluationResultQualifier": {
          "ResourceType": "AWS::EC2::Instance",
          "ResourceId": "i-1a2b3c4d",
          "ConfigRuleName": "RequiredTagForEC2Instances"
        }
      }
    }
  ]
}
```

```
    },
    "ResultRecordedTime": 1450314645.261,
    "ConfigRuleInvokedTime": 1450314642.948,
    "ComplianceType": "NON_COMPLIANT"
  }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetComplianceDetailsByResource](#)中的。

## get-compliance-summary-by-config-rule

以下代码示例演示了如何使用 `get-compliance-summary-by-config-rule`。

### AWS CLI

获取您的 AWS Config 规则的合规性摘要

以下命令返回合规规则的数量和不合规的规则数量：

```
aws configservice get-compliance-summary-by-config-rule
```

在输出中，每个 `CappedCount` 属性的值表示有多少规则合规或不合规。

输出：

```
{
  "ComplianceSummary": {
    "NonCompliantResourceCount": {
      "CappedCount": 3,
      "CapExceeded": false
    },
    "ComplianceSummaryTimestamp": 1452204131.493,
    "CompliantResourceCount": {
      "CappedCount": 2,
      "CapExceeded": false
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetComplianceSummaryByConfigRule](#)中的。

## get-compliance-summary-by-resource-type

以下代码示例演示了如何使用 `get-compliance-summary-by-resource-type`。

### AWS CLI

获取所有资源类型的合规性摘要

以下命令返回不合规的 AWS 资源数量和合规资源的数量：

```
aws configservice get-compliance-summary-by-resource-type
```

在输出中，每个 `CappedCount` 属性的值表示有多少资源合规或不合规。

输出：

```
{
  "ComplianceSummariesByResourceType": [
    {
      "ComplianceSummary": {
        "NonCompliantResourceCount": {
          "CappedCount": 16,
          "CapExceeded": false
        },
        "ComplianceSummaryTimestamp": 1453237464.543,
        "CompliantResourceCount": {
          "CappedCount": 10,
          "CapExceeded": false
        }
      }
    }
  ]
}
```

获取特定资源类型的合规性摘要

以下命令返回不合规的 EC2 实例的数量和合规的数量：

```
aws configservice get-compliance-summary-by-resource-type --resource-types
AWS::EC2::Instance
```

在输出中，每个 `CappedCount` 属性的值表示有多少资源合规或不合规。



输出：

```
{
  "ComplianceSummariesByResourceType": [
    {
      "ResourceType": "AWS::EC2::Instance",
      "ComplianceSummary": {
        "NonCompliantResourceCount": {
          "CappedCount": 3,
          "CapExceeded": false
        },
        "ComplianceSummaryTimestamp": 1452204923.518,
        "CompliantResourceCount": {
          "CappedCount": 7,
          "CapExceeded": false
        }
      }
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetComplianceSummaryByResourceType](#)中的。

## get-resource-config-history

以下代码示例演示了如何使用 `get-resource-config-history`。

### AWS CLI

获取 AWS 资源的配置历史记录

以下命令返回 ID 为 EC2 实例的配置项目列表 `i-1a2b3c4d`：

```
aws configservice get-resource-config-history --resource-type AWS::EC2::Instance --
resource-id i-1a2b3c4d
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetResourceConfigHistory](#)中的。

## get-status

以下代码示例演示了如何使用 `get-status`。

## AWS CLI

获取 AWS Config 的状态

以下命令返回传送渠道和配置记录器的状态：

```
aws configservice get-status
```

输出：

```
Configuration Recorders:

name: default
recorder: ON
last status: SUCCESS

Delivery Channels:

name: default
last stream delivery status: SUCCESS
last history delivery status: SUCCESS
last snapshot delivery status: SUCCESS
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetStatus](#) 中的。

## list-discovered-resources

以下代码示例演示了如何使用 list-discovered-resources。

### AWS CLI

列出 AWS Config 已发现的资源

以下命令列出了 AWS Config 已发现的 EC2 实例：

```
aws configservice list-discovered-resources --resource-type AWS::EC2::Instance
```

输出：

```
{
  "resourceIdentifiers": [
    {
```

```
    "resourceType": "AWS::EC2::Instance",
    "resourceId": "i-1a2b3c4d"
  },
  {
    "resourceType": "AWS::EC2::Instance",
    "resourceId": "i-2a2b3c4d"
  },
  {
    "resourceType": "AWS::EC2::Instance",
    "resourceId": "i-3a2b3c4d"
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDiscoveredResources](#)中的。

## put-config-rule

以下代码示例演示了如何使用 put-config-rule。

### AWS CLI

添加 AWS 托管 Config 规则

以下命令提供用于添加 AWS 托管 Config 规则的 JSON 代码：

```
aws configservice put-config-rule --config-rule file://
RequiredTagsForEC2Instances.json
```

RequiredTagsForEC2Instances.json 是一个包含规则配置的 JSON 文件：

```
{
  "ConfigRuleName": "RequiredTagsForEC2Instances",
  "Description": "Checks whether the CostCenter and Owner tags are applied to EC2
instances.",
  "Scope": {
    "ComplianceResourceTypes": [
      "AWS::EC2::Instance"
    ]
  },
  "Source": {
    "Owner": "AWS",
```

```

    "SourceIdentifier": "REQUIRED_TAGS"
  },
  "InputParameters": "{\"tag1Key\":\"CostCenter\",\"tag2Key\":\"Owner\"}"
}

```

对于该 `ComplianceResourceTypes` 属性，此 JSON 代码将范围限制为该 `AWS::EC2::Instance` 类型的资源，因此 AWS Config 将仅根据规则评估 EC2 实例。由于该规则是托管规则，因此 `Owner` 属性设置为 `AWS`，`SourceIdentifier` 属性设置为规则标识符 `REQUIRED_TAGS`。对于 `InputParameters` 属性，指定了规则所需的标签键 `CostCenter` 和 `Owner`。

如果命令成功，AWS Config 将不返回任何输出。要验证规则配置，请运行 `describe-config-rules` 命令并指定规则名称。

### 添加客户托管的 Config 规则

以下命令将提供用于添加客户托管 Config 规则的 JSON 代码：

```
aws configservice put-config-rule --config-rule file://InstanceTypesAreT2micro.json
```

`InstanceTypesAreT2micro.json` 是一个包含规则配置的 JSON 文件：

```

{
  "ConfigRuleName": "InstanceTypesAreT2micro",
  "Description": "Evaluates whether EC2 instances are the t2.micro type.",
  "Scope": {
    "ComplianceResourceTypes": [
      "AWS::EC2::Instance"
    ]
  },
  "Source": {
    "Owner": "CUSTOM_LAMBDA",
    "SourceIdentifier": "arn:aws:lambda:us-east-1:123456789012:function:InstanceTypeCheck",
    "SourceDetails": [
      {
        "EventSource": "aws.config",
        "MessageType": "ConfigurationItemChangeNotification"
      }
    ]
  },
  "InputParameters": "{\"desiredInstanceType\":\"t2.micro\"}"
}

```

```
}
```

对于该 `ComplianceResourceTypes` 属性，此 JSON 代码将范围限制为该 `AWS::EC2::Instance` 类型的资源，因此 AWS Config 将仅根据规则评估 EC2 实例。由于此规则是客户托管规则，因此该 `Owner` 属性设置为 `CUSTOM_LAMBDA`，并且该 `SourceIdentifier` 属性设置为 Lambda 函数 AWS 的 ARN。 `SourceDetails` 对象为必填项。当 AWS Config 调用 AWS Lambda 函数根据规则评估资源时，为该 `InputParameters` 属性指定的参数将传递给 Lambda 函数。

如果命令成功，AWS Config 将不返回任何输出。要验证规则配置，请运行 `describe-config-rules` 命令并指定规则名称。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutConfigRule](#) 中的。

## put-configuration-recorder

以下代码示例演示了如何使用 `put-configuration-recorder`。

### AWS CLI

#### 示例 1：记录所有支持的资源

以下命令创建一个配置记录器，用于跟踪对所有支持的资源类型（包括全局资源类型）的更改：

```
aws configservice put-configuration-recorder \  
  --configuration-recorder name=default,roleARN=arn:aws:iam::123456789012:role/  
config-role \  
  --recording-group allSupported=true,includeGlobalResourceTypes=true
```

如果命令成功，AWS Config 将不返回任何输出。要验证配置记录器的设置，请运行 `describe-configuration-recorders` 命令。

#### 示例 2：记录特定类型的资源

以下命令创建一个配置记录器，该记录器仅跟踪在 JSON 文件中为 `--recording-group` 选项指定的资源类型的更改：

```
aws configservice put-configuration-recorder \  
  --configuration-recorder name=default,roleARN=arn:aws:iam::123456789012:role/  
config-role \  
  --recording-group file://recordingGroup.json
```

RecordingGroup.json 是一个 JSON 文件，它指定了 Config AWS 将记录的资源类型：

```
{
  "allSupported": false,
  "includeGlobalResourceTypes": false,
  "resourceTypes": [
    "AWS::EC2::EIP",
    "AWS::EC2::Instance",
    "AWS::EC2::NetworkAcl",
    "AWS::EC2::SecurityGroup",
    "AWS::CloudTrail::Trail",
    "AWS::EC2::Volume",
    "AWS::EC2::VPC",
    "AWS::IAM::User",
    "AWS::IAM::Policy"
  ]
}
```

在为 ResourceTypes 键指定资源类型之前，必须将 AllSupported includeGlobalResource 和 Types 选项设置为 false 或将其省略。

如果命令成功，AWS Config 将不返回任何输出。要验证配置记录器的设置，请运行 describe-configuration-recorders 命令。

示例 3：选择除特定资源类型之外的所有支持的资源

以下命令创建一个配置记录器，用于跟踪所有当前和未来支持的资源类型的更改，但不包括在 JSON 文件中为 --recording-group 选项指定的资源类型：

```
aws configservice put-configuration-recorder \
  --configuration-recorder name=default,roleARN=arn:aws:iam::123456789012:role/
  config-role \
  --recording-group file://recordingGroup.json
```

RecordingGroup.json 是一个 JSON 文件，它指定了 Config AWS 将记录的资源类型：

```
{
  "allSupported": false,
  "exclusionByResourceTypes": {
    "resourceTypes": [
      "AWS::Redshift::ClusterSnapshot",
      "AWS::RDS::DBClusterSnapshot",
    ]
  }
}
```

```

    "AWS::CloudFront::StreamingDistribution"
  ]
},
  "includeGlobalResourceTypes": false,
  "recordingStrategy": {
    "useOnly": "EXCLUSION_BY_RESOURCE_TYPES"
  },
}

```

在指定要从录制中排除的资源类型之前：1) 必须将 `AllSupported` 和 `includeGlobalResourceTypes` 选项设置为 `false` 或将其省略；2) 必须将的 `useOnly` 字段设置为 `EXCLUSION_BY_RESOURCE_TYPES` `RecordingStrategy`。

如果命令成功，AWS Config 将不返回任何输出。要验证配置记录器的设置，请运行 `describe-configuration-recorders` 命令。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutConfigurationRecorder](#)中的。

## put-delivery-channel

以下代码示例演示了如何使用 `put-delivery-channel`。

### AWS CLI

#### 创建配送渠道

以下命令以 JSON 代码的形式提供传送渠道的设置：

```
aws configservice put-delivery-channel --delivery-channel file://
deliveryChannel.json
```

该 `deliveryChannel.json` 文件指定了配送渠道属性：

```

{
  "name": "default",
  "s3BucketName": "config-bucket-123456789012",
  "snsTopicARN": "arn:aws:sns:us-east-1:123456789012:config-topic",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Twelve_Hours"
  }
}

```

此示例设置了以下属性：

`name`- 配送渠道的名称。默认情况下，AWS Config 会 `default` 将名称分配给新的交付渠道。您无法使用命令更新传递渠道名称。`put-delivery-channel` 有关更改名称的步骤，请参阅 [重命名配送渠道](#)。`s3BucketName`- AWS Config 向其发送配置快照和配置历史记录文件的 Amazon S3 存储桶的名称。如果您指定的存储桶属于另一个 AWS 账户，则该存储桶必须具有授予对 Config AWS 的访问权限的策略。有关更多信息，请参阅 [Amazon S3 存储桶的权限](#)。

`snsTopicARN`- Amazon SNS 主题的亚马逊资源名称 (ARN)，AWS Config 会向其发送有关配置变更的通知。如果您从其他账户选择主题，则该主题必须具有授予对 Config 的访问权限的策略。AWS 有关更多信息，请参阅 [Amazon SNS 主题的权限](#)。

`configSnapshotDeliveryProperties`- 包含 `deliveryFrequency` 属性，该属性设置 AWS Config 提供配置快照的频率以及它为定期 Config 规则调用评估的频率。

如果命令成功，AWS Config 将不返回任何输出。要验证您的配送渠道的设置，请运行 `describe-delivery-channels` 命令。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutDeliveryChannel](#) 中的。

## **start-config-rules-evaluation**

以下代码示例演示了如何使用 `start-config-rules-evaluation`。

### AWS CLI

对 AWS Config 规则进行按需评估

以下命令启动对两个 AWS 托管规则的评估：

```
aws configservice start-config-rules-evaluation --config-rule-names s3-bucket-versioning-enabled cloudtrail-enabled
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartConfigRulesEvaluation](#) 中的。

## **start-configuration-recorder**

以下代码示例演示了如何使用 `start-configuration-recorder`。

### AWS CLI

启动配置记录器



以下命令启动默认配置记录器：

```
aws configservice start-configuration-recorder --configuration-recorder-name default
```

如果命令成功，AWS Config 将不返回任何输出。要验证 AWS Config 是否正在记录您的资源，请运行 `get-status` 命令。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartConfigurationRecorder](#)中的。

## stop-configuration-recorder

以下代码示例演示了如何使用 `stop-configuration-recorder`。

### AWS CLI

停止配置记录器

以下命令停止默认配置记录器：

```
aws configservice stop-configuration-recorder --configuration-recorder-name default
```

如果命令成功，AWS Config 将不返回任何输出。要验证 AWS Config 是否未记录您的资源，请运行 `get-status` 命令。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopConfigurationRecorder](#)中的。

## subscribe

以下代码示例演示了如何使用 `subscribe`。

### AWS CLI

订阅 AWS Config

以下命令创建默认传送渠道和配置记录器。该命令还指定 Config 将向其提供 AWS 配置信息的 Amazon S3 存储桶和 Amazon SNS 主题：

```
aws configservice subscribe --s3-bucket config-bucket-123456789012 --  
sns-topic arn:aws:sns:us-east-1:123456789012:config-topic --iam-role  
arn:aws:iam::123456789012:role/ConfigRole-A1B2C3D4E5F6
```

输出：

```
Using existing S3 bucket: config-bucket-123456789012
Using existing SNS topic: arn:aws:sns:us-east-1:123456789012:config-topic
Subscribe succeeded:

Configuration Recorders: [
  {
    "recordingGroup": {
      "allSupported": true,
      "resourceTypes": [],
      "includeGlobalResourceTypes": false
    },
    "roleARN": "arn:aws:iam::123456789012:role/ConfigRole-A1B2C3D4E5F6",
    "name": "default"
  }
]

Delivery Channels: [
  {
    "snsTopicARN": "arn:aws:sns:us-east-1:123456789012:config-topic",
    "name": "default",
    "s3BucketName": "config-bucket-123456789012"
  }
]
```

- 有关 API 详细信息，请参阅《AWS CLI Command Reference》中的 [Subscribe](#)。

## 使用 Amazon Connect 的示例 AWS CLI

以下代码示例向您展示了如何在 Amazon Connect 中 AWS Command Line Interface 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### create-user

以下代码示例演示了如何使用 create-user。

#### AWS CLI

##### 创建用户

以下create-user示例将具有指定属性的用户添加到指定的 Amazon Connect 实例。

```
aws connect create-user \  
  --username Mary \  
  --password Pass@Word1 \  
  --identity-info FirstName=Mary,LastName=Major \  
  --phone-config  
  PhoneType=DESK_PHONE,AutoAccept=true,AfterContactWorkTimeLimit=60,DeskPhoneNumber=  
+15555551212 \  
  --security-profile-id 12345678-1111-2222-aaaa-a1b2c3d4f5g7 \  
  --routing-profile-id 87654321-9999-3434-abcd-x1y2z3a1b2c3 \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "UserId": "87654321-2222-1234-1234-111234567891",  
  "UserArn": "arn:aws:connect:us-west-2:123456789012:instance/a1b2c3d4-5678-90ab-  
cdef-EXAMPLE11111/agent/87654321-2222-1234-1234-111234567891"  
}
```

有关更多信息，请参阅 Amazon Connect 管理员指南中的[添加用户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUser](#)中的。

### delete-user

以下代码示例演示了如何使用 delete-user。

#### AWS CLI

##### 删除用户

以下delete-user示例从指定的 Amazon Connect 实例中删除指定用户。

```
aws connect delete-user \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --user-id 87654321-2222-1234-1234-111234567891
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Connect 管理员指南中的[管理用户](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteUser](#) 中的。

## describe-user-hierarchy-group

以下代码示例演示了如何使用 describe-user-hierarchy-group。

### AWS CLI

显示层次结构组的详细信息

以下 describe-user-hierarchy-group 示例显示了指定 Amazon Connect 层次结构组的详细信息。

```
aws connect describe-user-hierarchy-group \  
  --hierarchy-group-id 12345678-1111-2222-800e-aaabbb555gg \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "HierarchyGroup": {  
    "Id": "12345678-1111-2222-800e-a2b3c4d5f6g7",  
    "Arn": "arn:aws:connect:us-west-2:123456789012:instance/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/agent-group/12345678-1111-2222-800e-a2b3c4d5f6g7",  
    "Name": "Example Corporation",  
    "LevelId": "1",  
    "HierarchyPath": {  
      "LevelOne": {  
        "Id": "abcdefgh-3333-4444-8af3-201123456789",  
        "Arn": "arn:aws:connect:us-west-2:123456789012:instance/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/agent-group/abcdefgh-3333-4444-8af3-201123456789",  
        "Name": "Example Corporation"  
      }  
    }  
  }  
}
```

```
}  
}
```

有关更多信息，请参阅 Amazon Connect 管理员指南中的[设置代理层次结构](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeUserHierarchyGroup](#) 中的。

## describe-user-hierarchy-structure

以下代码示例演示了如何使用 describe-user-hierarchy-structure。

### AWS CLI

显示层次结构的详细信息

以下 describe-user-hierarchy-structure 示例显示了指定 Amazon Connect 实例的层次结构的详细信息。

```
aws connect describe-user-hierarchy-group \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "HierarchyStructure": {  
    "LevelOne": {  
      "Id": "12345678-1111-2222-800e-aaabbb555gg",  
      "Arn": "arn:aws:connect:us-west-2:123456789012:instance/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/agent-group-level/1",  
      "Name": "Corporation"  
    },  
    "LevelTwo": {  
      "Id": "87654321-2222-3333-ac99-123456789102",  
      "Arn": "arn:aws:connect:us-west-2:123456789012:instance/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/agent-group-level/2",  
      "Name": "Services Division"  
    },  
    "LevelThree": {  
      "Id": "abcdefgh-3333-4444-8af3-201123456789",  
      "Arn": "arn:aws:connect:us-west-2:123456789012:instance/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/agent-group-level/3",  
      "Name": "EU Site"  
    }  
  }  
}
```

```
}  
}
```

有关更多信息，请参阅 Amazon Connect 管理员指南中的[设置代理层次结构](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeUserHierarchyStructure](#) 中的。

## describe-user

以下代码示例演示了如何使用 describe-user。

### AWS CLI

显示用户的详细信息

以下 describe-user 示例显示了指定 Amazon Connect 用户的详细信息。

```
aws connect describe-user \  
  --user-id 0c245dc0-0cf5-4e37-800e-2a7481cc8a60 \  
  --instance-id 40c83b68-ea62-414c-97bb-d018e39e158e
```

输出：

```
{  
  "User": {  
    "Id": "0c245dc0-0cf5-4e37-800e-2a7481cc8a60",  
    "Arn": "arn:aws:connect:us-west-2:123456789012:instance/40c83b68-  
ea62-414c-97bb-d018e39e158e/agent/0c245dc0-0cf5-4e37-800e-2a7481cc8a60",  
    "Username": "Jane",  
    "IdentityInfo": {  
      "FirstName": "Jane",  
      "LastName": "Doe",  
      "Email": "example.com"  
    },  
    "PhoneConfig": {  
      "PhoneType": "SOFT_PHONE",  
      "AutoAccept": false,  
      "AfterContactWorkTimeLimit": 0,  
      "DeskPhoneNumber": ""  
    },  
    "DirectoryUserId": "8b444cf6-b368-4f29-ba18-07af27405658",  
    "SecurityProfileIds": [  
      "b6f85a42-1dc5-443b-b621-de0abf70c9cf"  
    ]  
  }  
}
```

```
    ],  
    "RoutingProfileId": "0be36ee9-2b5f-4ef4-bcf7-87738e5be0e5",  
    "Tags": {}  
  }  
}
```

有关更多信息，请参阅 Amazon Connect 管理员指南中的[管理用户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeUser](#)中的。

## get-contact-attributes

以下代码示例演示了如何使用 get-contact-attributes。

### AWS CLI

检索联系人的属性

以下get-contact-attributes示例检索为指定的 Amazon Connect 联系人设置的属性。

```
aws connect get-contact-attributes \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --initial-contact-id 12345678-1111-2222-800e-a2b3c4d5f6g7
```

输出：

```
{  
  "Attributes": {  
    "greetingPlayed": "true"  
  }  
}
```

有关更多信息，请参阅 [Amazon Connect 管理员指南中的使用 Amazon Connect 联系人属性](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetContactAttributes](#)中的。

## list-contact-flows

以下代码示例演示了如何使用 list-contact-flows。

### AWS CLI

列出实例中的联系人流

以下 `list-contact-flows` 示例列出了指定 Amazon Connect 实例中的联系流程。

```
aws connect list-contact-flows \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "ContactFlowSummaryList": [  
    {  
      "Id": "12345678-1111-2222-800e-a2b3c4d5f6g7",  
      "Arn": "arn:aws:connect:us-west-2:123456789012:instance/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/contact-flow/12345678-1111-2222-800e-  
a2b3c4d5f6g7",  
      "Name": "Default queue transfer",  
      "ContactFlowType": "QUEUE_TRANSFER"  
    },  
    {  
      "Id": "87654321-2222-3333-ac99-123456789102",  
      "Arn": "arn:aws:connect:us-west-2:123456789012:instance/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/contact-flow/87654321-2222-3333-  
ac99-123456789102",  
      "Name": "Default agent hold",  
      "ContactFlowType": "AGENT_HOLD"  
    },  
    {  
      "Id": "abcdefgh-3333-4444-8af3-201123456789",  
      "Arn": "arn:aws:connect:us-west-2:123456789012:instance/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/contact-flow/  
abcdefgh-3333-4444-8af3-201123456789",  
      "Name": "Default customer hold",  
      "ContactFlowType": "CUSTOMER_HOLD"  
    }  
  ]  
}
```

有关更多信息，请参阅 [Amazon Connect 管理员指南中的创建 Amazon Connect 联系人流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListContactFlows](#) 中的。

## list-hours-of-operations

以下代码示例演示了如何使用 `list-hours-of-operations`。



## AWS CLI

### 列出实例的运行时间

以下`list-hours-of-operations`示例列出了指定 Amazon Connect 实例的运行时间。

```
aws connect list-hours-of-operations \  
  --instance-id 40c83b68-ea62-414c-97bb-d018e39e158e
```

输出：

```
{  
  "HoursOfOperationSummaryList": [  
    {  
      "Id": "d69f1f84-7457-4924-8fbe-e64875546259",  
      "Arn": "arn:aws:connect:us-west-2:123456789012:instance/40c83b68-  
ea62-414c-97bb-d018e39e158e/operating-hours/d69f1f84-7457-4924-8fbe-e64875546259",  
      "Name": "Basic Hours"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon Connect 管理员指南中的[设置队列的操作时间](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListHoursOfOperations](#)中的。

## list-phone-numbers

以下代码示例演示了如何使用 `list-phone-numbers`。

### AWS CLI

#### 列出实例中的电话号码

以下`list-phone-numbers`示例列出了指定 Amazon Connect 实例中的电话号码。

```
aws connect list-phone-numbers \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
```

```
"PhoneNumberSummaryList": [  
  {  
    "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "Arn": "arn:aws:connect:us-west-2:123456789012:instance/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/phone-number/xyz80zxy-xyz1-80zx-  
zx80-11111EXAMPLE",  
    "PhoneNumber": "+17065551212",  
    "PhoneNumberType": "DID",  
    "PhoneNumberCountryCode": "US"  
  },  
  {  
    "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
    "Arn": "arn:aws:connect:us-west-2:123456789012:instance/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/phone-number/ccc0ccc-xyz1-80zx-  
zx80-22222EXAMPLE",  
    "PhoneNumber": "+18555551212",  
    "PhoneNumberType": "TOLL_FREE",  
    "PhoneNumberCountryCode": "US"  
  }  
]  
}
```

有关更多信息，请参阅 Amazon Connect 管理员指南中的[为您的联络中心设置电话号码](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListPhoneNumbers](#)中的。

## list-queues

以下代码示例演示了如何使用 list-queues。

### AWS CLI

列出实例中的队列

以下 list-queues 示例列出了指定 Amazon Connect 实例中的队列。

```
aws connect list-queues \  
--instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
```

```
"QueueSummaryList": [  
  {  
    "Id": "12345678-1111-2222-800e-a2b3c4d5f6g7",  
    "Arn": "arn:aws:connect:us-west-2:123456789012:instance/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/queue/agent/12345678-1111-2222-800e-  
a2b3c4d5f6g7",  
    "QueueType": "AGENT"  
  },  
  {  
    "Id": "87654321-2222-3333-ac99-123456789102",  
    "Arn": "arn:aws:connect:us-west-2:123456789012:instance/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/queue/agent/87654321-2222-3333-  
ac99-123456789102",  
    "QueueType": "AGENT"  
  },  
  {  
    "Id": "abcdefgh-3333-4444-8af3-201123456789",  
    "Arn": "arn:aws:connect:us-west-2:123456789012:instance/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/queue/agent/  
abcdefgh-3333-4444-8af3-201123456789",  
    "QueueType": "AGENT"  
  },  
  {  
    "Id": "hgfedcba-4444-5555-a31f-123456789102",  
    "Arn": "arn:aws:connect:us-west-2:123456789012:instance/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/queue/hgfedcba-4444-5555-a31f-123456789102",  
    "Name": "BasicQueue",  
    "QueueType": "STANDARD"  
  },  
]  
}
```

有关更多信息，请参阅 Amazon Connect 管理员指南中的[创建队列](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListQueues](#)中的。

## list-routing-profiles

以下代码示例演示了如何使用 list-routing-profiles。

### AWS CLI

列出实例中的路由配置文件

以下 `list-routing-profiles` 示例列出了指定 Amazon Connect 实例中的路由配置文件。

```
aws connect list-routing-profiles \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "RoutingProfileSummaryList": [  
    {  
      "Id": "12345678-1111-2222-800e-a2b3c4d5f6g7",  
      "Arn": "arn:aws:connect:us-west-2:123456789012:instance/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/routing-profile/12345678-1111-2222-800e-  
a2b3c4d5f6g7",  
      "Name": "Basic Routing Profile"  
    },  
  ]  
}
```

有关更多信息，请参阅 Amazon Connect 管理员指南中的 [创建路由配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListRoutingProfiles](#) 中的。

## **list-security-profiles**

以下代码示例演示了如何使用 `list-security-profiles`。

AWS CLI

列出实例中的安全配置文件

以下 `list-security-profiles` 示例列出了指定 Amazon Connect 实例中的安全配置文件。

```
aws connect list-security-profiles \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "SecurityProfileSummaryList": [  
    {  
      "Id": "12345678-1111-2222-800e-a2b3c4d5f6g7",
```

```

    "Arn": "arn:aws:connect:us-west-2:123456789012:instance/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/security-profile/12345678-1111-2222-800e-
a2b3c4d5f6g7",
    "Name": "CallCenterManager"
  },
  {
    "Id": "87654321-2222-3333-ac99-123456789102",
    "Arn": "arn:aws:connect:us-west-2:123456789012:instance/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/security-profile/87654321-2222-3333-
ac99-123456789102",
    "Name": "QualityAnalyst"
  },
  {
    "Id": "abcdefgh-3333-4444-8af3-201123456789",
    "Arn": "arn:aws:connect:us-west-2:123456789012:instance/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/security-profile/
abcdefgh-3333-4444-8af3-201123456789",
    "Name": "Agent"
  },
  {
    "Id": "12345678-1111-2222-800e-x2y3c4d5fzzzz",
    "Arn": "arn:aws:connect:us-west-2:123456789012:instance/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/security-profile/12345678-1111-2222-800e-
x2y3c4d5fzzzz",
    "Name": "Admin"
  }
]
}

```

有关更多信息，请参阅 Amazon Connect 管理员指南中的[分配权限：安全配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListSecurityProfiles](#)中的。

## list-user-hierarchy-groups

以下代码示例演示了如何使用 list-user-hierarchy-groups。

### AWS CLI

列出实例中的用户层次结构组

以下 list-user-hierarchy-groups 示例列出了指定 Amazon Connect 实例中的用户层次结构群组。

```
aws connect list-user-hierarchy-groups \  
  --instance-id 40c83b68-ea62-414c-97bb-d018e39e158e
```

输出：

```
{  
  "UserHierarchyGroupSummaryList": [  
    {  
      "Id": "0e2f6d1d-b3ca-494b-8dbc-ba81d9f8182a",  
      "Arn": "arn:aws:connect:us-west-2:123456789012:instance/40c83b68-  
ea62-414c-97bb-d018e39e158e/agent-group/0e2f6d1d-b3ca-494b-8dbc-ba81d9f8182a",  
      "Name": "Example Corporation"  
    },  
  ]  
}
```

有关更多信息，请参阅 Amazon Connect 管理员指南中的[设置代理层次结构](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListUserHierarchyGroups](#)中的。

## list-users

以下代码示例演示了如何使用 `list-users`。

### AWS CLI

列出实例中的用户层次结构组

以下 `list-users` 示例列出了指定 Amazon Connect 实例中的用户。

```
aws connect list-users \  
  --instance-id 40c83b68-ea62-414c-97bb-d018e39e158e
```

输出：

```
{  
  "UserSummaryList": [  
    {  
      "Id": "0c245dc0-0cf5-4e37-800e-2a7481cc8a60",  
      "Arn": "arn:aws:connect:us-west-2:123456789012:instance/40c83b68-  
ea62-414c-97bb-d018e39e158e/agent/0c245dc0-0cf5-4e37-800e-2a7481cc8a60",  
      "Username": "Jane"  
    }  
  ]  
}
```

```
    },
    {
      "Id": "46f0c67c-3fc7-4806-ac99-403798788c14",
      "Arn": "arn:aws:connect:us-west-2:123456789012:instance/40c83b68-
ea62-414c-97bb-d018e39e158e/agent/46f0c67c-3fc7-4806-ac99-403798788c14",
      "Username": "Paulo"
    },
    {
      "Id": "55a83578-95e1-4710-8af3-2b7afe310e48",
      "Arn": "arn:aws:connect:us-west-2:123456789012:instance/40c83b68-
ea62-414c-97bb-d018e39e158e/agent/55a83578-95e1-4710-8af3-2b7afe310e48",
      "Username": "JohnD"
    },
    {
      "Id": "703e27b5-c9f0-4f1f-a239-64ccbb160125",
      "Arn": "arn:aws:connect:us-west-2:123456789012:instance/40c83b68-
ea62-414c-97bb-d018e39e158e/agent/703e27b5-c9f0-4f1f-a239-64ccbb160125",
      "Username": "JohnS"
    }
  ]
}
```

有关更多信息，请参阅 Amazon Connect 管理员指南中的[添加用户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUsers](#)中的。

## update-contact-attributes

以下代码示例演示了如何使用 update-contact-attributes。

### AWS CLI

#### 更新联系人的属性

以下update-contact-attributes示例更新了指定的 Amazon Connect 用户的greetingPlayed属性。

```
aws connect update-contact-attributes \  
  --initial-contact-id 11111111-2222-3333-4444-12345678910 \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --attributes greetingPlayed=false
```

此命令不生成任何输出。

有关更多信息，请参阅 [Amazon Connect 管理员指南中的使用 Amazon Connect 联系人属性](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateContactAttributes](#) 中的。

## update-user-hierarchy

以下代码示例演示了如何使用 update-user-hierarchy。

### AWS CLI

更新用户的层次结构

以下 update-user-hierarchy 示例更新了指定 Amazon Connect 用户的代理层次结构。

```
aws connect update-user-hierarchy \  
  --hierarchy-group-id 12345678-a1b2-c3d4-e5f6-123456789abc \  
  --user-id 87654321-2222-1234-1234-111234567891 \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Connect 管理员指南中的 [配置代理设置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateUserHierarchy](#) 中的。

## update-user-identity-info

以下代码示例演示了如何使用 update-user-identity-info。

### AWS CLI

更新用户的身份信息

以下 update-user-identity-info 示例更新了指定 Amazon Connect 用户的身份信息。

```
aws connect update-user-identity-info \  
  --identity-info FirstName=Mary,LastName=Major,Email=marym@example.com \  
  --user-id 87654321-2222-1234-1234-111234567891 \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Connect 管理员指南中的 [配置代理设置](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUserIdentityInfo](#)中的。

## update-user-phone-config

以下代码示例演示了如何使用 update-user-phone-config。

### AWS CLI

#### 更新用户的电话配置

以下update-user-phone-config示例更新了指定用户的电话配置。

```
aws connect update-user-phone-config \  
  --phone-config  
  PhoneType=SOFT_PHONE,AutoAccept=false,AfterContactWorkTimeLimit=60,DeskPhoneNumber=  
+18005551212 \  
  --user-id 12345678-4444-3333-2222-111122223333 \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Connect 管理员指南中的[配置代理设置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUserPhoneConfig](#)中的。

## update-user-routing-profile

以下代码示例演示了如何使用 update-user-routing-profile。

### AWS CLI

#### 更新用户的路由配置文件

以下update-user-routing-profile示例更新了指定 Amazon Connect 用户的路由配置文件。

```
aws connect update-user-routing-profile \  
  --routing-profile-id 12345678-1111-3333-2222-4444EXAMPLE \  
  --user-id 87654321-2222-1234-1234-111234567891 \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Connect 管理员指南中的[配置代理设置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUserRoutingProfile](#)中的。

## update-user-security-profiles

以下代码示例演示了如何使用 update-user-security-profiles。

### AWS CLI

#### 更新用户的安全配置文件

以下update-user-security-profiles示例更新了指定 Amazon Connect 用户的安全配置文件。

```
aws connect update-user-security-profiles \  
  --security-profile-ids 12345678-1234-1234-1234-1234567892111 \  
  --user-id 87654321-2222-1234-1234-111234567891 \  
  --instance-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Connect 管理员指南中的[分配权限：安全配置文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUserSecurityProfiles](#)中的。

## AWS 成本和使用情况报告 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS 成本和使用情况报告。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### **delete-report-definition**

以下代码示例演示了如何使用 `delete-report-definition`。

#### AWS CLI

删除 AWS 成本和使用情况报告

此示例删除了 AWS 成本和使用情况报告。

命令:

```
aws cur --region us-east-1 delete-report-definition --report-name "ExampleReport"
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteReportDefinition](#) 中的。

### **describe-report-definitions**

以下代码示例演示了如何使用 `describe-report-definitions`。

#### AWS CLI

检索 AWS 成本和使用情况报告列表

此示例描述了账户拥有 AWS 的成本和使用情况报告列表。

命令:

```
aws cur --region us-east-1 describe-report-definitions --max-items 5
```

输出:

```
{
  "ReportDefinitions": [
    {
      "ReportName": "ExampleReport",
      "Compression": "ZIP",
      "S3Region": "us-east-1",
      "Format": "textORcsv",

```

```
    "S3Prefix": "exampleprefix",
    "S3Bucket": "example-s3-bucket",
    "TimeUnit": "DAILY",
    "AdditionalArtifacts": [
      "REDSHIFT",
      "QUICKSIGHT"
    ],
    "AdditionalSchemaElements": [
      "RESOURCES"
    ]
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeReportDefinitions](#)中的。

## put-report-definition

以下代码示例演示了如何使用 put-report-definition。

### AWS CLI

#### 创建 AWS 成本和使用情况报告

以下put-report-definition示例创建了每日 AWS 成本和使用情况报告，您可以将其上传到亚马逊 Redshift 或亚马逊。QuickSight

```
aws cur put-report-definition --report-definition file://report-definition.json
```

report-definition.json 的内容：

```
{
  "ReportName": "ExampleReport",
  "TimeUnit": "DAILY",
  "Format": "textORcsv",
  "Compression": "ZIP",
  "AdditionalSchemaElements": [
    "RESOURCES"
  ],
  "S3Bucket": "example-s3-bucket",
  "S3Prefix": "exampleprefix",
  "S3Region": "us-east-1",
```

```
    "AdditionalArtifacts": [  
        "REDSHIFT",  
        "QUICKSIGHT"  
    ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutReportDefinition](#)中的。

## 使用 Cost Explorer 服务示例 AWS CLI

以下代码示例向您展示了如何使用 with Cost Explorer 服务来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### get-cost-and-usage

以下代码示例演示了如何使用 get-cost-and-usage。

AWS CLI

检索账户在 2017 年 9 月的 S3 使用情况

以下get-cost-and-usage示例检索账户在 2017 年 9 月的 S3 使用情况。

```
aws ce get-cost-and-usage \  
  --time-period Start=2017-09-01,End=2017-10-01 \  
  --granularity MONTHLY \  
  --metrics "BlendedCost" "UnblendedCost" "UsageQuantity" \  
  --
```

```
--group-by Type=DIMENSION,Key=SERVICE Type=TAG,Key=Environment \  
--filter file://filters.json
```

filters.json 的内容：

```
{  
  "Dimensions": {  
    "Key": "SERVICE",  
    "Values": [  
      "Amazon Simple Storage Service"  
    ]  
  }  
}
```

输出：

```
{  
  "GroupDefinitions": [  
    {  
      "Type": "DIMENSION",  
      "Key": "SERVICE"  
    },  
    {  
      "Type": "TAG",  
      "Key": "Environment"  
    }  
  ],  
  "ResultsByTime": [  
    {  
      "Estimated": false,  
      "TimePeriod": {  
        "Start": "2017-09-01",  
        "End": "2017-10-01"  
      },  
      "Total": {},  
      "Groups": [  
        {  
          "Keys": [  
            "Amazon Simple Storage Service",  
            "Environment$"  
          ],  
          "Metrics": {  
            "BlendedCost": {
```

```

        "Amount": "40.3527508453",
        "Unit": "USD"
    },
    "UnblendedCost": {
        "Amount": "40.3543773134",
        "Unit": "USD"
    },
    "UsageQuantity": {
        "Amount": "9312771.098461578",
        "Unit": "N/A"
    }
}
},
{
    "Keys": [
        "Amazon Simple Storage Service",
        "Environment$Dev"
    ],
    "Metrics": {
        "BlendedCost": {
            "Amount": "0.2682364644",
            "Unit": "USD"
        },
        "UnblendedCost": {
            "Amount": "0.2682364644",
            "Unit": "USD"
        },
        "UsageQuantity": {
            "Amount": "22403.4395271182",
            "Unit": "N/A"
        }
    }
}
]
}
]
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCostAndUsage](#)中的。

## get-dimension-values

以下代码示例演示了如何使用 get-dimension-values。

## AWS CLI

检索维度服务的标签，值为“Elastic”

此示例检索 2017 年 1 月 1 日至 2017 年 5 月 18 日的维度服务的标签，其值为“Elastic”。

命令：

```
aws ce get-dimension-values --search-string Elastic --time-period
Start=2017-01-01,End=2017-05-18 --dimension SERVICE
```

输出：

```
{
  "TotalSize": 6,
  "DimensionValues": [
    {
      "Attributes": {},
      "Value": "Amazon ElastiCache"
    },
    {
      "Attributes": {},
      "Value": "EC2 - Other"
    },
    {
      "Attributes": {},
      "Value": "Amazon Elastic Compute Cloud - Compute"
    },
    {
      "Attributes": {},
      "Value": "Amazon Elastic Load Balancing"
    },
    {
      "Attributes": {},
      "Value": "Amazon Elastic MapReduce"
    },
    {
      "Attributes": {},
      "Value": "Amazon Elasticsearch Service"
    }
  ],
  "ReturnSize": 6
}
```



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDimensionValues](#)中的。

## get-reservation-coverage

以下代码示例演示了如何使用 get-reservation-coverage。

### AWS CLI

检索 us-east-1 区域中 EC2 t2.nano 实例的预留覆盖范围

此示例检索 2017 年 7 月至 9 月在 us-east-1 区域的 EC2 t2.nano 实例的预留覆盖范围。

命令：

```
aws ce get-reservation-coverage --time-period Start=2017-07-01,End=2017-10-01 --group-by Type=Dimension,Key=REGION --filter file://filters.json
```

filters.json :

```
{
  "And": [
    {
      "Dimensions": {
        "Key": "INSTANCE_TYPE",
        "Values": [
          "t2.nano"
        ]
      },
      "Dimensions": {
        "Key": "REGION",
        "Values": [
          "us-east-1"
        ]
      }
    }
  ]
}
```

输出：

```
{
  "TotalSize": 6,
```

```
"DimensionValues": [
  {
    "Attributes": {},
    "Value": "Amazon ElastiCache"
  },
  {
    "Attributes": {},
    "Value": "EC2 - Other"
  },
  {
    "Attributes": {},
    "Value": "Amazon Elastic Compute Cloud - Compute"
  },
  {
    "Attributes": {},
    "Value": "Amazon Elastic Load Balancing"
  },
  {
    "Attributes": {},
    "Value": "Amazon Elastic MapReduce"
  },
  {
    "Attributes": {},
    "Value": "Amazon Elasticsearch Service"
  }
],
"ReturnSize": 6
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetReservationCoverage](#)中的。

## get-reservation-purchase-recommendation

以下代码示例演示了如何使用 `get-reservation-purchase-recommendation`。

### AWS CLI

检索期限为三年的部分预付 EC2 RI 的预留建议

以下`get-reservation-purchase-recommendation`示例根据过去 60 天的 EC2 使用情况，检索针对期限为三年的部分预付 EC2 实例的建议。

```
aws ce get-reservation-purchase-recommendation \
```

```
--service "Amazon Redshift" \  
--lookback-period-in-days SIXTY_DAYS \  
--term-in-years THREE_YEARS \  
--payment-option PARTIAL_UPFRONT
```

输出：

```
{  
  "Recommendations": [],  
  "Metadata": {  
    "GenerationTimestamp": "2018-08-08T15:20:57Z",  
    "RecommendationId": "00d59dde-a1ad-473f-8ff2-iexample3330b"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetReservationPurchaseRecommendation](#)中的。

## get-reservation-utilization

以下代码示例演示了如何使用 get-reservation-utilization。

### AWS CLI

检索您账户的预订使用情况

以下get-reservation-utilization示例检索该账户在 2018-03-01 到 2018-08-01 期间所有 t2.nano 实例类型的预留实例使用率。

```
aws ce get-reservation-utilization \  
--time-period Start=2018-03-01,End=2018-08-01 \  
--filter file://filters.json
```

filters.json 的内容：

```
{  
  "Dimensions": {  
    "Key": "INSTANCE_TYPE",  
    "Values": [  
      "t2.nano"  
    ]  
  }  
}
```

```
}
}
```

输出：

```
{
  "Total": {
    "TotalAmortizedFee": "0",
    "UtilizationPercentage": "0",
    "PurchasedHours": "0",
    "NetRISavings": "0",
    "TotalActualHours": "0",
    "AmortizedRecurringFee": "0",
    "UnusedHours": "0",
    "TotalPotentialRISavings": "0",
    "OnDemandCostOfRIHoursUsed": "0",
    "AmortizedUpfrontFee": "0"
  },
  "UtilizationsByTime": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetReservationUtilization](#)中的。

## get-tags

以下代码示例演示了如何使用 get-tags。

### AWS CLI

检索成本分配标签的密钥和值

此示例检索密钥为“项目”且值包含“SecretProject”的所有成本分配标签。

命令：

```
aws ce get-tags --search-string secretProject --time-period
Start=2017-01-01,End=2017-05-18 --tag-key Project
```

输出：

```
{
```

```
"ReturnSize": 2,
"Tags": [
  "secretProject1",
  "secretProject2"
],
"TotalSize": 2
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTags](#)中的。

## 使用的 Firehose 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Firehose 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **list-delivery-streams**

以下代码示例演示了如何使用 list-delivery-streams。

#### AWS CLI

##### 列出可用的直播流

以下list-delivery-streams示例列出了您 AWS 账户中可用的直播流。

```
aws firehose list-delivery-streams
```

输出：

```
{
  "DeliveryStreamNames": [
    "my-stream"
  ],
  "HasMoreDeliveryStreams": false
}
```

有关更多信息，请参阅 Amazon Kinesis Data Firehose 开发人员指南中的[创建 Amazon Kinesis Data Firehose 传输流](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListDeliveryStreams](#)中的。

## put-record-batch

以下代码示例演示了如何使用 put-record-batch。

### AWS CLI

将多条记录写入流中

以下 put-record-batch 示例将三条记录写入流中。数据以 Base64 格式编码。

```
aws firehose put-record-batch \
  --delivery-stream-name my-stream \
  --records file://records.json
```

myfile.json 的内容：

```
[
  {"Data": "Rmlyc3QgdGhpbmc="},
  {"Data": "U2Vjb25kIHRoaW5n"},
  {"Data": "VGhpcmQgdGhpbmc="}
]
```

输出：

```
{
  "FailedPutCount": 0,
  "Encrypted": false,
```

```

    "RequestResponses": [
      {
        "RecordId": "9D20J6t2EqCTZTXwGzeSv/EVHxRoRCw89xd+o3+sXg8DhY0aWKPSmZy/
CGlRVEys1u1xbeKh6VofEYKkoeiDrcjrxhQp9iF7sUW7pujiMEQ5LzlrzCkGosxQn
+3boDnURDEaD42V7Giixp0yLJkYZcae1i7HzlCEoy9LJhMr8EjDSi40m/9Vc2uhwwuAtGE0XKpxJ2WD7ZRwtAnY1KAnv
      },
      {
        "RecordId": "jFirejqxCLlK5xjH/UNmLMVcjkTEN76I7916X9PaZ
+PVa0SXDFu1WG0qEZhxq2js7xcZ552eoeDxsuTU1MSq9nZTbVfb6cQTIXnm/GsuF37Uhg67GkmR5z9016XKJ
+ /+pDloFv7Hh9a3oUS6wYm3DcNRLTHHAimANp1PhkQvWpvLRfzbuCUkBphR2QVzhP90iHLbzGwy8/
DfH8sqWEUYASNJKS8GXP5s"
      },
      {
        "RecordId":
        "oy0amQ40o5Y2YV4vxzufdcM00w6n3EPi3tpPJGoYVVKH4APPVqNcbUgefo1stEFRg4hTLrf2k6eliHu/9+YJ5R3iie
DTBt3qBlmTj7Xq8SKVb01S7YvMTpWkMKA86f8JfmT8BMKoMb4XZS/s0kQLe+qh0sYKXWl"
      }
    ]
  }

```

有关更多信息，请参阅《Amazon Kinesis Data Firehose 开发人员指南》中的[将数据发送到 Amazon Kinesis Data Firehose 传输流](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[PutRecordBatch](#)中的。

## put-record

以下代码示例演示了如何使用 `put-record`。

### AWS CLI

向直播中写入记录

以下 `put-record` 示例将数据写入流。数据以 Base64 格式编码。

```

aws firehose put-record \
  --delivery-stream-name my-stream \
  --record '{"Data": "SGVsbG8gd29ybGQ="}'

```

输出：

```
{
```

```
"RecordId": "RjB5K/nnoGFHqwTsZ1Nd/
TTqvjE8V5dsyXZTQn2JXrdpMT0wssyEb6nfC8fwf1whhwnItt4mvrn+gsqeK5jB7QjuLg283+Ps4Sz/
j1Xujv31iDhnPdaLw4B0yM9Amv7PcCuB2079RuM0NhoakbyUymLwY8yt20G8X2420wu1j1Fafhci4erAt7QhDEvpwuK8
"Encrypted": false
}
```

有关更多信息，请参阅《Amazon Kinesis Data Firehose 开发人员指南》中的[将数据发送到 Amazon Kinesis Data Firehose 传输流](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutRecord](#) 中的。

## 使用 Amazon Data Lifecycle Manager 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon Data Lifecycle Manager 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **create-default-role**

以下代码示例演示了如何使用 create-default-role。

#### AWS CLI

为 Amazon DLM 创建所需的 IAM 角色

以下 dlm create-default-role 示例创建了用于管理快照的 AWS DataLifecycleManagerDefaultRole 默认角色。

```
aws dlm create-default-role \
```



```
--resource-type snapshot
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊弹性计算云用户指南》中的 Amazon Data Lifecycle Manager 的 [默认服务角色](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDefaultRole](#) 中的。

## create-lifecycle-policy

以下代码示例演示了如何使用 create-lifecycle-policy。

### AWS CLI

#### 创建生命周期策略

以下 create-lifecycle-policy 示例创建了一个生命周期策略，该策略在指定时间创建卷的每日快照。指定的标签将添加到快照中，标签也将从卷中复制并添加到快照中。如果创建的新快照超过指定的最大数量，则最旧的快照将被删除。

```
aws dlm create-lifecycle-policy \  
  --description "My first policy" \  
  --state ENABLED \  
  --execution-role-arn arn:aws:iam::12345678910:role/  
AWSDataLifecycleManagerDefaultRole \  
  --policy-details file://policyDetails.json
```

policyDetails.json 的内容：

```
{  
  "ResourceTypes": [  
    "VOLUME"  
  ],  
  "TargetTags": [  
    {  
      "Key": "costCenter",  
      "Value": "115"  
    }  
  ],  
  "Schedules": [  
    {
```

```
    "Name": "DailySnapshots",
    "CopyTags": true,
    "TagsToAdd": [
      {
        "Key": "type",
        "Value": "myDailySnapshot"
      }
    ],
    "CreateRule": {
      "Interval": 24,
      "IntervalUnit": "HOURS",
      "Times": [
        "03:00"
      ]
    },
    "RetainRule": {
      "Count": 5
    }
  }
]
```

输出：

```
{
  "PolicyId": "policy-0123456789abcdef0"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLifecyclePolicy](#)中的。

## delete-lifecycle-policy

以下代码示例演示了如何使用 delete-lifecycle-policy。

### AWS CLI

#### 删除生命周期策略

以下示例删除了指定的生命周期策略。：

```
aws dlm delete-lifecycle-policy --policy-id policy-0123456789abcdef0
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLifecyclePolicy](#)中的。

## get-lifecycle-policies

以下代码示例演示了如何使用 get-lifecycle-policies。

### AWS CLI

获取生命周期政策摘要

以下get-lifecycle-policies示例列出了您的所有生命周期策略。

```
aws dlm get-lifecycle-policies
```

输出：

```
{
  "Policies": [
    {
      "PolicyId": "policy-0123456789abcdef0",
      "Description": "My first policy",
      "State": "ENABLED"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLifecyclePolicies](#)中的。

## get-lifecycle-policy

以下代码示例演示了如何使用 get-lifecycle-policy。

### AWS CLI

描述生命周期策略

以下get-lifecycle-policy示例显示了指定生命周期策略的详细信息。

```
aws dlm get-lifecycle-policy \
  --policy-id policy-0123456789abcdef0
```

输出：

```
{
  "Policy": {
    "PolicyId": "policy-0123456789abcdef0",
    "Description": "My policy",
    "State": "ENABLED",
    "ExecutionRoleArn": "arn:aws:iam::123456789012:role/
AWSDataLifecycleManagerDefaultRole",
    "DateCreated": "2019-08-08T17:45:42Z",
    "DateModified": "2019-08-08T17:45:42Z",
    "PolicyDetails": {
      "PolicyType": "EBS_SNAPSHOT_MANAGEMENT",
      "ResourceTypes": [
        "VOLUME"
      ],
      "TargetTags": [
        {
          "Key": "costCenter",
          "Value": "115"
        }
      ],
      "Schedules": [
        {
          "Name": "DailySnapshots",
          "CopyTags": true,
          "TagsToAdd": [
            {
              "Key": "type",
              "Value": "myDailySnapshot"
            }
          ],
          "CreateRule": {
            "Interval": 24,
            "IntervalUnit": "HOURS",
            "Times": [
              "03:00"
            ]
          },
          "RetainRule": {
            "Count": 5
          }
        }
      ]
    }
  }
}
```

```
    }  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLifecyclePolicy](#)中的。

## update-lifecycle-policy

以下代码示例演示了如何使用 update-lifecycle-policy。

### AWS CLI

#### 示例 1：启用生命周期策略

以下update-lifecycle-policy示例启用了指定的生命周期策略。

```
aws dlm update-lifecycle-policy \  
  --policy-id policy-0123456789abcdef0 \  
  --state ENABLED
```

#### 示例 2：禁用生命周期策略

以下update-lifecycle-policy示例禁用了指定的生命周期策略。

```
aws dlm update-lifecycle-policy \  
  --policy-id policy-0123456789abcdef0 \  
  --state DISABLED
```

#### 示例 3：更新生命周期策略的详细信息

以下update-lifecycle-policy示例更新了指定生命周期策略的目标标签。

```
aws dlm update-lifecycle-policy \  
  --policy-id policy-0123456789abcdef0 \  
  --policy-details file://policyDetails.json
```

policyDetails.json 的内容。该命令不会更改此文件中未引用的其他细节。

```
{  
  "TargetTags": [  
    {  
      "Key": "Environment",  
      "Value": "Production"  
    }  
  ]  
}
```

```
    {
      "Key": "costCenter",
      "Value": "120"
    },
    {
      "Key": "project",
      "Value": "lima"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLifecyclePolicy](#)中的。

## AWS Data Pipeline 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Data Pipeline。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### activate-pipeline

以下代码示例演示了如何使用 activate-pipeline。

AWS CLI

激活管道

此示例激活指定的管道：

```
aws datapipeline activate-pipeline --pipeline-id df-00627471S0VYZEXAMPLE
```

要在特定日期和时间激活管道，请使用以下命令：

```
aws datapipeline activate-pipeline --pipeline-id df-00627471S0VYZEXAMPLE --start-timestamp 2015-04-07T00:00:00Z
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ActivatePipeline](#)中的。

## add-tags

以下代码示例演示了如何使用 add-tags。

### AWS CLI

#### 向管道添加标签

此示例将指定的标签添加到指定的管道：

```
aws datapipeline add-tags --pipeline-id df-00627471S0VYZEXAMPLE --tags key=environment,value=production key=owner,value=sales
```

要查看标签，请使用 describe-pipelines 命令。例如，示例命令中添加的标签在 describe-pipelines 的输出中如下所示：

```
{
  ...
  "tags": [
    {
      "value": "production",
      "key": "environment"
    },
    {
      "value": "sales",
      "key": "owner"
    }
  ]
  ...
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddTags](#)中的。

## create-pipeline

以下代码示例演示了如何使用 create-pipeline。

### AWS CLI

#### 创建管道

此示例创建了一个管道：

```
aws datapipeline create-pipeline --name my-pipeline --unique-id my-pipeline-token
```

下面是示例输出：

```
{
  "pipelineId": "df-00627471S0VYZEXAMPLE"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePipeline](#)中的。

## deactivate-pipeline

以下代码示例演示了如何使用 deactivate-pipeline。

### AWS CLI

#### 停用管道

此示例停用了指定的管道：

```
aws datapipeline deactivate-pipeline --pipeline-id df-00627471S0VYZEXAMPLE
```

要仅在所有正在运行的活动完成后才停用管道，请使用以下命令：

```
aws datapipeline deactivate-pipeline --pipeline-id df-00627471S0VYZEXAMPLE --no-cancel-active
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeactivatePipeline](#)中的。



## delete-pipeline

以下代码示例演示了如何使用 delete-pipeline。

### AWS CLI

#### 删除管道

此示例删除了指定的管道：

```
aws datapipeline delete-pipeline --pipeline-id df-00627471S0VYZEXAMPLE
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePipeline](#)中的。

## describe-pipelines

以下代码示例演示了如何使用 describe-pipelines。

### AWS CLI

#### 描述您的管道

此示例描述了指定的管道：

```
aws datapipeline describe-pipelines --pipeline-ids df-00627471S0VYZEXAMPLE
```

下面是示例输出：

```
{
  "pipelineDescriptionList": [
    {
      "fields": [
        {
          "stringValue": "PENDING",
          "key": "@pipelineState"
        },
        {
          "stringValue": "my-pipeline",
          "key": "name"
        },
        {
          "stringValue": "2015-04-07T16:05:58",
```

```

        "key": "@creationTime"
      },
      {
        "stringValue": "df-00627471S0VYZEXAMPLE",
        "key": "@id"
      },
      {
        "stringValue": "123456789012",
        "key": "pipelineCreator"
      },
      {
        "stringValue": "PIPELINE",
        "key": "@sphere"
      },
      {
        "stringValue": "123456789012",
        "key": "@userId"
      },
      {
        "stringValue": "123456789012",
        "key": "@accountId"
      },
      {
        "stringValue": "my-pipeline-token",
        "key": "uniqueId"
      }
    ],
    "pipelineId": "df-00627471S0VYZEXAMPLE",
    "name": "my-pipeline",
    "tags": []
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePipelines](#)中的。

## get-pipeline-definition

以下代码示例演示了如何使用 get-pipeline-definition。

### AWS CLI

#### 获取管道定义

此示例获取指定管道的管道定义：

```
aws datapipeline get-pipeline-definition --pipeline-id df-00627471S0VYZEXAMPLE
```

下面是示例输出：

```
{
  "parameters": [
    {
      "type": "AWS::S3::ObjectKey",
      "id": "myS3OutputLoc",
      "description": "S3 output folder"
    },
    {
      "default": "s3://us-east-1.elasticmapreduce.samples/pig-apache-logs/data",
      "type": "AWS::S3::ObjectKey",
      "id": "myS3InputLoc",
      "description": "S3 input folder"
    },
    {
      "default": "grep -rc \"GET\" ${INPUT1_STAGING_DIR}/* >
${OUTPUT1_STAGING_DIR}/output.txt",
      "type": "String",
      "id": "myShellCmd",
      "description": "Shell command to run"
    }
  ],
  "objects": [
    {
      "type": "Ec2Resource",
      "terminateAfter": "20 Minutes",
      "instanceType": "t1.micro",
      "id": "EC2ResourceObj",
      "name": "EC2ResourceObj"
    },
    {
      "name": "Default",
      "failureAndRerunMode": "CASCADE",
      "resourceRole": "DataPipelineDefaultResourceRole",
      "schedule": {
        "ref": "DefaultSchedule"
      },
      "role": "DataPipelineDefaultRole",
    }
  ]
}
```

```

        "scheduleType": "cron",
        "id": "Default"
    },
    {
        "directoryPath": "#{myS3OutputLoc}/#{format(@scheduledStartTime, 'YYYY-MM-
dd-HH-mm-ss')}",
        "type": "S3DataNode",
        "id": "S3OutputLocation",
        "name": "S3OutputLocation"
    },
    {
        "directoryPath": "#{myS3InputLoc}",
        "type": "S3DataNode",
        "id": "S3InputLocation",
        "name": "S3InputLocation"
    },
    {
        "startAt": "FIRST_ACTIVATION_DATE_TIME",
        "name": "Every 15 minutes",
        "period": "15 minutes",
        "occurrences": "4",
        "type": "Schedule",
        "id": "DefaultSchedule"
    },
    {
        "name": "ShellCommandActivityObj",
        "command": "#{myShellCmd}",
        "output": {
            "ref": "S3OutputLocation"
        },
        "input": {
            "ref": "S3InputLocation"
        },
        "stage": "true",
        "type": "ShellCommandActivity",
        "id": "ShellCommandActivityObj",
        "runsOn": {
            "ref": "EC2ResourceObj"
        }
    }
],
"values": {
    "myS3OutputLoc": "s3://my-s3-bucket/",

```

```
    "myS3InputLoc": "s3://us-east-1.elasticmapreduce.samples/pig-apache-logs/
data",
    "myShellCmd": "grep -rc \"GET\" ${INPUT1_STAGING_DIR}/* >
${OUTPUT1_STAGING_DIR}/output.txt"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPipelineDefinition](#)中的。

## list-pipelines

以下代码示例演示了如何使用 list-pipelines。

### AWS CLI

列出您的管道

此示例列出了您的管道：

```
aws datapipeline list-pipelines
```

下面是示例输出：

```
{
  "pipelineIdList": [
    {
      "id": "df-00627471S0VYZEXAMPLE",
      "name": "my-pipeline"
    },
    {
      "id": "df-09028963KNVMREXAMPLE",
      "name": "ImportDDB"
    },
    {
      "id": "df-0870198233ZYVEXAMPLE",
      "name": "CrossRegionDDB"
    },
    {
      "id": "df-00189603TB4MZEXAMPLE",
      "name": "CopyRedshift"
    }
  ]
}
```

```
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPipelines](#)中的。

## list-runs

以下代码示例演示了如何使用 `list-runs`。

### AWS CLI

示例 1：列出您的管道运行情况

以下`list-runs`示例列出了指定管道的运行情况。

```
aws datapipeline list-runs --pipeline-id df-00627471S0VYZEXAMPLE
```

输出：

Name	Scheduled Start	Status	Ended	ID
	Started			
1. EC2ResourceObj	2015-04-12T17:33:02	CREATING		
@EC2ResourceObj_2015-04-12T17:33:02	2015-04-12T17:33:10			
2. S3InputLocation	2015-04-12T17:33:02	FINISHED		
@S3InputLocation_2015-04-12T17:33:02	2015-04-12T17:33:09			
3. S3OutputLocation	2015-04-12T17:33:02	WAITING_ON_DEPENDENCIES		
@S3OutputLocation_2015-04-12T17:33:02	2015-04-12T17:33:09			
4. ShellCommandActivityObj	2015-04-12T17:33:02	WAITING_FOR_RUNNER		
@ShellCommandActivityObj_2015-04-12T17:33:02	2015-04-12T17:33:09			

示例 2：列出指定日期之间的管道运行

以下`list-runs`示例使用`--start-interval`来指定要包含在输出中的日期。

```
aws datapipeline list-runs --pipeline-id df-01434553B58A2SHZUK05 --start-interval
2017-10-07T00:00:00,2017-10-08T00:00:00
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRuns](#)中的。

## put-pipeline-definition

以下代码示例演示了如何使用 `put-pipeline-definition`。

### AWS CLI

上传管道定义

此示例将指定的管道定义上传到指定的管道：

```
aws datapipeline put-pipeline-definition --pipeline-id df-00627471S0VYZEXAMPLE --
pipeline-definition file://my-pipeline-definition.json
```

下面是示例输出：

```
{
  "validationErrors": [],
  "errored": false,
  "validationWarnings": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutPipelineDefinition](#)中的。

## remove-tags

以下代码示例演示了如何使用 `remove-tags`。

### AWS CLI

从管道中移除标签

此示例从指定的管道中删除指定的标签：

```
aws datapipeline remove-tags --pipeline-id df-00627471S0VYZEXAMPLE --tag-keys
environment
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveTags](#)中的。

## DataSync 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 DataSync。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **update-location-azure-blob**

以下代码示例演示了如何使用 update-location-azure-blob。

AWS CLI

使用新代理更新您的转账地点

以下update-location-object-storage示例使用新代理更新你在 Microsoft Azure Blob 存储中的 DataSync 位置。

```
aws datasync update-location-azure-blob \  
  --location-arn arn:aws:datasync:us-west-2:123456789012:location/loc-  
abcdef01234567890 \  
  --agent-arns arn:aws:datasync:us-west-2:123456789012:agent/  
agent-1234567890abcdef0 \  
  --sas-configuration '{ \  
    "Token": "sas-token-for-azure-blob-storage-access" \  
  }'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS DataSync 用户指南》中的“[更换代理](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLocationAzureBlob](#)中的。

### **update-location-hdfs**

以下代码示例演示了如何使用 update-location-hdfs。



## AWS CLI

### 使用新代理更新您的转账地点

以下update-location-hdfs示例使用新代理更新您 DataSync 的 HDFS 位置。只有当您的 HDFS 集群使用 Kerberos 身份验证时，才需要--kerberos-keytab和--kerberos-krb5-conf选项。

```
aws datasync update-location-hdfs \  
  --location-arn arn:aws:datasync:us-west-2:123456789012:location/loc-  
abcdef01234567890 \  
  --agent-arns arn:aws:datasync:us-west-2:123456789012:agent/  
agent-1234567890abcdef0 \  
  --kerberos-keytab file://hdfs.keytab  
  --kerberos-krb5-conf file://krb5.conf
```

hdfs.keytab 的内容：

```
N/A. The content of this file is encrypted and not human readable.
```

krb5.conf 的内容：

```
[libdefaults]  
  default_realm = EXAMPLE.COM  
  dns_lookup_realm = false  
  dns_lookup_kdc = false  
  rdns = true  
  ticket_lifetime = 24h  
  forwardable = true  
  udp_preference_limit = 1000000  
  default_tkt_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 des3-cbc-  
sha1  
  default_tgs_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 des3-cbc-  
sha1  
  permitted_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 des3-cbc-  
sha1  
  
[realms]  
  EXAMPLE.COM = {  
    kdc = kdc1.example.com  
    admin_server = krbadmin.example.com  
    default_domain = example.com
```

```
}

[domain_realm]
    .example.com = EXAMPLE.COM
    example.com = EXAMPLE.COM

[logging]
    kdc = FILE:/var/log/krb5kdc.log
    admin_server = FILE:/var/log/kerberos/kadmin.log
    default = FILE:/var/log/krb5libs.log
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS DataSync 用户指南》中的“[更换代理](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLocationHdfs](#)中的。

## update-location-nfs

以下代码示例演示了如何使用 update-location-nfs。

### AWS CLI

使用新代理更新您的转账地点

以下update-location-nfs示例使用新代理更新您 DataSync 的 NFS 位置。

```
aws datasync update-location-nfs \
    --location-arn arn:aws:datasync:us-west-2:123456789012:location/loc-
    abcdef01234567890 \
    --on-prem-config AgentArns=arn:aws:datasync:us-west-2:123456789012:agent/
    agent-1234567890abcdef0
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS DataSync 用户指南》中的“[更换代理](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLocationNfs](#)中的。

## update-location-object-storage

以下代码示例演示了如何使用 update-location-object-storage。

## AWS CLI

使用新代理更新您的转账地点

以下update-location-object-storage示例使用新代理更新您的 DataSync 对象存储位置。

```
aws datasync update-location-object-storage \  
  --location-arn arn:aws:datasync:us-west-2:123456789012:location/loc-  
abcdef01234567890 \  
  --agent-arns arn:aws:datasync:us-west-2:123456789012:agent/  
agent-1234567890abcdef0 \  
  --secret-key secret-key-for-object-storage
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS DataSync 用户指南》中的“[更换代理](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLocationObjectStorage](#)中的。

## update-location-smb

以下代码示例演示了如何使用 update-location-smb。

## AWS CLI

使用新代理更新您的转账地点

以下update-location-smb示例使用新代理更新您的 DataSync SMB 位置。

```
aws datasync update-location-smb \  
  --location-arn arn:aws:datasync:us-west-2:123456789012:location/loc-  
abcdef01234567890 \  
  --agent-arns arn:aws:datasync:us-west-2:123456789012:agent/  
agent-1234567890abcdef0 \  
  --password smb-file-server-password
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS DataSync 用户指南》中的“[更换代理](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLocationSmb](#)中的。

## 使用的 DAX 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 DAX 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### create-cluster

以下代码示例演示了如何使用 create-cluster。

AWS CLI

#### 创建 DAX 集群

以下create-cluster示例使用指定设置创建一个 DAX 集群。

```
aws dax create-cluster \  
  --cluster-name daxcluster \  
  --node-type dax.r4.large \  
  --replication-factor 3 \  
  --iam-role-arn roleARN \  
  --sse-specification Enabled=true
```

输出：

```
{  
  "Cluster": {  
    "ClusterName": "daxcluster",  
    "ClusterArn": "arn:aws:dax:us-west-2:123456789012:cache/daxcluster",  
    "TotalNodes": 3,
```

```

    "ActiveNodes": 0,
    "NodeType": "dax.r4.large",
    "Status": "creating",
    "ClusterDiscoveryEndpoint": {
      "Port": 8111
    },
    "PreferredMaintenanceWindow": "thu:13:00-thu:14:00",
    "SubnetGroup": "default",
    "SecurityGroups": [
      {
        "SecurityGroupIdentifier": "sg-1af6e36e",
        "Status": "active"
      }
    ],
    "IamRoleArn": "arn:aws:iam::123456789012:role/DAXServiceRoleForDynamoDBAccess",
    "ParameterGroup": {
      "ParameterGroupName": "default.dax1.0",
      "ParameterApplyStatus": "in-sync",
      "NodeIdsToReboot": []
    },
    "SSEDescription": {
      "Status": "ENABLED"
    }
  }
}

```

有关更多信息，请参阅 Amazon DynamoDB 开发者指南中的步骤 3：创建 [DAX 集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateCluster](#) 中的。

## create-parameter-group

以下代码示例演示了如何使用 create-parameter-group。

### AWS CLI

#### 创建参数组

以下 `create-parameter-group` 示例使用指定设置创建参数组。

```

aws dax create-parameter-group \
  --parameter-group-name daxparametergroup \

```

```
--description "A new parameter group"
```

输出：

```
{
  "ParameterGroup": {
    "ParameterGroupName": "daxparametergroup",
    "Description": "A new parameter group"
  }
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 DA [X 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateParameterGroup](#)中的。

## create-subnet-group

以下代码示例演示了如何使用 create-subnet-group。

### AWS CLI

#### 创建 DAX 子网组

以下create-subnet-group示例使用指定设置创建子网组。

```
aws dax create-subnet-group \
  --subnet-group-name daxSubnetGroup \
  --subnet-ids subnet-11111111 subnet-22222222
```

输出：

```
{
  "SubnetGroup": {
    "SubnetGroupName": "daxSubnetGroup",
    "VpcId": "vpc-05a1fa8e00c325226",
    "Subnets": [
      {
        "SubnetIdentifier": "subnet-11111111",
        "SubnetAvailabilityZone": "us-west-2b"
      },
      {
        "SubnetIdentifier": "subnet-22222222",
        "SubnetAvailabilityZone": "us-west-2c"
      }
    ]
  }
}
```

```
    }  
  ]  
}  
}
```

有关更多信息，请参阅 Amazon DynamoDB 开发者指南中的[步骤 2：创建子网组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateSubnetGroup](#) 中的。

## decrease-replication-factor

以下代码示例演示了如何使用 decrease-replication-factor。

### AWS CLI

从集群中移除一个或多个节点

以下 decrease-replication-factor 示例将指定 DAX 集群中的节点数减少到一个。

```
aws dax decrease-replication-factor \  
  --cluster-name daxcluster \  
  --new-replication-factor 1
```

输出：

```
{  
  "Cluster": {  
    "ClusterName": "daxcluster",  
    "ClusterArn": "arn:aws:dax:us-west-2:123456789012:cache/daxcluster",  
    "TotalNodes": 3,  
    "ActiveNodes": 3,  
    "NodeType": "dax.r4.large",  
    "Status": "modifying",  
    "ClusterDiscoveryEndpoint": {  
      "Address": "daxcluster.ey3o9d.clustercfg.dax.usw2.cache.amazonaws.com",  
      "Port": 8111  
    },  
    "Nodes": [  
      {  
        "NodeId": "daxcluster-a",  
        "Endpoint": {  
          "Address": "daxcluster-  
a.ey3o9d.0001.dax.usw2.cache.amazonaws.com",
```

```
        "Port": 8111
      },
      "NodeCreateTime": 1576625059.509,
      "AvailabilityZone": "us-west-2c",
      "NodeStatus": "available",
      "ParameterGroupStatus": "in-sync"
    },
    {
      "NodeId": "daxcluster-b",
      "Endpoint": {
        "Address": "daxcluster-
b.ey3o9d.0001.dax.usw2.cache.amazonaws.com",
        "Port": 8111
      },
      "NodeCreateTime": 1576625059.509,
      "AvailabilityZone": "us-west-2a",
      "NodeStatus": "available",
      "ParameterGroupStatus": "in-sync"
    },
    {
      "NodeId": "daxcluster-c",
      "Endpoint": {
        "Address": "daxcluster-
c.ey3o9d.0001.dax.usw2.cache.amazonaws.com",
        "Port": 8111
      },
      "NodeCreateTime": 1576625059.509,
      "AvailabilityZone": "us-west-2b",
      "NodeStatus": "available",
      "ParameterGroupStatus": "in-sync"
    }
  ],
  "PreferredMaintenanceWindow": "thu:13:00-thu:14:00",
  "SubnetGroup": "default",
  "SecurityGroups": [
    {
      "SecurityGroupIdentifier": "sg-1af6e36e",
      "Status": "active"
    }
  ],
  "IamRoleArn": "arn:aws:iam::123456789012:role/
DAXServiceRoleForDynamoDBAccess",
  "ParameterGroup": {
    "ParameterGroupName": "default.dax1.0",
```



```

        "ParameterApplyStatus": "in-sync",
        "NodeIdsToReboot": []
    },
    "SSEDescription": {
        "Status": "ENABLED"
    }
}
}

```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 DAX [集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DecreaseReplicationFactor](#) 中的。

## delete-cluster

以下代码示例演示了如何使用 delete-cluster。

### AWS CLI

#### 删除 DAX 集群

以下 delete-cluster 示例删除指定的 DAX 集群。

```

aws dax delete-cluster \
  --cluster-name daxcluster

```

输出：

```

{
  "Cluster": {
    "ClusterName": "daxcluster",
    "ClusterArn": "arn:aws:dax:us-west-2:123456789012:cache/daxcluster",
    "TotalNodes": 3,
    "ActiveNodes": 0,
    "NodeType": "dax.r4.large",
    "Status": "deleting",
    "ClusterDiscoveryEndpoint": {
      "Address": "dd.ey3o9d.clustercfg.dax.usw2.cache.amazonaws.com",
      "Port": 8111
    },
    "PreferredMaintenanceWindow": "fri:06:00-fri:07:00",
    "SubnetGroup": "default",
    "SecurityGroups": [

```

```
    {
      "SecurityGroupIdentifier": "sg-1af6e36e",
      "Status": "active"
    }
  ],
  "IamRoleArn": "arn:aws:iam::123456789012:role/
DAXServiceRoleForDynamoDBAccess",
  "ParameterGroup": {
    "ParameterGroupName": "default.dax1.0",
    "ParameterApplyStatus": "in-sync",
    "NodeIdsToReboot": []
  },
  "SSEDescription": {
    "Status": "ENABLED"
  }
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 DA [X 集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteCluster](#) 中的。

## delete-parameter-group

以下代码示例演示了如何使用 delete-parameter-group。

### AWS CLI

#### 删除参数组

以下 delete-parameter-group 示例删除指定的 DAX 参数组。

```
aws dax delete-parameter-group \
  --parameter-group-name daxparametergroup
```

输出：

```
{
  "DeletionMessage": "Parameter group daxparametergroup has been deleted."
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 DA [X 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteParameterGroup](#)中的。

## delete-subnet-group

以下代码示例演示了如何使用 delete-subnet-group。

### AWS CLI

删除子网组

以下delete-subnet-group示例删除指定的 DAX 子网组。

```
aws dax delete-subnet-group \  
  --subnet-group-name daxSubnetGroup
```

输出：

```
{  
  "DeletionMessage": "Subnet group daxSubnetGroup has been deleted."  
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 DA [X 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSubnetGroup](#)中的。

## describe-clusters

以下代码示例演示了如何使用 describe-clusters。

### AWS CLI

返回有关所有已预配置的 DAX 集群的信息

以下describe-clusters示例显示有关所有已配置的 DAX 集群的详细信息。

```
aws dax describe-clusters
```

输出：

```
{  
  "Clusters": [  
    {
```

```
    "ClusterName": "daxcluster",
    "ClusterArn": "arn:aws:dax:us-west-2:123456789012:cache/daxcluster",
    "TotalNodes": 1,
    "ActiveNodes": 1,
    "NodeType": "dax.r4.large",
    "Status": "available",
    "ClusterDiscoveryEndpoint": {
      "Address":
"daxcluster.ey3o9d.clustercfg.dax.usw2.cache.amazonaws.com",
      "Port": 8111
    },
    "Nodes": [
      {
        "NodeId": "daxcluster-a",
        "Endpoint": {
          "Address": "daxcluster-
a.ey3o9d.0001.dax.usw2.cache.amazonaws.com",
          "Port": 8111
        },
        "NodeCreateTime": 1576625059.509,
        "AvailabilityZone": "us-west-2c",
        "NodeStatus": "available",
        "ParameterGroupStatus": "in-sync"
      }
    ],
    "PreferredMaintenanceWindow": "thu:13:00-thu:14:00",
    "SubnetGroup": "default",
    "SecurityGroups": [
      {
        "SecurityGroupIdentifier": "sg-1af6e36e",
        "Status": "active"
      }
    ],
    "IamRoleArn": "arn:aws:iam::123456789012:role/
DAXServiceRoleForDynamoDBAccess",
    "ParameterGroup": {
      "ParameterGroupName": "default.dax1.0",
      "ParameterApplyStatus": "in-sync",
      "NodeIdsToReboot": []
    },
    "SSEDescription": {
      "Status": "ENABLED"
    }
  }
}
```

```
]
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 [DAX 集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeClusters](#) 中的。

## describe-default-parameters

以下代码示例演示了如何使用 describe-default-parameters。

### AWS CLI

返回 DAX 的默认系统参数信息

以下 describe-default-parameters 示例显示了 DAX 的默认系统参数信息。

```
aws dax describe-default-parameters
```

输出：

```
{
  "Parameters": [
    {
      "ParameterName": "query-ttl-millis",
      "ParameterType": "DEFAULT",
      "ParameterValue": "300000",
      "NodeTypeSpecificValues": [],
      "Description": "Duration in milliseconds for queries to remain cached",
      "Source": "user",
      "DataType": "integer",
      "AllowedValues": "0-",
      "IsModifiable": "TRUE",
      "ChangeType": "IMMEDIATE"
    },
    {
      "ParameterName": "record-ttl-millis",
      "ParameterType": "DEFAULT",
      "ParameterValue": "300000",
      "NodeTypeSpecificValues": [],
      "Description": "Duration in milliseconds for records to remain valid in
cache (Default: 0 = infinite)",
      "Source": "user",
```

```
        "DataType": "integer",
        "AllowedValues": "0-",
        "IsModifiable": "TRUE",
        "ChangeType": "IMMEDIATE"
    }
]
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 DAX [集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDefaultParameters](#) 中的。

## describe-events

以下代码示例演示了如何使用 describe-events。

### AWS CLI

返回与 DAX 集群和参数组相关的所有事件

以下 describe-events 示例显示了与 DAX 集群和参数组相关的事件的详细信息。

```
aws dax describe-events
```

输出：

```
{
  "Events": [
    {
      "SourceName": "daxcluster",
      "SourceType": "CLUSTER",
      "Message": "Cluster deleted.",
      "Date": 1576702736.706
    },
    {
      "SourceName": "daxcluster",
      "SourceType": "CLUSTER",
      "Message": "Removed node daxcluster-b.",
      "Date": 1576702691.738
    },
    {
      "SourceName": "daxcluster",
      "SourceType": "CLUSTER",
```

```

    "Message": "Removed node daxcluster-a.",
    "Date": 1576702633.498
  },
  {
    "SourceName": "daxcluster",
    "SourceType": "CLUSTER",
    "Message": "Removed node daxcluster-c.",
    "Date": 1576702631.329
  },
  {
    "SourceName": "daxcluster",
    "SourceType": "CLUSTER",
    "Message": "Cluster created.",
    "Date": 1576626560.057
  }
]
}

```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 [DAX 集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEvents](#) 中的。

## describe-parameter-groups

以下代码示例演示了如何使用 describe-parameter-groups。

### AWS CLI

描述 DAX 中定义的参数组

以下 describe-parameter-groups 示例检索有关在 DAX 中定义的参数组的详细信息。

```
aws dax describe-parameter-groups
```

输出：

```

{
  "ParameterGroups": [
    {
      "ParameterGroupName": "default.dax1.0",
      "Description": "Default parameter group for dax1.0"
    }
  ]
}

```

```
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 DAX [集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeParameterGroups](#)中的。

## describe-parameters

以下代码示例演示了如何使用 describe-parameters。

### AWS CLI

描述在 DAX 参数组中定义的参数

以下describe-parameters示例检索有关在指定 DAX 参数组中定义的参数的详细信息。

```
aws dax describe-parameters \  
  --parameter-group-name default.dax1.0
```

输出：

```
{  
  "Parameters": [  
    {  
      "ParameterName": "query-ttl-millis",  
      "ParameterType": "DEFAULT",  
      "ParameterValue": "300000",  
      "NodeTypeSpecificValues": [],  
      "Description": "Duration in milliseconds for queries to remain cached",  
      "Source": "user",  
      "DataType": "integer",  
      "AllowedValues": "0-",  
      "IsModifiable": "TRUE",  
      "ChangeType": "IMMEDIATE"  
    },  
    {  
      "ParameterName": "record-ttl-millis",  
      "ParameterType": "DEFAULT",  
      "ParameterValue": "300000",  
      "NodeTypeSpecificValues": [],  
      "Description": "Duration in milliseconds for records to remain valid in  
cache (Default: 0 = infinite)",  
      "Source": "user",
```



```
        "DataType": "integer",
        "AllowedValues": "0-",
        "IsModifiable": "TRUE",
        "ChangeType": "IMMEDIATE"
    }
]
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 [DAX 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeParameters](#)中的。

## describe-subnet-groups

以下代码示例演示了如何使用 describe-subnet-groups。

### AWS CLI

描述 DAX 中定义的子网组

以下describe-subnet-groups示例检索 DAX 中定义的子网组的详细信息。

```
aws dax describe-subnet-groups
```

输出：

```
{
  "SubnetGroups": [
    {
      "SubnetGroupName": "default",
      "Description": "Default CacheSubnetGroup",
      "VpcId": "vpc-ee70a196",
      "Subnets": [
        {
          "SubnetIdentifier": "subnet-874953af",
          "SubnetAvailabilityZone": "us-west-2d"
        },
        {
          "SubnetIdentifier": "subnet-bd3d1fc4",
          "SubnetAvailabilityZone": "us-west-2a"
        },
        {
          "SubnetIdentifier": "subnet-72c2ff28",
```

```

        "SubnetAvailabilityZone": "us-west-2c"
      },
      {
        "SubnetIdentifier": "subnet-09e6aa42",
        "SubnetAvailabilityZone": "us-west-2b"
      }
    ]
  }
]
}

```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 [DAX 集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeSubnetGroups](#) 中的。

## increase-replication-factor

以下代码示例演示了如何使用 `increase-replication-factor`。

### AWS CLI

提高 DAX 集群的重复因子

以下 `increase-replication-factor` 示例将指定 DAX 集群的重复因子增加到 3。

```

aws dax increase-replication-factor \
  --cluster-name daxcluster \
  --new-replication-factor 3

```

输出：

```

{
  "Cluster": {
    "ClusterName": "daxcluster",
    "ClusterArn": "arn:aws:dax:us-west-2:123456789012:cache/daxcluster",
    "TotalNodes": 3,
    "ActiveNodes": 1,
    "NodeType": "dax.r4.large",
    "Status": "modifying",
    "ClusterDiscoveryEndpoint": {
      "Address": "daxcluster.ey3o9d.clustercfg.dax.usw2.cache.amazonaws.com",
      "Port": 8111
    }
  },

```

```
    "Nodes": [
      {
        "NodeId": "daxcluster-a",
        "Endpoint": {
          "Address": "daxcluster-
a.ey3o9d.0001.dax.usw2.cache.amazonaws.com",
          "Port": 8111
        },
        "NodeCreateTime": 1576625059.509,
        "AvailabilityZone": "us-west-2c",
        "NodeStatus": "available",
        "ParameterGroupStatus": "in-sync"
      },
      {
        "NodeId": "daxcluster-b",
        "NodeStatus": "creating"
      },
      {
        "NodeId": "daxcluster-c",
        "NodeStatus": "creating"
      }
    ],
    "PreferredMaintenanceWindow": "thu:13:00-thu:14:00",
    "SubnetGroup": "default",
    "SecurityGroups": [
      {
        "SecurityGroupIdentifier": "sg-1af6e36e",
        "Status": "active"
      }
    ],
    "IamRoleArn": "arn:aws:iam::123456789012:role/
DAXServiceRoleForDynamoDBAccess",
    "ParameterGroup": {
      "ParameterGroupName": "default.dax1.0",
      "ParameterApplyStatus": "in-sync",
      "NodeIdsToReboot": []
    },
    "SSEDescription": {
      "Status": "ENABLED"
    }
  }
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 [DA X 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[IncreaseReplicationFactor](#)中的。

## list-tags

以下代码示例演示了如何使用 list-tags。

### AWS CLI

在 DAX 资源上列出标签

以下list-tags示例列出了附加到指定 DAX 集群的标签密钥和值。

```
aws dax list-tags \  
  --resource-name arn:aws:dax:us-west-2:123456789012:cache/daxcluster
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "ClusterUsage",  
      "Value": "prod"  
    }  
  ]  
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 [DA X 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTags](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为 DAX 资源添加标签

以下tag-resource示例将指定的标签键名称和关联值附加到指定的 DAX 集群，以描述集群使用情况。

```
aws dax tag-resource \  
  --resource-name arn:aws:dax:us-west-2:123456789012:cache/daxcluster \  
  --tags="Key=ClusterUsage,Value=prod"
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "ClusterUsage",  
      "Value": "prod"  
    }  
  ]  
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 DA [X 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从 DAX 资源中移除标签

以下 untag-resource 示例从 DAX 集群中移除具有指定密钥名称的标签。

```
aws dax untag-resource \  
  --resource-name arn:aws:dax:us-west-2:123456789012:cache/daxcluster \  
  --tag-keys="ClusterUsage"
```

输出：

```
{  
  "Tags": []  
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的管理 DA [X 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## 使用 Detective 示例 AWS CLI

以下代码示例向您展示了如何使用 with Detective 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **accept-invitation**

以下代码示例演示了如何使用 accept-invitation。

AWS CLI

在行为图中接受成为成员账户的邀请

以下accept-invitation示例接受了行为图中成为会员账户的邀请 arn:aws:detective:us-east-1:111122223333:graph:123412341234。

```
aws detective accept-invitation \  
  --graph-arn arn:aws:detective:us-east-1:111122223333:graph:123412341234
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Detective 管理指南》中的[回复行为图邀请](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AcceptInvitation](#)中的。

### **create-graph**

以下代码示例演示了如何使用 create-graph。

## AWS CLI

### 启用 Amazon Detective 并创建新的行为图

以下 `create-graph` 示例为在运行命令的区域中运行命令的 AWS 账户启用 Detective。创建了一个以该帐户作为其管理员帐户的新行为图。该命令还将值“财务”分配给“部门”标签。

```
aws detective create-graph \  
  --tags '{"Department": "Finance"}
```

输出：

```
{  
  "GraphArn": "arn:aws:detective:us-  
east-1:111122223333:graph:027c7c4610ea4aacaf0b883093cab899"  
}
```

有关更多信息，请参阅 [《亚马逊侦探管理指南》中的启用 Amazon Detective](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateGraph](#) 中的。

## create-members

以下代码示例演示了如何使用 `create-members`。

## AWS CLI

### 邀请成员账号访问行为图

以下 `create-members` 示例邀请两个 AWS 账户成为行为图中的成员账户 `arn:aws:detective:us-east-1:111122223333:graph:123412341234`。对于每个账户，请求都会提供 AWS 账户 ID 和账户 root 用户电子邮件地址。该请求包括要插入邀请电子邮件的自定义消息。

```
aws detective create-members \  
  --accounts AccountId=444455556666,EmailAddress=mmajor@example.com  
  AccountId=123456789012,EmailAddress=jstiles@example.com \  
  --graph-arn arn:aws:detective:us-east-1:111122223333:graph:123412341234 \  
  --message "This is Paul Santos. I need to add your account to the data we use  
for security investigation in Amazon Detective. If you have any questions, contact  
me at psantos@example.com."
```

输出：

```
{
  "Members": [
    {
      "AccountId": "444455556666",
      "AdministratorId": "111122223333",
      "EmailAddress": "mmajor@example.com",
      "GraphArn": "arn:aws:detective:us-east-1:111122223333:graph:123412341234",
      "InvitedTime": 1579826107000,
      "MasterId": "111122223333",
      "Status": "INVITED",
      "UpdatedTime": 1579826107000
    },
    {
      "AccountId": "123456789012",
      "AdministratorId": "111122223333",
      "EmailAddress": "jstiles@example.com",
      "GraphArn": "arn:aws:detective:us-east-1:111122223333:graph:123412341234",
      "InvitedTime": 1579826107000,
      "MasterId": "111122223333",
      "Status": "VERIFICATION_IN_PROGRESS",
      "UpdatedTime": 1579826107000
    }
  ],
  "UnprocessedAccounts": [ ]
}
```

有关更多信息，请参阅《亚马逊侦探graph-admin-add-member管理指南》中的“邀请成员账户访问行为图 < <https://docs.aws.amazon.com/detective/latest/adminguide/-accounts.html>>”。

在不发送邀请电子邮件的情况下邀请成员账户

以下create-members示例邀请两个 AWS 账户成为行为图中的成员账户 arn:aws:detective:us-east-1:111122223333:graph:123412341234。对于每个账户，请求都会提供 AWS 账户 ID 和账户 root 用户电子邮件地址。成员账户不会收到邀请电子邮件。

```
aws detective create-members \
  --accounts AccountId=444455556666,EmailAddress=mmajor@example.com
  AccountId=123456789012,EmailAddress=jstiles@example.com \
  --graph-arn arn:aws:detective:us-east-1:111122223333:graph:123412341234 \
  --disable-email-notification
```

输出：



```
{
  "Members": [
    {
      "AccountId": "444455556666",
      "AdministratorId": "111122223333",
      "EmailAddress": "mmajor@example.com",
      "GraphArn": "arn:aws:detective:us-east-1:111122223333:graph:123412341234",
      "InvitedTime": 1579826107000,
      "MasterId": "111122223333",
      "Status": "INVITED",
      "UpdatedTime": 1579826107000
    },
    {
      "AccountId": "123456789012",
      "AdministratorId": "111122223333",
      "EmailAddress": "jstiles@example.com",
      "GraphArn": "arn:aws:detective:us-east-1:111122223333:graph:123412341234",
      "InvitedTime": 1579826107000,
      "MasterId": "111122223333",
      "Status": "VERIFICATION_IN_PROGRESS",
      "UpdatedTime": 1579826107000
    }
  ],
  "UnprocessedAccounts": [ ]
}
```

有关更多信息，请参阅《亚马逊侦探graph-admin-add-member管理指南》中的“邀请成员账户访问行为图 < <https://docs.aws.amazon.com/detective/latest/adminguide/-accounts.html> >”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateMembers](#)中的。

## delete-graph

以下代码示例演示了如何使用 delete-graph。

### AWS CLI

禁用 Detective 并删除行为图

以下delete-graph示例禁用 Detective 并删除指定的行为图。

```
aws detective delete-graph \
```

```
--graph-arn arn:aws:detective:us-east-1:111122223333:graph:123412341234
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊侦探管理指南](#)》中的[禁用 Amazon Detective](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteGraph](#)中的。

## delete-members

以下代码示例演示了如何使用 delete-members。

### AWS CLI

从行为图中移除成员账户

以下delete-members示例从行为图中删除了两个成员账户 arn:aws:detective:us-east-1:111122223333:graph:12341234121234。为了识别账户，请求提供了 AWS 账户 ID。

```
aws detective delete-members \  
  --account-ids 444455556666 123456789012 \  
  --graph-arn arn:aws:detective:us-east-1:111122223333:graph:123412341234
```

输出：

```
{  
  "AccountIds": [ "444455556666", "123456789012" ],  
  "UnprocessedAccounts": [ ]  
}
```

有关更多信息，请参阅《[亚马逊侦探graph-admin-remove-member管理指南](#)》中的“从行为图中删除成员账户 < <https://docs.aws.amazon.com/detective/latest/adminguide/-accounts.html>”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMembers](#)中的。

## disassociate-membership

以下代码示例演示了如何使用 disassociate-membership。

### AWS CLI

退出行为图的成员资格

以下取消关联成员资格示例从行为图中删除了运行该命令的 AWS 账户 `arn:aws:detective:us-east-1:111122223333:graph:123412341234`。

```
aws detective disassociate-membership \  
  --graph-arn arn:aws:detective:us-east-1:111122223333:graph:123412341234
```

有关更多信息，请参阅《亚马逊侦探管理指南》`member-remove-self-from`中的“从行为图中删除账户 <<https://docs.aws.amazon.com/detective/latest/adminguide/-graph.html>>”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateMembership](#)中的。

## get-members

以下代码示例演示了如何使用 `get-members`。

### AWS CLI

检索有关所选行为图成员账户的信息

以下`get-members`示例在行为图中检索有关两个成员账户的信息 `arn:aws:detective:us-east-1:111122223333:graph:123412341234`。对于这两个账户，请求会提供 AWS 账户 ID。

```
aws detective get-members \  
  --account-ids 444455556666 123456789012 \  
  --graph-arn arn:aws:detective:us-east-1:111122223333:graph:123412341234
```

输出：

```
{  
  "MemberDetails": [  
    {  
      "AccountId": "444455556666",  
      "AdministratorId": "111122223333",  
      "EmailAddress": "mmajor@example.com",  
      "GraphArn": "arn:aws:detective:us-east-1:111122223333:graph:123412341234",  
      "InvitedTime": 1579826107000,  
      "MasterId": "111122223333",  
      "Status": "INVITED",  
      "UpdatedTime": 1579826107000  
    }  
  ]  
}
```

```
    "AccountId": "123456789012",
    "AdministratorId": "111122223333",
    "EmailAddress": "jstiles@example.com",
    "GraphArn": "arn:aws:detective:us-east-1:111122223333:graph:123412341234",
    "InvitedTime": 1579826107000,
    "MasterId": "111122223333",
    "Status": "INVITED",
    "UpdatedTime": 1579826107000
  }
],
  "UnprocessedAccounts": [ ]
}
```

有关更多信息，请参阅《Amazon Detective 管理指南》中的“在行为图graph-admin-view-accounts中查看账户列表 < <https://docs.aws.amazon.com/detective/latest/adminguide/.html>”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMembers](#)中的。

## list-graphs

以下代码示例演示了如何使用 list-graphs。

### AWS CLI

查看您的账户作为管理员的行为图表列表

以下list-graphs示例检索当前区域内主叫账号作为管理员的行为图。

```
aws detective list-graphs
```

输出：

```
{
  "GraphList": [
    {
      "Arn": "arn:aws:detective:us-east-1:111122223333:graph:123412341234",
      "CreatedTime": 1579736111000
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListGraphs](#)中的。

## list-invitations

以下代码示例演示了如何使用 `list-invitations`。

### AWS CLI

查看账户加入或受邀加入的行为图表列表

以下 `list-invitations` 示例检索主叫账号已被邀请访问的行为图表。结果仅包括未完成和已接受的邀请。它们不包括被拒绝的邀请或已删除的会员资格。

```
aws detective list-invitations
```

输出：

```
{
  "Invitations": [
    {
      "AccountId": "444455556666",
      "AdministratorId": "111122223333",
      "EmailAddress": "mmajor@example.com",
      "GraphArn": "arn:aws:detective:us-east-1:111122223333:graph:123412341234",
      "InvitedTime": 1579826107000,
      "MasterId": "111122223333",
      "Status": "INVITED",
      "UpdatedTime": 1579826107000
    }
  ]
}
```

有关更多信息，请参阅《Amazon Detective 管理指南》中的“查看行为图邀请列表 < <https://docs.aws.amazon.com/detective/latest/adminguide/member-view-graph-invitations.html> >”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListInvitations](#) 中的。

## list-members

以下代码示例演示了如何使用 `list-members`。

### AWS CLI

在行为图中列出成员账户

以下list-members示例检索行为图arn:aws:detective:us-east-1:111122223333:graph:123412341234的已邀请和已启用的成员账户。结果不包括已删除的成员账户。

```
aws detective list-members \  
  --graph-arn arn:aws:detective:us-east-1:111122223333:graph:123412341234
```

输出：

```
{  
  "MemberDetails": [  
    {  
      "AccountId": "444455556666",  
      "AdministratorId": "111122223333",  
      "EmailAddress": "mmajor@example.com",  
      "GraphArn": "arn:aws:detective:us-east-1:111122223333:graph:123412341234",  
      "InvitedTime": 1579826107000,  
      "MasterId": "111122223333",  
      "Status": "INVITED",  
      "UpdateTime": 1579826107000  
    },  
    {  
      "AccountId": "123456789012",  
      "AdministratorId": "111122223333",  
      "EmailAddress": "jstiles@example.com",  
      "GraphArn": "arn:aws:detective:us-east-1:111122223333:graph:123412341234",  
      "InvitedTime": 1579826107000,  
      "MasterId": "111122223333",  
      "PercentOfGraphUtilization": 2,  
      "PercentOfGraphUtilizationUpdateTime": 1586287843,  
      "Status": "ENABLED",  
      "UpdateTime": 1579973711000,  
      "VolumeUsageInBytes": 200,  
      "VolumeUsageUpdateTime": 1586287843  
    }  
  ]  
}
```

有关更多信息，请参阅 [《Amazon Detective 管理指南》](#) 中的在行为图中查看账户列表。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListMembers](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

检索分配给行为图的标签

以下 `list-tags-for-resource` 示例返回分配给指定行为图的标签。

```
aws detective list-tags-for-resource \  
  --resource-arn arn:aws:detective:us-east-1:111122223333:graph:123412341234
```

输出：

```
{  
  "Tags": {  
    "Department" : "Finance"  
  }  
}
```

有关更多信息，请参阅《Amazon Detective 管理指南》中的管理行为 [图的标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## reject-invitation

以下代码示例演示了如何使用 `reject-invitation`。

### AWS CLI

在行为图中拒绝成为成员账户的邀请

以下 `reject-invitation` 示例拒绝了行为图中成为成员账户的邀请 `arn:aws:detective:us-east-1:111122223333:graph:123412341234`。

```
aws detective reject-invitation \  
  --graph-arn arn:aws:detective:us-east-1:111122223333:graph:123412341234
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Detective 管理指南》中的回复行为图邀请 < <https://docs.aws.amazon.com/detective/latest/adminguide/member-invitation-response.html> >。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RejectInvitation](#) 中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源分配标签

以下 tag-resource 示例将 Department 标签的值分配给指定的行为图。

```
aws detective tag-resource \  
  --resource-arn arn:aws:detective:us-east-1:111122223333:graph:123412341234 \  
  --tags '{"Department":"Finance"}
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Detective 管理指南》中的管理行为 [图的标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从资源中移除标签值

以下 untag-resource 示例从指定的行为图中删除“部门”标签。

```
aws detective untag-resource \  
  --resource-arn arn:aws:detective:us-east-1:111122223333:graph:123412341234 \  
  --tag-keys "Department"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Detective 管理指南》中的管理行为 [图的标签](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## 使用 Device Farm 示例 AWS CLI

以下代码示例向您展示了如何使用 with Device Farm 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-device-pool**

以下代码示例演示了如何使用 create-device-pool。

AWS CLI

创建设备池

以下命令为项目创建 Android 设备池：

```
aws devicefarm create-device-pool --name pool1 --rules file://  
device-pool-rules.json --project-arn "arn:aws:devicefarm:us-  
west-2:123456789012:project:070fc3ca-7ec1-4741-9c1f-d3e044efc506"
```

您可以从或的输出中获取项目 ARN。create-project list-projects该文件device-pool-rules.json是当前文件夹中的一个 JSON 文档，用于指定设备平台：

```
[  
  {  
    "attribute": "PLATFORM",
```

```

        "operator": "EQUALS",
        "value": "\"ANDROID\""
    }
]

```

输出：

```

{
  "devicePool": {
    "rules": [
      {
        "operator": "EQUALS",
        "attribute": "PLATFORM",
        "value": "\"ANDROID\""
      }
    ],
    "type": "PRIVATE",
    "name": "pool1",
    "arn": "arn:aws:devicefarm:us-
west-2:123456789012:devicepool:070fc3ca-7ec1-4741-9c1f-
d3e044efc506/2aa8d2a9-5e73-47ca-b929-659cb34b7dcd"
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDevicePool](#)中的。

## create-project

以下代码示例演示了如何使用 create-project。

### AWS CLI

#### 创建项目

以下命令创建一个名为的新项目my-project：

```
aws devicefarm create-project --name my-project
```

输出：

```
{
```

```
"project": {
  "name": "myproject",
  "arn": "arn:aws:devicefarm:us-
west-2:123456789012:project:070fc3ca-7ec1-4741-9c1f-d3e044efc506",
  "created": 1503612890.057
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateProject](#)中的。

## create-upload

以下代码示例演示了如何使用 create-upload。

### AWS CLI

#### 创建上传

以下命令为 Android 应用程序创建上传：

```
aws devicefarm create-upload --project-arn "arn:aws:devicefarm:us-
west-2:123456789012:project:070fc3ca-7ec1-4741-9c1f-d3e044efc506" --name app.apk --
type ANDROID_APP
```

您可以从创建项目或列表项目的输出中获取项目 ARN。

输出：

```
{
  "upload": {
    "status": "INITIALIZED",
    "name": "app.apk",
    "created": 1503614408.769,
    "url": "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
arn%3Aaws%3Adevicefarm%3Aus-west-2%3A123456789012%3Aproject%3A070fc3ca-
c7e1-4471-91cf-d3e4efc50604/uploads/arn%3Aaws%3Adevicefarm%3Aus-
west-2%3A123456789012%3Aupload%3A070fc3ca-7ec1-4741-9c1f-d3e044efc506/dd72723a-
ae9e-4087-09e6-f4cea3599514/app.apk?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20170824T224008Z&X-Amz-SignedHeaders=host&X-Amz-Expires=86400&X-Amz-
Credential=AKIAEXAMPLEPBUMBC3GA%2F20170824%2Fus-west-2%2Fs%2Faws4_request&X-Amz-
Signature=05050370c38894ef5bd09f5d009f36fc8f96fa4bb04e1bba9aca71b8dbe49a0f",
    "type": "ANDROID_APP",
```

```

    "arn": "arn:aws:devicefarm:us-
west-2:123456789012:upload:070fc3ca-7ec1-4741-9c1f-d3e044efc506/dd72723a-
ae9e-4087-09e6-f4cea3599514"
  }
}

```

使用输出中的签名 URL 将文件上传到 Device Farm :

```

curl -T app.apk "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
arn%3Aaws%3Adevicefarm%3Aus-west-2%3A123456789012%3Aproject%3A070fc3ca-
c7e1-4471-91cf-d3e4efc50604/uploads/arn%3Aaws%3Adevicefarm%3Aus-
west-2%3A123456789012%3Aupload%3A070fc3ca-7ec1-4741-9c1f-d3e044efc506/dd72723a-
ae9e-4087-09e6-f4cea3599514/app.apk?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20170824T224008Z&X-Amz-SignedHeaders=host&X-Amz-Expires=86400&X-Amz-
Credential=AKIAEXAMPLEBUMBC3GA%2F20170824%2Fus-west-2%2Fs3%2Faws4_request&X-Amz-
Signature=05050370c38894ef5bd09f5d009f36fc8f96fa4bb04e1bba9aca71b8dbe49a0f"

```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateUpload](#) 中的。

## get-upload

以下代码示例演示了如何使用 get-upload。

### AWS CLI

查看上传

以下命令检索有关上传的信息：

```

aws devicefarm get-upload --arn "arn:aws:devicefarm:us-
west-2:123456789012:upload:070fc3ca-7ec1-4741-9c1f-d3e044efc506/dd72723a-
ae9e-4087-09e6-f4cea3599514"

```

您可以从的输出中获取上传 ARN。create-upload

输出：

```

{
  "upload": {
    "status": "SUCCEEDED",
    "name": "app.apk",
    "created": 1505262773.186,

```

```
    "type": "ANDROID_APP",
    "arn": "arn:aws:devicefarm:us-
west-2:123456789012:upload:070fc3ca-7ec1-4741-9c1f-d3e044efc506/dd72723a-
ae9e-4087-09e6-f4cea3599514",
    "metadata": "{ \"device_admin\":false, \"activity_name\":
\\\"ccom.example.client.LauncherActivity\\\", \"version_name\":\\\"1.0.2.94\\\", \"screens
\\\": [\\\"small\\\", \\\"normal\\\", \\\"large\\\", \\\"xlarge\\\"], \"error_type\":null, \"sdk_version
\\\": \\\"16\\\", \"package_name\":\\\"com.example.client\\\", \"version_code\":\\\"20994\\\",
\\\"native_code\\\": [\\\"armeabi-v7a\\\"], \"target_sdk_version\":\\\"25\\\"}"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetUpload](#)中的。

## list-projects

以下代码示例演示了如何使用 list-projects。

### AWS CLI

列出项目

以下内容检索项目列表：

```
aws devicefarm list-projects
```

输出：

```
{
  "projects": [
    {
      "name": "myproject",
      "arn": "arn:aws:devicefarm:us-
west-2:123456789012:project:070fc3ca-7ec1-4741-9c1f-d3e044efc506",
      "created": 1503612890.057
    },
    {
      "name": "otherproject",
      "arn": "arn:aws:devicefarm:us-
west-2:123456789012:project:a5f5b752-8098-49d1-86bf-5f7682c1c77e",
      "created": 1505257519.337
    }
  ]
}
```

```
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListProjects](#)中的。

## AWS Direct Connect 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Direct Connect。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **accept-direct-connect-gateway-association-proposal**

以下代码示例演示了如何使用 `accept-direct-connect-gateway-association-proposal`。

AWS CLI

接受网关关联提案

以下人员`accept-direct-connect-gateway-association-proposal`接受指定的提案。

```
aws directconnect accept-direct-connect-gateway-association-proposal \
  --direct-connect-gateway-id 11460968-4ac1-4fd3-bdb2-00599EXAMPLE \
  --proposal-id cb7f41cb-8128-43a5-93b1-dcaedEXAMPLE \
  --associated-gateway-owner-account 111122223333

{
  "directConnectGatewayAssociation": {
    "directConnectGatewayId": "11460968-4ac1-4fd3-bdb2-00599EXAMPLE",
    "directConnectGatewayOwnerAccount": "111122223333",
```

```
    "associationState": "associating",
    "associatedGateway": {
      "id": "tgw-02f776b1a7EXAMPLE",
      "type": "transitGateway",
      "ownerAccount": "111122223333",
      "region": "us-east-1"
    },
    "associationId": "6441f8bf-5917-4279-ade1-9708bEXAMPLE",
    "allowedPrefixesToDirectConnectGateway": [
      {
        "cidr": "192.168.1.0/30"
      }
    ]
  }
}
```

有关更多信息，请参阅 [Dire AWS ct Connect 用户指南中的接受或拒绝 Transit Gateway 关联提案](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `AcceptDirectConnectGatewayAssociationProposal`](#) 中的。

## allocate-connection-on-interconnect

以下代码示例演示了如何使用 `allocate-connection-on-interconnect`。

### AWS CLI

在互连上创建托管连接

以下 `allocate-connection-on-interconnect` 命令在互连上创建托管连接：

```
aws directconnect allocate-connection-on-interconnect --bandwidth 500Mbps --
connection-name mydcinterconnect --owner-account 123456789012 --interconnect-id
dxcon-fgktov66 --vlan 101
```

输出：

```
{
  "partnerName": "TIVIT",
  "vlan": 101,
  "ownerAccount": "123456789012",
```

```
"connectionId": "dxcon-ffzc51m1",
"connectionState": "ordering",
"bandwidth": "500Mbps",
"location": "TIVIT",
"connectionName": "mydcinterconnect",
"region": "sa-east-1"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AllocateConnectionOnInterconnect](#)中的。

## allocate-hosted-connection

以下代码示例演示了如何使用 `allocate-hosted-connection`。

### AWS CLI

在互连上创建托管连接

以下`allocate-hosted-connection`示例在指定的互连上创建托管连接。

```
aws directconnect allocate-hosted-connection \
  --bandwidth 500Mbps \
  --connection-name mydcinterconnect \
  --owner-account 123456789012
  -connection-id dxcon-fgktov66
  -vlan 101
```

输出：

```
{
  "partnerName": "TIVIT",
  "vlan": 101,
  "ownerAccount": "123456789012",
  "connectionId": "dxcon-ffzc51m1",
  "connectionState": "ordering",
  "bandwidth": "500Mbps",
  "location": "TIVIT",
  "connectionName": "mydcinterconnect",
  "region": "sa-east-1"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AllocateHostedConnection](#)中的。



## allocate-private-virtual-interface

以下代码示例演示了如何使用 `allocate-private-virtual-interface`。

### AWS CLI

#### 配置私有虚拟接口

以下 `allocate-private-virtual-interface` 命令配置一个由其他客户拥有的私有虚拟接口：

```
aws directconnect allocate-private-virtual-interface --connection-id dxcon-ffjrnx17 --owner-account 123456789012 --new-private-virtual-interface-allocation virtualInterfaceName=PrivateVirtualInterface,vlan=1000,asn=65000,authKey=asdf34example,amaz
```

输出：

```
{
  "virtualInterfaceState": "confirming",
  "asn": 65000,
  "vlan": 1000,
  "customerAddress": "192.168.1.2/30",
  "ownerAccount": "123456789012",
  "connectionId": "dxcon-ffjrnx17",
  "virtualInterfaceId": "dxvif-fgy8orxu",
  "authKey": "asdf34example",
  "routeFilterPrefixes": [],
  "location": "TIVIT",
  "customerRouterConfig": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n <logical_connection id=\"dxvif-fgy8orxu\">\n <vlan>1000</vlan>\n <customer_address>192.168.1.2/30</customer_address>\n <amazon_address>192.168.1.1/30</amazon_address>\n <bgp_asn>65000</bgp_asn>\n <bgp_auth_key>asdf34example</bgp_auth_key>\n <amazon_bgp_asn>7224</amazon_bgp_asn>\n <connection_type>private</connection_type>\n</logical_connection>\n",
  "amazonAddress": "192.168.1.1/30",
  "virtualInterfaceType": "private",
  "virtualInterfaceName": "PrivateVirtualInterface"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AllocatePrivateVirtualInterface](#)中的。

## allocate-public-virtual-interface

以下代码示例演示了如何使用 `allocate-public-virtual-interface`。

## AWS CLI

### 配置公共虚拟接口

以下`allocate-public-virtual-interface`命令配置一个由其他客户拥有的公共虚拟接口：

```
aws directconnect allocate-public-virtual-interface --connection-id dxcon-ffjrkrx17 --owner-account 123456789012 --new-public-virtual-interface-allocation virtualInterfaceName=PublicVirtualInterface,vlan=2000,asn=65000,authKey=asdf34example,amazonAddress=203.0.113.1/30,{cidr=203.0.113.4/30}]
```

输出：

```
{
  "virtualInterfaceState": "confirming",
  "asn": 65000,
  "vlan": 2000,
  "customerAddress": "203.0.113.2/30",
  "ownerAccount": "123456789012",
  "connectionId": "dxcon-ffjrkrx17",
  "virtualInterfaceId": "dxvif-fg9xo9vp",
  "authKey": "asdf34example",
  "routeFilterPrefixes": [
    {
      "cidr": "203.0.113.0/30"
    },
    {
      "cidr": "203.0.113.4/30"
    }
  ],
  "location": "TIVIT",
  "customerRouterConfig": "<?xml version='1.0' encoding='UTF-8'?>\n<logical_connection id='dxvif-fg9xo9vp'>\n  <vlan>2000</vlan>\n  <customer_address>203.0.113.2/30</customer_address>\n  <amazon_address>203.0.113.1/30</amazon_address>\n  <bgp_asn>65000</bgp_asn>\n  <bgp_auth_key>asdf34example</bgp_auth_key>\n  <amazon_bgp_asn>7224</amazon_bgp_asn>\n  <connection_type>public</connection_type>\n</logical_connection>\n",
  "amazonAddress": "203.0.113.1/30",
  "virtualInterfaceType": "public",
  "virtualInterfaceName": "PublicVirtualInterface"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AllocatePublicVirtualInterface](#)中的。

## allocate-transit-virtual-interface

以下代码示例演示了如何使用 `allocate-transit-virtual-interface`。

### AWS CLI

配置由指定 AWS 账户拥有的传输虚拟接口

以下 `allocate-transit-virtual-interface` 示例为指定账户配置传输虚拟接口。

```
aws directconnect allocate-transit-virtual-interface \  
  --connection-id dxlag-fEXAMPLE \  
  --owner-account 123456789012 \  
  --new-transit-virtual-interface-allocation "virtualInterfaceName=Example Transit  
Virtual  
Interface,vlan=126,asn=65110,mtu=1500,authKey=0xzxcgA9YoW9h58u8SEXAMPLE,amazonAddress=192.16
```

输出：

```
{  
  "virtualInterface": {  
    "ownerAccount": "123456789012",  
    "virtualInterfaceId": "dxvif-fEXAMPLE",  
    "location": "loc1",  
    "connectionId": "dxlag-fEXAMPLE",  
    "virtualInterfaceType": "transit",  
    "virtualInterfaceName": "Example Transit Virtual Interface",  
    "vlan": 126,  
    "asn": 65110,  
    "amazonSideAsn": 7224,  
    "authKey": "0xzxcgA9YoW9h58u8SEXAMPLE",  
    "amazonAddress": "192.168.1.1/30",  
    "customerAddress": "192.168.1.2/30",  
    "addressFamily": "ipv4",  
    "virtualInterfaceState": "confirming",  
    "customerRouterConfig": "<?xml version=\"1.0\" encoding=  
\"UTF-8\"?>\n<logical_connection id=\"dxvif-fEXAMPLE\">\n  <vlan>126</  
vlan>\n  <customer_address>192.168.1.2/30</customer_address>\n  <amazon_address>192.168.1.1/30</amazon_address>\n  <bgp_asn>65110</bgp_asn>\n  <bgp_auth_key>0xzxcgA9YoW9h58u8SEXAMPLE</bgp_auth_key>\n  <amazon_bgp_asn>7224</  
amazon_bgp_asn>\n  <connection_type>transit</connection_type>\n</logical_connection>  
\n",  
    "mtu": 1500,
```

```

    "jumboFrameCapable": true,
    "virtualGatewayId": "",
    "directConnectGatewayId": "",
    "routeFilterPrefixes": [],
    "bgpPeers": [
      {
        "bgpPeerId": "dxpeer-fEXAMPLE",
        "asn": 65110,
        "authKey": "0xzxcgA9YoW9h58u8EXAMPLE",
        "addressFamily": "ipv4",
        "amazonAddress": "192.168.1.1/30",
        "customerAddress": "192.168.1.2/30",
        "bgpPeerState": "pending",
        "bgpStatus": "down",
        "awsDeviceV2": "loc1-26wz6vEXAMPLE"
      }
    ],
    "region": "sa-east-1",
    "awsDeviceV2": "loc1-26wz6vEXAMPLE",
    "tags": [
      {
        "key": "Tag",
        "value": "Example"
      }
    ]
  }
}

```

有关更多信息，请参阅 Di AWS rect Connect 用户指南中的[创建托管公交虚拟接口](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AllocateTransitVirtualInterface](#)中的。

## associate-connection-with-lag

以下代码示例演示了如何使用 associate-connection-with-lag。

### AWS CLI

将连接与 LAG 关联

以下示例将指定的连接与指定的 LAG 相关联。

命令:

```
aws directconnect associate-connection-with-lag --lag-id dxlag-fhccu14t --
connection-id dxcon-fg9607vm
```

输出：

```
{
  "ownerAccount": "123456789012",
  "connectionId": "dxcon-fg9607vm",
  "lagId": "dxlag-fhccu14t",
  "connectionState": "requested",
  "bandwidth": "1Gbps",
  "location": "EqDC2",
  "connectionName": "Con2ForLag",
  "region": "us-east-1"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateConnectionWithLag](#)中的。

## associate-hosted-connection

以下代码示例演示了如何使用 `associate-hosted-connection`。

### AWS CLI

将托管连接与 LAG 关联

以下示例将指定的托管连接与指定的 LAG 相关联。

命令：

```
aws directconnect associate-hosted-connection --parent-connection-id dxlag-fhccu14t
--connection-id dxcon-fg9607vm
```

输出：

```
{
  "partnerName": "TIVIT",
  "vlan": 101,
  "ownerAccount": "123456789012",
  "connectionId": "dxcon-fg9607vm",
}
```

```
"lagId": "dxlag-fhccu14t",
"connectionState": "ordering",
"bandwidth": "500Mbps",
"location": "TIVIT",
"connectionName": "mydcinterconnect",
"region": "sa-east-1"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateHostedConnection](#)中的。

## associate-virtual-interface

以下代码示例演示了如何使用 `associate-virtual-interface`。

### AWS CLI

将虚拟接口与连接关联

以下示例将指定的虚拟接口与指定的 LAG 相关联。或者，要将虚拟接口与连接关联，请为其指定 Di AWS rect Connect 连接的 ID `--connection-id`；例如，`dxcon-ffnikghc`。

命令：

```
aws directconnect associate-virtual-interface --connection-id dxlag-ffjhj91x --
virtual-interface-id dxvif-fgputw0j
```

输出：

```
{
  "virtualInterfaceState": "pending",
  "asn": 65000,
  "vlan": 123,
  "customerAddress": "169.254.255.2/30",
  "ownerAccount": "123456789012",
  "connectionId": "dxlag-ffjhj91x",
  "addressFamily": "ipv4",
  "virtualGatewayId": "vgw-38e90b51",
  "virtualInterfaceId": "dxvif-fgputw0j",
  "authKey": "0x123pK5_VBqv.UQ3kJ4123_",
  "routeFilterPrefixes": [],
  "location": "CSVA1",
```

```

"bgpPeers": [
  {
    "bgpStatus": "down",
    "customerAddress": "169.254.255.2/30",
    "addressFamily": "ipv4",
    "authKey": "0x123pK5_VBqv.UQ3kJ4123_",
    "bgpPeerState": "deleting",
    "amazonAddress": "169.254.255.1/30",
    "asn": 65000
  },
  {
    "bgpStatus": "down",
    "customerAddress": "169.254.255.2/30",
    "addressFamily": "ipv4",
    "authKey": "0x123pK5_VBqv.UQ3kJ4123_",
    "bgpPeerState": "pending",
    "amazonAddress": "169.254.255.1/30",
    "asn": 65000
  }
],
"customerRouterConfig": "<?xml version=\"1.0\" encoding=\"UTF-8\"?
>\n<logical_connection id=\"dxvif-fgputw0j\">\n  <vlan>123</vlan>
\n  <customer_address>169.254.255.2/30</customer_address>\n
  <amazon_address>169.254.255.1/30</amazon_address>\n  <bgp_asn>65000</bgp_asn>\n
  <bgp_auth_key>0x123pK5_VBqv.UQ3kJ4123_</bgp_auth_key>\n  <amazon_bgp_asn>7224</
amazon_bgp_asn>\n  <connection_type>private</connection_type>\n</logical_connection>
\n",
"amazonAddress": "169.254.255.1/30",
"virtualInterfaceType": "private",
"virtualInterfaceName": "VIF1A"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateVirtualInterface](#)中的。

## confirm-connection

以下代码示例演示了如何使用 confirm-connection。

### AWS CLI

确认在互连上创建托管连接

以下confirm-connection命令确认在互连上创建托管连接：

```
aws directconnect confirm-connection --connection-id dxcon-fg2wi7hy
```

输出：

```
{
  "connectionState": "pending"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ConfirmConnection](#)中的。

## confirm-private-virtual-interface

以下代码示例演示了如何使用 `confirm-private-virtual-interface`。

AWS CLI

接受私有虚拟接口的所有权

以下`confirm-private-virtual-interface`命令接受其他客户创建的私有虚拟接口的所有权：

```
aws directconnect confirm-private-virtual-interface --virtual-interface-id dxvif-fgy8orxu --virtual-gateway-id vgw-e4a47df9
```

输出：

```
{
  "virtualInterfaceState": "pending"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ConfirmPrivateVirtualInterface](#)中的。

## confirm-public-virtual-interface

以下代码示例演示了如何使用 `confirm-public-virtual-interface`。

AWS CLI

接受公共虚拟接口的所有权



以下 `confirm-public-virtual-interface` 命令接受其他客户创建的公共虚拟接口的所有权：

```
aws directconnect confirm-public-virtual-interface --virtual-interface-id dxvif-  
fg9xo9vp
```

输出：

```
{  
  "virtualInterfaceState": "verifying"  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ConfirmPublicVirtualInterface](#) 中的。

## **confirm-transit-virtual-interface**

以下代码示例演示了如何使用 `confirm-transit-virtual-interface`。

### AWS CLI

接受传输虚拟接口的所有权

以下内容 `confirm-transit-virtual-interface` 接受其他客户创建的传输虚拟接口的所有权。

```
aws directconnect confirm-transit-virtual-interface \  
  --virtual-interface-id dxvif-fEXAMPLE \  
  --direct-connect-gateway-id 4112ccf9-25e9-4111-8237-b6c5dEXAMPLE
```

输出：

```
{  
  "virtualInterfaceState": "pending"  
}
```

有关更多信息，请参阅 *Di AWS rect Connect 用户指南* 中的 [接受托管虚拟接口](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ConfirmTransitVirtualInterface](#) 中的。

## **create-bgp-peer**

以下代码示例演示了如何使用 `create-bgp-peer`。

## AWS CLI

### 创建 IPv6 BGP 对等会话

以下示例在私有虚拟接口上创建 IPv6 BGP 对等会话。dxvif-fg1vuj3d 对等 IPv6 地址由亚马逊自动分配。

命令:

```
aws directconnect create-bgp-peer --virtual-interface-id dxvif-fg1vuj3d --new-bgp-peer asn=64600,addressFamily=ipv6
```

输出:

```
{
  "virtualInterface": {
    "virtualInterfaceState": "available",
    "asn": 65000,
    "vlan": 125,
    "customerAddress": "169.254.255.2/30",
    "ownerAccount": "123456789012",
    "connectionId": "dxcon-fguhmq1c",
    "addressFamily": "ipv4",
    "virtualGatewayId": "vgw-f9eb0c90",
    "virtualInterfaceId": "dxvif-fg1vuj3d",
    "authKey": "0xC_ukbCer16EYA0example",
    "routeFilterPrefixes": [],
    "location": "EqDC2",
    "bgpPeers": [
      {
        "bgpStatus": "down",
        "customerAddress": "169.254.255.2/30",
        "addressFamily": "ipv4",
        "authKey": "0xC_ukbCer16EYA0uexample",
        "bgpPeerState": "available",
        "amazonAddress": "169.254.255.1/30",
        "asn": 65000
      },
      {
        "bgpStatus": "down",
        "customerAddress": "2001:db8:1100:2f0:0:1:9cb4:4216/125",
        "addressFamily": "ipv6",
        "authKey": "0xS27kAIU_VHPjjAexample",

```

```

        "bgpPeerState": "pending",
        "amazonAddress": "2001:db8:1100:2f0:0:1:9cb4:4211/125",
        "asn": 64600
    }
],
"customerRouterConfig": "<?xml version=\"1.0\" encoding=
\"UTF-8\"?>\n<logical_connection id=\"dxvif-fg1vuj3d\">\n  <vlan>125</
vlan>\n  <customer_address>169.254.255.2/30</customer_address>\n
  <amazon_address>169.254.255.1/30</amazon_address>\n  <bgp_asn>65000</
bgp_asn>\n  <bgp_auth_key>0xC_ukbCerl6EYA0uexample</bgp_auth_key>\n
  <ipv6_customer_address>2001:db8:1100:2f0:0:1:9cb4:4216/125</ipv6_customer_address>
\n  <ipv6_amazon_address>2001:db8:1100:2f0:0:1:9cb4:4211/125</ipv6_amazon_address>\n
  <ipv6_bgp_asn>64600</ipv6_bgp_asn>\n  <ipv6_bgp_auth_key>0xS27kAIU_VHPjjAexample</
ipv6_bgp_auth_key>\n  <amazon_bgp_asn>7224</amazon_bgp_asn>\n
  <connection_type>private</connection_type>\n</logical_connection>\n",
    "amazonAddress": "169.254.255.1/30",
    "virtualInterfaceType": "private",
    "virtualInterfaceName": "Test"
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateBgpPeer](#)中的。

## create-connection

以下代码示例演示了如何使用 create-connection。

### AWS CLI

创建从您的网络到 Di AWS rect Connect 位置的连接

以下create-connection命令创建从您的网络到 Di AWS rect Connect 位置的连接：

```
aws directconnect create-connection --location TIVIT --bandwidth 1Gbps --connection-
name "Connection to AWS"
```

输出：

```
{
  "ownerAccount": "123456789012",
  "connectionId": "dxcon-fg31dyv6",
  "connectionState": "requested",
```

```
"bandwidth": "1Gbps",
"location": "TIVIT",
"connectionName": "Connection to AWS",
"region": "sa-east-1"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateConnection](#)中的。

## create-direct-connect-gateway-association-proposal

以下代码示例演示了如何使用 `create-direct-connect-gateway-association-proposal`。

### AWS CLI

创建将指定传输网关与指定的 Direct Connect 网关关联的提案

以下`create-direct-connect-gateway-association-proposal`示例创建了一个提案，该提案将指定的传输网关与指定的 Direct Connect 网关关联起来。

```
aws directconnect create-direct-connect-gateway-association-proposal \
  --direct-connect-gateway-id 11460968-4ac1-4fd3-bdb2-00599EXAMPLE \
  --direct-connect-gateway-owner-account 111122223333 \
  --gateway-id tgw-02f776b1a7EXAMPLE \
  --add-allowed-prefixes-to-direct-connect-gateway cidr=192.168.1.0/30
```

输出：

```
{
  "directConnectGatewayAssociationProposal": {
    "proposalId": "cb7f41cb-8128-43a5-93b1-dcaedEXAMPLE",
    "directConnectGatewayId": "11460968-4ac1-4fd3-bdb2-00599EXAMPLE",
    "directConnectGatewayOwnerAccount": "111122223333",
    "proposalState": "requested",
    "associatedGateway": {
      "id": "tgw-02f776b1a7EXAMPLE",
      "type": "transitGateway",
      "ownerAccount": "111122223333",
      "region": "us-east-1"
    },
    "requestedAllowedPrefixesToDirectConnectGateway": [
      {
```

```

        "cidr": "192.168.1.0/30"
      }
    ]
  }
}

```

有关更多信息，请参阅 [Di AWS Direct Connect 用户指南中的创建 Transit Gateway 关联提案](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `CreateDirectConnectGatewayAssociationProposal`](#) 中的。

## create-direct-connect-gateway-association

以下代码示例演示了如何使用 `create-direct-connect-gateway-association`。

### AWS CLI

将虚拟专用网关与 Direct Connect 网关关联

以下示例将虚拟专用网关 `vgw-6efe725e` 与 Direct Connect 网关 `5f294f92-bafb-4011-916d-9b0bexample` 相关联。您必须在虚拟专用网关所在的区域中运行该命令。

命令：

```
aws directconnect create-direct-connect-gateway-association --direct-connect-gateway-id 5f294f92-bafb-4011-916d-9b0bexample --virtual-gateway-id vgw-6efe725e
```

输出：

```
{
  "directConnectGatewayAssociation": {
    "associationState": "associating",
    "virtualGatewayOwnerAccount": "123456789012",
    "directConnectGatewayId": "5f294f92-bafb-4011-916d-9b0bexample",
    "virtualGatewayId": "vgw-6efe725e",
    "virtualGatewayRegion": "us-east-2"
  }
}
```

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `CreateDirectConnectGatewayAssociation`](#) 中的。

## create-direct-connect-gateway

以下代码示例演示了如何使用 `create-direct-connect-gateway`。

### AWS CLI

创建 Direct Connect 网关

以下示例创建了一个名为的 Direct Connect 网关DxGateway1。

命令:

```
aws directconnect create-direct-connect-gateway --direct-connect-gateway-name
"DxGateway1"
```

输出 :

```
{
  "directConnectGateway": {
    "amazonSideAsn": 64512,
    "directConnectGatewayId": "5f294f92-bafb-4011-916d-9b0bdexample",
    "ownerAccount": "123456789012",
    "directConnectGatewayName": "DxGateway1",
    "directConnectGatewayState": "available"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDirectConnectGateway](#)中的。

## create-interconnect

以下代码示例演示了如何使用 `create-interconnect`。

### AWS CLI

在合作伙伴的网络和之间建立互连 AWS

以下`create-interconnect`命令在 Direct Connect 合作伙伴的网络和特定 AWS 的 Di AWS rect Connect 位置之间创建互连：

```
aws directconnect create-interconnect --interconnect-name "1G Interconnect to AWS"
--bandwidth 1Gbps --location TIVIT
```

输出：

```
{
  "region": "sa-east-1",
  "bandwidth": "1Gbps",
  "location": "TIVIT",
  "interconnectName": "1G Interconnect to AWS",
  "interconnectId": "dxcon-fgktov66",
  "interconnectState": "requested"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateInterconnect](#)中的。

## create-lag

以下代码示例演示了如何使用 create-lag。

### AWS CLI

#### 使用新连接创建 LAG

以下示例创建了一个 LAG，并为 LAG 请求两个带宽为 1 Gbps 的新 AWS Direct Connect 连接。

命令：

```
aws directconnect create-lag --location CSVA1 --number-of-connections 2 --
connections-bandwidth 1Gbps --lag-name 1GBLag
```

输出：

```
{
  "awsDevice": "CSVA1-23u8tlpaz8iks",
  "numberOfConnections": 2,
  "lagState": "pending",
  "ownerAccount": "123456789012",
  "lagName": "1GBLag",
  "connections": [
    {
      "ownerAccount": "123456789012",
      "connectionId": "dxcon-ffqr6x5q",
      "lagId": "dxlag-ffjhj9lx",
    }
  ]
}
```

```

        "connectionState": "requested",
        "bandwidth": "1Gbps",
        "location": "CSVA1",
        "connectionName": "Requested Connection 1 for Lag dxlag-ffjhj9lx",
        "region": "us-east-1"
    },
    {
        "ownerAccount": "123456789012",
        "connectionId": "dxcon-fflqyj95",
        "lagId": "dxlag-ffjhj9lx",
        "connectionState": "requested",
        "bandwidth": "1Gbps",
        "location": "CSVA1",
        "connectionName": "Requested Connection 2 for Lag dxlag-ffjhj9lx",
        "region": "us-east-1"
    }
],
"lagId": "dxlag-ffjhj9lx",
"minimumLinks": 0,
"connectionsBandwidth": "1Gbps",
"region": "us-east-1",
"location": "CSVA1"
}

```

## 使用现有连接创建 LAG

以下示例使用您账户中的现有连接创建 LAG，并请求使用与现有连接相同的带宽和位置为 LAG 创建第二个新连接。

命令:

```
aws directconnect create-lag --location EqDC2 --number-of-connections 2 --
connections-bandwidth 1Gbps --lag-name 2ConnLAG --connection-id dxcon-fgk145dr
```

输出:

```

{
  "awsDevice": "EqDC2-4h6ce2r1bes6",
  "numberOfConnections": 2,
  "lagState": "pending",
  "ownerAccount": "123456789012",
  "lagName": "2ConnLAG",

```



```

"connections": [
  {
    "ownerAccount": "123456789012",
    "connectionId": "dxcon-fh6ljcvo",
    "lagId": "dxlag-fhccu14t",
    "connectionState": "requested",
    "bandwidth": "1Gbps",
    "location": "EqDC2",
    "connectionName": "Requested Connection 1 for Lag dxlag-fhccu14t",
    "region": "us-east-1"
  },
  {
    "ownerAccount": "123456789012",
    "connectionId": "dxcon-fgk145dr",
    "lagId": "dxlag-fhccu14t",
    "connectionState": "down",
    "bandwidth": "1Gbps",
    "location": "EqDC2",
    "connectionName": "VAConn1",
    "region": "us-east-1"
  }
],
"lagId": "dxlag-fhccu14t",
"minimumLinks": 0,
"connectionsBandwidth": "1Gbps",
"region": "us-east-1",
"location": "EqDC2"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLag](#)中的。

## create-private-virtual-interface

以下代码示例演示了如何使用 create-private-virtual-interface。

### AWS CLI

#### 创建私有虚拟接口

以下create-private-virtual-interface命令创建私有虚拟接口：

```

aws directconnect create-private-virtual-interface --
connection-id dxcon-ffjrnx17 --new-private-virtual-interface

```

```
virtualInterfaceName=PrivateVirtualInterface,vlan=101,asn=65000,authKey=asdf34example,amazonAddress=192.168.1.1/30,amazonAsn=7224,amazonAuthKey=asdf34example,amazonBgpAsn=7224,amazonBgpAuthKey=asdf34example,amazonConnectionType=private,amazonVlan=101,amazonVpnId=vpn-123456789012,amazonVpnIdPrefix=vpn-
```

输出：

```
{
  "virtualInterfaceState": "pending",
  "asn": 65000,
  "vlan": 101,
  "customerAddress": "192.168.1.2/30",
  "ownerAccount": "123456789012",
  "connectionId": "dxcon-ffjrkx17",
  "virtualGatewayId": "vgw-aba37db6",
  "virtualInterfaceId": "dxvif-ffhkh74f",
  "authKey": "asdf34example",
  "routeFilterPrefixes": [],
  "location": "TIVIT",
  "customerRouterConfig": "<?xml version='1.0' encoding='UTF-8'?'>\n<logical_connection id='dxvif-ffhkh74f'>\n  <vlan>101</vlan>\n  <customer_address>192.168.1.2/30</customer_address>\n  <amazon_address>192.168.1.1/30</amazon_address>\n  <bgp_asn>65000</bgp_asn>\n  <bgp_auth_key>asdf34example</bgp_auth_key>\n  <amazon_bgp_asn>7224</amazon_bgp_asn>\n  <connection_type>private</connection_type>\n</logical_connection>\n",
  "amazonAddress": "192.168.1.1/30",
  "virtualInterfaceType": "private",
  "virtualInterfaceName": "PrivateVirtualInterface"
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreatePrivateVirtualInterface](#) 中的。

## create-public-virtual-interface

以下代码示例演示了如何使用 `create-public-virtual-interface`。

### AWS CLI

#### 创建公共虚拟接口

以下 `create-public-virtual-interface` 命令创建公共虚拟接口：

```
aws directconnect create-public-virtual-interface --
connection-id dxcon-ffjrkx17 --new-public-virtual-interface
```

```
virtualInterfaceName=PublicVirtualInterface,vlan=2000,asn=65000,authKey=asdf34example,amazonAddress=203.0.113.1/30,{cidr=203.0.113.4/30}]
```

输出：

```
{
  "virtualInterfaceState": "verifying",
  "asn": 65000,
  "vlan": 2000,
  "customerAddress": "203.0.113.2/30",
  "ownerAccount": "123456789012",
  "connectionId": "dxcon-ffjrkrx17",
  "virtualInterfaceId": "dxvif-fgh0hcrk",
  "authKey": "asdf34example",
  "routeFilterPrefixes": [
    {
      "cidr": "203.0.113.0/30"
    },
    {
      "cidr": "203.0.113.4/30"
    }
  ],
  "location": "TIVIT",
  "customerRouterConfig": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<logical_connection id=\"dxvif-fgh0hcrk\">\n  <vlan>2000</vlan>\n  <customer_address>203.0.113.2/30</customer_address>\n  <amazon_address>203.0.113.1/30</amazon_address>\n  <bgp_asn>65000</bgp_asn>\n  <bgp_auth_key>asdf34example</bgp_auth_key>\n  <amazon_bgp_asn>7224</amazon_bgp_asn>\n  <connection_type>public</connection_type>\n</logical_connection>\n",
  "amazonAddress": "203.0.113.1/30",
  "virtualInterfaceType": "public",
  "virtualInterfaceName": "PublicVirtualInterface"
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreatePublicVirtualInterface](#) 中的。

## create-transit-virtual-interface

以下代码示例演示了如何使用 create-transit-virtual-interface。

### AWS CLI

#### 创建中转虚拟接口

以下create-transit-virtual-interface示例为指定连接创建中转虚拟接口。

```
aws directconnect create-transit-virtual-interface \
  --connection-id dxlag-fEXAMPLE \
  --new-transit-virtual-interface "virtualInterfaceName=Example Transit Virtual
  Interface,vlan=126,asn=65110,mtu=1500,authKey=0xzxcgA9YoW9h58u8SvEXAMPLE,amazonAddress=192.1
  aada-5a1baEXAMPLE,tags=[{key=Tag,value=Example}]"
```

输出：

```
{
  "virtualInterface": {
    "ownerAccount": "1111222233333",
    "virtualInterfaceId": "dxvif-fEXAMPLE",
    "location": "loc1",
    "connectionId": "dxlag-fEXAMPLE",
    "virtualInterfaceType": "transit",
    "virtualInterfaceName": "Example Transit Virtual Interface",
    "vlan": 126,
    "asn": 65110,
    "amazonSideAsn": 4200000000,
    "authKey": "0xzxcgA9YoW9h58u8SEXAMPLE",
    "amazonAddress": "192.168.1.1/30",
    "customerAddress": "192.168.1.2/30",
    "addressFamily": "ipv4",
    "virtualInterfaceState": "pending",
    "customerRouterConfig": "<?xml version='1.0' encoding=
  \"UTF-8\"?>\n<logical_connection id='dxvif-fEXAMPLE'\>\n  <vlan>126</
  vlan>\n  <customer_address>192.168.1.2/30</customer_address>\n
  <amazon_address>192.168.1.1/30</amazon_address>\n  <bgp_asn>65110</
  bgp_asn>\n  <bgp_auth_key>0xzxcgA9YoW9h58u8Sv0mXRTw</bgp_auth_key>\n
  <amazon_bgp_asn>4200000000</amazon_bgp_asn>\n  <connection_type>transit</
  connection_type>\n</logical_connection>\n",
    "mtu": 1500,
    "jumboFrameCapable": true,
    "virtualGatewayId": "",
    "directConnectGatewayId": "8384da05-13ce-4a91-aada-5a1baEXAMPLE",
    "routeFilterPrefixes": [],
    "bgpPeers": [
      {
        "bgpPeerId": "dxpeer-EXAMPLE",
        "asn": 65110,
        "authKey": "0xzxcgA9YoW9h58u8SEXAMPLE",
```

```
        "addressFamily": "ipv4",
        "amazonAddress": "192.168.1.1/30",
        "customerAddress": "192.168.1.2/30",
        "bgpPeerState": "pending",
        "bgpStatus": "down",
        "awsDeviceV2": "loc1-26wz6vEXAMPLE"
    }
],
"region": "sa-east-1",
"awsDeviceV2": "loc1-26wz6vEXAMPLE",
"tags": [
    {
        "key": "Tag",
        "value": "Example"
    }
]
}
}
```

有关更多信息，请参阅 Direct Connect 用户指南中的创建到 Direct Connect [网关的AWS 传输虚拟接口](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTransitVirtualInterface](#)中的。

## delete-bgp-peer

以下代码示例演示了如何使用 delete-bgp-peer。

### AWS CLI

从虚拟接口删除 BGP 对等体

以下示例从虚拟接口dxvif-fg1vuj3d删除 IPv6 BGP 对等体。

命令:

```
aws directconnect delete-bgp-peer --virtual-interface-id dxvif-fg1vuj3d --asn 64600
--customer-address 2001:db8:1100:2f0:0:1:9cb4:4216/125
```

输出:

```
{
  "virtualInterface": {
```

```

    "virtualInterfaceState": "available",
    "asn": 65000,
    "vlan": 125,
    "customerAddress": "169.254.255.2/30",
    "ownerAccount": "123456789012",
    "connectionId": "dxcon-fguhmq1c",
    "addressFamily": "ipv4",
    "virtualGatewayId": "vgw-f9eb0c90",
    "virtualInterfaceId": "dxvif-fg1vuj3d",
    "authKey": "0xC_ukbCer16EYA0example",
    "routeFilterPrefixes": [],
    "location": "EqDC2",
    "bgpPeers": [
      {
        "bgpStatus": "down",
        "customerAddress": "169.254.255.2/30",
        "addressFamily": "ipv4",
        "authKey": "0xC_ukbCer16EYA0uexample",
        "bgpPeerState": "available",
        "amazonAddress": "169.254.255.1/30",
        "asn": 65000
      },
      {
        "bgpStatus": "down",
        "customerAddress": "2001:db8:1100:2f0:0:1:9cb4:4216/125",
        "addressFamily": "ipv6",
        "authKey": "0xS27kAIU_VHPjjAexample",
        "bgpPeerState": "deleting",
        "amazonAddress": "2001:db8:1100:2f0:0:1:9cb4:4211/125",
        "asn": 64600
      }
    ],
    "customerRouterConfig": "<?xml version=\"1.0\" encoding=
\\\"UTF-8\\\"?>\\n<logical_connection id=\\\"dxvif-fg1vuj3d\\\">\\n  <vlan>125</
vlan>\\n  <customer_address>169.254.255.2/30</customer_address>\\n
  <amazon_address>169.254.255.1/30</amazon_address>\\n  <bgp_asn>65000</bgp_asn>\\n
  <bgp_auth_key>0xC_ukbCer16EYA0example</bgp_auth_key>\\n  <amazon_bgp_asn>7224</
amazon_bgp_asn>\\n  <connection_type>private</connection_type>\\n</logical_connection>
\\n",
    "amazonAddress": "169.254.255.1/30",
    "virtualInterfaceType": "private",
    "virtualInterfaceName": "Test"
  }
}

```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBgpPeer](#)中的。

## delete-connection

以下代码示例演示了如何使用 delete-connection。

### AWS CLI

#### 删除连接

以下delete-connection命令删除指定的连接：

```
aws directconnect delete-connection --connection-id dxcon-fg31dyv6
```

输出：

```
{
  "ownerAccount": "123456789012",
  "connectionId": "dxcon-fg31dyv6",
  "connectionState": "deleted",
  "bandwidth": "1Gbps",
  "location": "TIVIT",
  "connectionName": "Connection to AWS",
  "region": "sa-east-1"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteConnection](#)中的。

## delete-direct-connect-gateway-association

以下代码示例演示了如何使用 delete-direct-connect-gateway-association。

### AWS CLI

#### 删除 Direct Connect 网关关联

以下delete-direct-connect-gateway-association示例删除了与具有指定关联 ID 的传输网关的 Direct Connect 网关关联。

```
aws directconnect delete-direct-connect-gateway-association --association-id
be85116d-46eb-4b43-a27a-da0c2ad648de
```

输出：

```
{
  "directConnectGatewayAssociation": {
    "directConnectGatewayId": "11460968-4ac1-4fd3-bdb2-00599EXAMPLE",
    "directConnectGatewayOwnerAccount": "123456789012",
    "associationState": "disassociating",
    "associatedGateway": {
      "id": "tgw-095b3b0b54EXAMPLE",
      "type": "transitGateway",
      "ownerAccount": "123456789012",
      "region": "us-east-1"
    },
    "associationId": " be85116d-46eb-4b43-a27a-da0c2ad648deEXAMPLE ",
    "allowedPrefixesToDirectConnectGateway": [
      {
        "cidr": "192.0.1.0/28"
      }
    ]
  }
}
```

有关更多信息，请参阅 Direct Connect 用户指南中的[关联和取消关联中转网关](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteDirectConnectGatewayAssociation](#) 中的。

## delete-direct-connect-gateway

以下代码示例演示了如何使用 delete-direct-connect-gateway。

### AWS CLI

删除 Direct Connect 网关

以下示例删除了 Direct Connect 网关 5f294f92-bafb-4011-916d-9b0bexample。

命令：



```
aws directconnect delete-direct-connect-gateway --direct-connect-gateway-id
5f294f92-bafb-4011-916d-9b0bexample
```

输出：

```
{
  "directConnectGateway": {
    "amazonSideAsn": 64512,
    "directConnectGatewayId": "5f294f92-bafb-4011-916d-9b0bexample",
    "ownerAccount": "123456789012",
    "directConnectGatewayName": "DxGateway1",
    "directConnectGatewayState": "deleting"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDirectConnectGateway](#)中的。

## delete-interconnect

以下代码示例演示了如何使用 delete-interconnect。

AWS CLI

删除互连

以下delete-interconnect命令删除指定的互连：

```
aws directconnect delete-interconnect --interconnect-id dxcon-fgktov66
```

输出：

```
{
  "interconnectState": "deleted"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInterconnect](#)中的。

## delete-lag

以下代码示例演示了如何使用 delete-lag。

## AWS CLI

### 删除 LAG

以下示例删除指定的 LAG。

命令:

```
aws directconnect delete-lag --lag-id dxlag-ffrhowd9
```

输出:

```
{
  "awsDevice": "EqDC2-4h6ce2r1bes6",
  "numberOfConnections": 0,
  "lagState": "deleted",
  "ownerAccount": "123456789012",
  "lagName": "TestLAG",
  "connections": [],
  "lagId": "dxlag-ffrhowd9",
  "minimumLinks": 0,
  "connectionsBandwidth": "1Gbps",
  "region": "us-east-1",
  "location": "EqDC2"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLag](#)中的。

## delete-virtual-interface

以下代码示例演示了如何使用 delete-virtual-interface。

## AWS CLI

### 删除虚拟接口

以下delete-virtual-interface命令删除指定的虚拟接口：

```
aws directconnect delete-virtual-interface --virtual-interface-id dxvif-ffhkh74f
```

输出:

```
{
  "virtualInterfaceState": "deleting"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVirtualInterface](#)中的。

## describe-connection-loa

以下代码示例演示了如何使用 describe-connection-loa。

### AWS CLI

描述使用 Linux 或 Mac OS X 进行连接的 LOA-CFA

以下示例描述了用于连接的 LOA-CFA。dxcon-fh6ayh1dLOA-CFA 的内容采用 base64 编码。此命令使用 --output 和 --query 参数来控制输出并提取 loaContent 结构的内容。命令的最后部分使用该 base64 实用程序对内容进行解码，并将输出发送到 PDF 文件。

```
aws directconnect describe-connection-loa --connection-id dxcon-fh6ayh1d --output
text --query loa.loaContent|base64 --decode > myLoaCfa.pdf
```

描述使用 Windows 进行连接的 LOA-CFA

前面的示例需要使用该 base64 实用程序来解码输出。在 Windows 计算机上，你可以 certutil 改用。在以下示例中，第一个命令描述了用于连接的 LOA-CFA，dxcon-fh6ayh1d 并使用 --output 和 --query 参数控制输出并将 loaContent 结构的内容提取到名为的文件中。myLoaCfa.base64 第二个命令使用 certutil 实用工具解码文件并将输出发送到 PDF 文件。

```
aws directconnect describe-connection-loa --connection-id dxcon-fh6ayh1d --output
text --query loa.loaContent > myLoaCfa.base64
```

```
certutil -decode myLoaCfa.base64 myLoaCfa.pdf
```

有关控制 AWS CLI 输出的更多信息，请参阅 [《AWS 命令行界面用户指南》中的通过AWS 命令行界面控制命令输出](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeConnectionLoa](#)中的。

## describe-connections-on-interconnect

以下代码示例演示了如何使用 `describe-connections-on-interconnect`。

### AWS CLI

列出互连上的连接

以下 `describe-connections-on-interconnect` 命令列出了已在给定互连上置备的连接：

```
aws directconnect describe-connections-on-interconnect --interconnect-id dxcon-  
fgktov66
```

输出：

```
{  
  "connections": [  
    {  
      "partnerName": "TIVIT",  
      "vlan": 101,  
      "ownerAccount": "123456789012",  
      "connectionId": "dxcon-ffzc51m1",  
      "connectionState": "ordering",  
      "bandwidth": "500Mbps",  
      "location": "TIVIT",  
      "connectionName": "mydcinterconnect",  
      "region": "sa-east-1"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeConnectionsOnInterconnect](#) 中的。

## describe-connections

以下代码示例演示了如何使用 `describe-connections`。

### AWS CLI

列出当前区域的所有连接

以下 `describe-connections` 命令列出了当前区域中的所有连接：

```
aws directconnect describe-connections
```

输出：

```
{
  "connections": [
    {
      "awsDevice": "EqDC2-123h49s71dabc",
      "ownerAccount": "123456789012",
      "connectionId": "dxcon-fguhmq1c",
      "lagId": "dxlag-ffrz71kw",
      "connectionState": "down",
      "bandwidth": "1Gbps",
      "location": "EqDC2",
      "connectionName": "My_Connection",
      "loaIssueTime": 1491568964.0,
      "region": "us-east-1"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeConnections](#)中的。

## describe-direct-connect-gateway-association-proposals

以下代码示例演示了如何使用 `describe-direct-connect-gateway-association-proposals`。

### AWS CLI

描述您的 Direct Connect 网关关联提案

以下 `describe-direct-connect-gateway-association-proposals` 示例显示了有关您的 Direct Connect 网关关联提案的详细信息。

```
aws directconnect describe-direct-connect-gateway-association-proposals
```

输出：

```
{
  "directConnectGatewayAssociationProposals": [
```

```
{
  "proposalId": "c2ede9b4-bbc6-4d33-923c-bc4feEXAMPLE",
  "directConnectGatewayId": "11460968-4ac1-4fd3-bdb2-00599EXAMPLE",
  "directConnectGatewayOwnerAccount": "111122223333",
  "proposalState": "requested",
  "associatedGateway": {
    "id": "tgw-02f776b1a7EXAMPLE",
    "type": "transitGateway",
    "ownerAccount": "111122223333",
    "region": "us-east-1"
  },
  "existingAllowedPrefixesToDirectConnectGateway": [
    {
      "cidr": "192.168.2.0/30"
    },
    {
      "cidr": "192.168.1.0/30"
    }
  ],
  "requestedAllowedPrefixesToDirectConnectGateway": [
    {
      "cidr": "192.168.1.0/30"
    }
  ]
},
{
  "proposalId": "cb7f41cb-8128-43a5-93b1-dcaedEXAMPLE",
  "directConnectGatewayId": "11560968-4ac1-4fd3-bcb2-00599EXAMPLE",
  "directConnectGatewayOwnerAccount": "111122223333",
  "proposalState": "accepted",
  "associatedGateway": {
    "id": "tgw-045776b1a7EXAMPLE",
    "type": "transitGateway",
    "ownerAccount": "111122223333",
    "region": "us-east-1"
  },
  "existingAllowedPrefixesToDirectConnectGateway": [
    {
      "cidr": "192.168.4.0/30"
    },
    {
      "cidr": "192.168.5.0/30"
    }
  ]
},
```

```

        "requestedAllowedPrefixesToDirectConnectGateway": [
            {
                "cidr": "192.168.5.0/30"
            }
        ]
    }
]
}

```

有关更多信息，请参阅 Direct Connect 用户指南中的[关联和取消关联中转网关](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDirectConnectGatewayAssociationProposals](#) 中的。

## describe-direct-connect-gateway-associations

以下代码示例演示了如何使用 describe-direct-connect-gateway-associations。

### AWS CLI

描述 Direct Connect 网关关联

以下示例描述了与 Direct Connect 网关的所有关联 5f294f92-bafb-4011-916d-9b0bexample。

命令:

```
aws directconnect describe-direct-connect-gateway-associations --direct-connect-gateway-id 5f294f92-bafb-4011-916d-9b0bexample
```

输出:

```

{
  "nextToken":
  "eyJ2IjoxLCJzIjoxLCJpIjoiOU83OTFodzdycnZCbkn4MExHeHVwQT09IiwieyI6InIxTEN0UEVHV0I1UF1kaWFnN1",
  "directConnectGatewayAssociations": [
    {
      "associationState": "associating",
      "virtualGatewayOwnerAccount": "123456789012",
      "directConnectGatewayId": "5f294f92-bafb-4011-916d-9b0bexample",
      "virtualGatewayId": "vgw-6efe725e",
      "virtualGatewayRegion": "us-east-2"
    }
  ]
}

```

```
    },
    {
      "associationState": "disassociating",
      "virtualGatewayOwnerAccount": "123456789012",
      "directConnectGatewayId": "5f294f92-bafb-4011-916d-9b0bexample",
      "virtualGatewayId": "vgw-ebaa27db",
      "virtualGatewayRegion": "us-east-2"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDirectConnectGatewayAssociations](#)中的。

## describe-direct-connect-gateway-attachments

以下代码示例演示了如何使用 describe-direct-connect-gateway-attachments。

### AWS CLI

描述 Direct Connect 网关附件

以下示例描述了连接到 Direct Connect 网关的虚拟接口5f294f92-bafb-4011-916d-9b0bexample。

命令:

```
aws directconnect describe-direct-connect-gateway-attachments --direct-connect-gateway-id 5f294f92-bafb-4011-916d-9b0bexample
```

输出:

```
{
  "directConnectGatewayAttachments": [
    {
      "virtualInterfaceOwnerAccount": "123456789012",
      "directConnectGatewayId": "5f294f92-bafb-4011-916d-9b0bexample",
      "virtualInterfaceRegion": "us-east-2",
      "attachmentState": "attaching",
      "virtualInterfaceId": "dxvif-fg9zyabc"
    }
  ]
}
```



```
],  
  "nextToken":  
  "eyJ2IjoxLCJzIjoxLCJpIjoibEhXd1NpUXF5RzhoL1JyUW52S1V2QT09IiwieYyI6Im5wQjFHQ0RyQUdRS3puNnNXcU  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDirectConnectGatewayAttachments](#)中的。

## describe-direct-connect-gateways

以下代码示例演示了如何使用 describe-direct-connect-gateways。

### AWS CLI

描述您的 Direct Connect 网关

以下示例描述了您的所有 Direct Connect 网关。

命令：

```
aws directconnect describe-direct-connect-gateways
```

输出：

```
{  
  "directConnectGateways": [  
    {  
      "amazonSideAsn": 64512,  
      "directConnectGatewayId": "cf68415c-f4ae-48f2-87a7-3b52cexample",  
      "ownerAccount": "123456789012",  
      "directConnectGatewayName": "DxGateway2",  
      "directConnectGatewayState": "available"  
    },  
    {  
      "amazonSideAsn": 64512,  
      "directConnectGatewayId": "5f294f92-bafb-4011-916d-9b0bdexample",  
      "ownerAccount": "123456789012",  
      "directConnectGatewayName": "DxGateway1",  
      "directConnectGatewayState": "available"  
    }  
  ]  
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDirectConnectGateways](#)中的。

## describe-hosted-connections

以下代码示例演示了如何使用 describe-hosted-connections。

### AWS CLI

列出互连上的连接

以下示例列出了已在给定互连上置备的连接。

命令:

```
aws directconnect describe-hosted-connections --connection-id dxcon-fgktov66
```

输出:

```
{
  "connections": [
    {
      "partnerName": "TIVIT",
      "vlan": 101,
      "ownerAccount": "123456789012",
      "connectionId": "dxcon-ffzc51m1",
      "connectionState": "ordering",
      "bandwidth": "500Mbps",
      "location": "TIVIT",
      "connectionName": "mydcinterconnect",
      "region": "sa-east-1"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeHostedConnections](#)中的。

## describe-interconnect-loa

以下代码示例演示了如何使用 describe-interconnect-loa。

## AWS CLI

### 描述使用 Linux 或 Mac OS X 进行互连的 LOA-CFA

以下示例描述了用于互连的 LOA-CFA。dxcon-fh6ayh1dLOA-CFA 的内容采用 base64 编码。此命令使用 `--output` 和 `--query` 参数来控制输出并提取 `loaContent` 结构的内容。命令的最后部分使用该 base64 实用程序对内容进行解码，并将输出发送到 PDF 文件。

```
aws directconnect describe-interconnect-loa --interconnect-id dxcon-fh6ayh1d --output text --query loa.loaContent|base64 --decode > myLoaCfa.pdf
```

### 描述使用 Windows 进行互连的 LOA-CFA

前面的示例需要使用该 base64 实用程序来解码输出。在 Windows 计算机上，你可以 `certutil` 改用。在以下示例中，第一个命令描述了用于互连的 LOA-CFA，dxcon-fh6ayh1d 并使用 `--output` 和 `--query` 参数控制输出并将 `loaContent` 结构的内容提取到名为 `myLoaCfa.base64` 的文件中。第二个命令使用 `certutil` 实用工具解码文件并将输出发送到 PDF 文件。

```
aws directconnect describe-interconnect-loa --interconnect-id dxcon-fh6ayh1d --output text --query loa.loaContent > myLoaCfa.base64
```

```
certutil -decode myLoaCfa.base64 myLoaCfa.pdf
```

有关控制 AWS CLI 输出的更多信息，请参阅 [《AWS 命令行界面用户指南》中的通过 AWS 命令行界面控制命令输出](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeInterconnectLoa](#) 中的。

## describe-interconnects

以下代码示例演示了如何使用 `describe-interconnects`。

### AWS CLI

#### 列出互连

以下 `describe-interconnects` 命令列出了您的 AWS 账户拥有的互连：

```
aws directconnect describe-interconnects
```

输出：

```
{
  "interconnects": [
    {
      "region": "sa-east-1",
      "bandwidth": "1Gbps",
      "location": "TIVIT",
      "interconnectName": "1G Interconnect to AWS",
      "interconnectId": "dxcon-fgktov66",
      "interconnectState": "down"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInterconnects](#)中的。

## describe-lags

以下代码示例演示了如何使用 describe-lags。

### AWS CLI

描述你的 LAG

以下命令描述了当前区域的所有 LAG。

命令：

```
aws directconnect describe-lags
```

输出：

```
{
  "lags": [
    {
      "awsDevice": "EqDC2-19y7z3m17xpuz",
      "numberOfConnections": 2,
      "lagState": "down",
      "ownerAccount": "123456789012",
      "lagName": "DA-LAG",
      "connections": [
```

```

    {
      "ownerAccount": "123456789012",
      "connectionId": "dxcon-ffnikghc",
      "lagId": "dxlag-fgsu9erb",
      "connectionState": "requested",
      "bandwidth": "10Gbps",
      "location": "EqDC2",
      "connectionName": "Requested Connection 1 for Lag dxlag-fgsu9erb",
      "region": "us-east-1"
    },
    {
      "ownerAccount": "123456789012",
      "connectionId": "dxcon-fglgbdea",
      "lagId": "dxlag-fgsu9erb",
      "connectionState": "requested",
      "bandwidth": "10Gbps",
      "location": "EqDC2",
      "connectionName": "Requested Connection 2 for Lag dxlag-fgsu9erb",
      "region": "us-east-1"
    }
  ],
  "lagId": "dxlag-fgsu9erb",
  "minimumLinks": 0,
  "connectionsBandwidth": "10Gbps",
  "region": "us-east-1",
  "location": "EqDC2"
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLags](#)中的。

## describe-loa

以下代码示例演示了如何使用 describe-loa。

### AWS CLI

描述使用 Linux 或 Mac OS X 进行连接的 LOA-CFA

以下示例描述了用于连接的 LOA-CFA。dxcon-fh6ayh1dLOA-CFA 的内容采用 base64 编码。此命令使用 --output 和 --query 参数来控制输出并提取 loaContent 结构的内容。命令的最后部分使用该 base64 实用程序对内容进行解码，并将输出发送到 PDF 文件。

```
aws directconnect describe-loa --connection-id dxcon-fh6ayh1d --output text --query loa.loaContent|base64 --decode > myLoaCfa.pdf
```

## 描述使用 Windows 进行连接的 LOA-CFA

前面的示例需要使用该base64实用程序来解码输出。在 Windows 计算机上，你可以certutil改用。在以下示例中，第一个命令描述了用于连接的 LOA-CFA，dxcon-fh6ayh1d并使用--output和--query参数控制输出并将loaContent结构的内容提取到名为的文件中。myLoaCfa.base64第二个命令使用 certutil 实用工具解码文件并将输出发送到 PDF 文件。

```
aws directconnect describe-loa --connection-id dxcon-fh6ayh1d --output text --query loa.loaContent > myLoaCfa.base64
```

```
certutil -decode myLoaCfa.base64 myLoaCfa.pdf
```

有关控制 AWS CLI 输出的更多信息，请参阅 [《AWS 命令行界面用户指南》中的通过AWS 命令行界面控制命令输出](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoa](#)中的。

## describe-locations

以下代码示例演示了如何使用 describe-locations。

### AWS CLI

列出 AWS Direct Connect 合作伙伴和地点

以下describe-locations命令列出当前区域的 AWS Direct Connect 合作伙伴和地点：

```
aws directconnect describe-locations
```

输出：

```
{
  "locations": [
    {
```

```

        "locationName": "NAP do Brasil, Barueri, Sao Paulo",
        "locationCode": "TNDB"
    },
    {
        "locationName": "Tivit - Site Transamerica (Sao Paulo)",
        "locationCode": "TIVIT"
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLocations](#)中的。

## describe-tags

以下代码示例演示了如何使用 describe-tags。

### AWS CLI

描述您的 Di AWS rect Connect 资源的标签

以下命令描述了连接的标签dxcon-abcabc12。

命令:

```
aws directconnect describe-tags --resource-arns arn:aws:directconnect:us-east-1:123456789012:dxcon/dxcon-abcabc12
```

输出:

```

{
  "resourceTags": [
    {
      "resourceArn": "arn:aws:directconnect:us-east-1:123456789012:dxcon/dxcon-abcabc12",
      "tags": [
        {
          "value": "VAConnection",
          "key": "Name"
        }
      ]
    }
  ]
}

```

```
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTags](#)中的。

## describe-virtual-gateways

以下代码示例演示了如何使用 describe-virtual-gateways。

### AWS CLI

列出虚拟专用网关

以下describe-virtual-gateways命令列出了您的 AWS 账户拥有的虚拟专用网关：

```
aws directconnect describe-virtual-gateways
```

输出：

```
{
  "virtualGateways": [
    {
      "virtualGatewayId": "vgw-aba37db6",
      "virtualGatewayState": "available"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVirtualGateways](#)中的。

## describe-virtual-interfaces

以下代码示例演示了如何使用 describe-virtual-interfaces。

### AWS CLI

列出所有虚拟接口

以下describe-virtual-interfaces命令列出了与您的 AWS 账户关联的所有虚拟接口的相关信息：



```
aws directconnect describe-virtual-interfaces --connection-id dxcon-ffjrnx17
```

输出：

```
{
  "virtualInterfaces": [
    {
      "virtualInterfaceState": "down",
      "asn": 65000,
      "vlan": 101,
      "customerAddress": "192.168.1.2/30",
      "ownerAccount": "123456789012",
      "connectionId": "dxcon-ffjrnx17",
      "virtualGatewayId": "vgw-aba37db6",
      "virtualInterfaceId": "dxvif-ffhkh74f",
      "authKey": "asdf34example",
      "routeFilterPrefixes": [],
      "location": "TIVIT",
      "customerRouterConfig": "<?xml version=\"1.0\" encoding=
UTF-8\"?>\n<logical_connection id=\"dxvif-ffhkh74f\">\n  <vlan>101</
vlan>\n  <customer_address>192.168.1.2/30</customer_address>\n
  <amazon_address>192.168.1.1/30</amazon_address>\n  <bgp_asn>65000</bgp_asn>\n
  <bgp_auth_key>asdf34example</bgp_auth_key>\n  <amazon_bgp_asn>7224</amazon_bgp_asn>
\n  <connection_type>private</connection_type>\n</logical_connection>\n",
      "amazonAddress": "192.168.1.1/30",
      "virtualInterfaceType": "private",
      "virtualInterfaceName": "PrivateVirtualInterface"
    },
    {
      "virtualInterfaceState": "verifying",
      "asn": 65000,
      "vlan": 2000,
      "customerAddress": "203.0.113.2/30",
      "ownerAccount": "123456789012",
      "connectionId": "dxcon-ffjrnx17",
      "virtualGatewayId": "",
      "virtualInterfaceId": "dxvif-fgh0hcrk",
      "authKey": "asdf34example",
      "routeFilterPrefixes": [
        {
          "cidr": "203.0.113.4/30"
        }
      ]
    }
  ]
}
```

```

        "cidr": "203.0.113.0/30"
      }
    ],
    "location": "TIVIT",
    "customerRouterConfig": "<?xml version=\"1.0\" encoding=
\"UTF-8\"?>\n<logical_connection id=\"dxvif-fgh0hcrk\">\n  <vlan>2000</
vlan>\n  <customer_address>203.0.113.2/30</customer_address>\n
  <amazon_address>203.0.113.1/30</amazon_address>\n  <bgp_asn>65000</bgp_asn>\n
  <bgp_auth_key>asdf34example</bgp_auth_key>\n  <amazon_bgp_asn>7224</amazon_bgp_asn>
\n  <connection_type>public</connection_type>\n</logical_connection>\n",
    "amazonAddress": "203.0.113.1/30",
    "virtualInterfaceType": "public",
    "virtualInterfaceName": "PublicVirtualInterface"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVirtualInterfaces](#)中的。

## disassociate-connection-from-lag

以下代码示例演示了如何使用 disassociate-connection-from-lag。

### AWS CLI

取消连接与 LAG 的关联

以下示例取消指定连接与指定 LAG 的关联。

命令:

```
aws directconnect disassociate-connection-from-lag --lag-id dxlag-fhccu14t --
connection-id dxcon-fg9607vm
```

输出:

```
{
  "ownerAccount": "123456789012",
  "connectionId": "dxcon-fg9607vm",
  "connectionState": "requested",
  "bandwidth": "1Gbps",
  "location": "EqDC2",
}
```

```
"connectionName": "Con2ForLag",  
"region": "us-east-1"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateConnectionFromLag](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

向 Di AWS rect Connect 资源添加标签

以下命令向连接添加一个密钥为Name、值VAConnection为的标签dxcon-abcabc12。如果命令成功，则不返回任何输出。

命令:

```
aws directconnect tag-resource --resource-arn arn:aws:directconnect:us-  
east-1:123456789012:dxcon/dxcon-abcabc12 --tags "key=Name,value=VAConnection"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从 Di AWS rect Connect 资源中移除标签

以下命令Name从连接中删除带有密钥的标签dxcon-abcabc12。如果命令成功，则不返回任何输出。

命令:

```
aws directconnect untag-resource --resource-arn arn:aws:directconnect:us-  
east-1:123456789012:dxcon/dxcon-abcabc12 --tag-keys Name
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-direct-connect-gateway-association

以下代码示例演示了如何使用 `update-direct-connect-gateway-association`。

### AWS CLI

更新 Direct Connect 网关关联的指定属性

以下 `update-direct-connect-gateway-association` 示例将指定的 CIDR 块添加到 Direct Connect 网关关联。

```
aws directconnect update-direct-connect-gateway-association \  
  --association-id 820a6e4f-5374-4004-8317-3f64bEXAMPLE \  
  --add-allowed-prefixes-to-direct-connect-gateway cidr=192.168.2.0/30
```

输出：

```
{  
  "directConnectGatewayAssociation": {  
    "directConnectGatewayId": "11460968-4ac1-4fd3-bdb2-00599EXAMPLE",  
    "directConnectGatewayOwnerAccount": "111122223333",  
    "associationState": "updating",  
    "associatedGateway": {  
      "id": "tgw-02f776b1a7EXAMPLE",  
      "type": "transitGateway",  
      "ownerAccount": "111122223333",  
      "region": "us-east-1"  
    },  
    "associationId": "820a6e4f-5374-4004-8317-3f64bEXAMPLE",  
    "allowedPrefixesToDirectConnectGateway": [  
      {  
        "cidr": "192.168.2.0/30"  
      },  
      {  
        "cidr": "192.168.1.0/30"  
      }  
    ]  
  }  
}
```

有关更多信息，请参阅 [Direct Connect 用户指南中的使用 Direct Connect 网关](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateDirectConnectGatewayAssociation](#) 中的。

## update-lag

以下代码示例演示了如何使用 update-lag。

### AWS CLI

#### 更新 LAG

以下示例更改了指定 LAG 的名称。

命令:

```
aws directconnect update-lag --lag-id dxlag-ffjhj9lx --lag-name 2ConnLag
```

输出:

```
{
  "awsDevice": "CSVA1-23u8tlpaz8iks",
  "numberOfConnections": 2,
  "lagState": "down",
  "ownerAccount": "123456789012",
  "lagName": "2ConnLag",
  "connections": [
    {
      "ownerAccount": "123456789012",
      "connectionId": "dxcon-fflqyj95",
      "lagId": "dxlag-ffjhj9lx",
      "connectionState": "requested",
      "bandwidth": "1Gbps",
      "location": "CSVA1",
      "connectionName": "Requested Connection 2 for Lag dxlag-ffjhj9lx",
      "region": "us-east-1"
    },
    {
      "ownerAccount": "123456789012",
      "connectionId": "dxcon-ffqr6x5q",
      "lagId": "dxlag-ffjhj9lx",
      "connectionState": "requested",
```

```

        "bandwidth": "1Gbps",
        "location": "CSVA1",
        "connectionName": "Requested Connection 1 for Lag dxlag-ffjhj91x",
        "region": "us-east-1"
    }
],
"lagId": "dxlag-ffjhj91x",
"minimumLinks": 0,
"connectionsBandwidth": "1Gbps",
"region": "us-east-1",
"location": "CSVA1"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLag](#)中的。

## update-virtual-interface-attributes

以下代码示例演示了如何使用 `update-virtual-interface-attributes`。

### AWS CLI

#### 更新虚拟接口的 MTU

以下 `update-virtual-interface-attributes` 示例更新指定虚拟接口的 MTU。

```

aws directconnect update-virtual-interface-attributes \
  --virtual-interface-id dxvif-fEXAMPLE \
  --mtu 1500

```

输出：

```

{
  "ownerAccount": "1111222233333",
  "virtualInterfaceId": "dxvif-fEXAMPLE",
  "location": "loc1",
  "connectionId": "dxlag-fEXAMPLE",
  "virtualInterfaceType": "transit",
  "virtualInterfaceName": "example transit virtual interface",
  "vlan": 125,
  "asn": 650001,
  "amazonSideAsn": 64512,
  "authKey": "0xzxgA9YoW9h58u8SEXAMPLE",

```

```

    "amazonAddress": "169.254.248.1/30",
    "customerAddress": "169.254.248.2/30",
    "addressFamily": "ipv4",
    "virtualInterfaceState": "down",
    "customerRouterConfig": "<?xml version=\"1.0\" encoding=\"UTF-8\"?
>\n<logical_connection id=\"dxvif-fEXAMPLE\">\n  <vlan>125</vlan>
\n  <customer_address>169.254.248.2/30</customer_address>\n
  <amazon_address>169.254.248.1/30</amazon_address>\n  <bgp_asn>650001</bgp_asn>\n
  <bgp_auth_key>0xzxcgA9YoW9h58u8SEXAMPLE</bgp_auth_key>\n  <amazon_bgp_asn>64512</
amazon_bgp_asn>\n  <connection_type>transit</connection_type>\n</logical_connection>
\n",
    "mtu": 1500,
    "jumboFrameCapable": true,
    "virtualGatewayId": "",
    "directConnectGatewayId": "879b76a1-403d-4700-8b53-4a56ed85436e",
    "routeFilterPrefixes": [],
    "bgpPeers": [
      {
        "bgpPeerId": "dxpeer-fEXAMPLE",
        "asn": 650001,
        "authKey": "0xzxcgA9YoW9h58u8SEXAMPLE",
        "addressFamily": "ipv4",
        "amazonAddress": "169.254.248.1/30",
        "customerAddress": "169.254.248.2/30",
        "bgpPeerState": "available",
        "bgpStatus": "down",
        "awsDeviceV2": "loc1-26wz6vEXAMPLE"
      }
    ],
    "region": "sa-east-1",
    "awsDeviceV2": "loc1-26wz6vEXAMPLE",
    "tags": []
  }
}

```

有关更多信息，请参阅 [Di AWS Direct Connect 用户指南中的为私有虚拟接口或传输虚拟接口设置网络 MTU](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateVirtualInterfaceAttributes](#) 中的。

## AWS Directory Service 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS Directory Service。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **describe-directories**

以下代码示例演示了如何使用 `describe-directories`。

AWS CLI

获取有关您的目录的详细信息

以下 `describe-directories` 示例显示有关指定目录的详细信息。

```
aws ds describe-directories \
  --directory-id d-a1b2c3d4e5
```

输出：

```
{
  "DirectoryDescriptions": [
    {
      "DirectoryId": "d-a1b2c3d4e5",
      "Name": "mydirectory.example.com",
      "ShortName": "mydirectory",
      "Size": "Small",
      "Edition": "Standard",
      "Alias": "d-a1b2c3d4e5",
      "AccessUrl": "d-a1b2c3d4e5.awsapps.com",
      "Stage": "Active",
      "ShareStatus": "Shared",
      "ShareMethod": "HANDSHAKE",
      "ShareNotes": "These are my share notes",
```



```

    "LaunchTime": "2019-07-08T15:33:46.327000-07:00",
    "StageLastUpdatedDateTime": "2019-07-08T15:59:12.307000-07:00",
    "Type": "SharedMicrosoftAD",
    "SsoEnabled": false,
    "DesiredNumberOfDomainControllers": 0,
    "OwnerDirectoryDescription": {
      "DirectoryId": "d-b2c3d4e5f6",
      "AccountId": "123456789111",
      "DnsIpAddrs": [
        "203.113.0.248",
        "203.113.0.253"
      ],
      "VpcSettings": {
        "VpcId": "vpc-a1b2c3d4",
        "SubnetIds": [
          "subnet-a1b2c3d4",
          "subnet-d4c3b2a1"
        ],
        "AvailabilityZones": [
          "us-west-2a",
          "us-west-2c"
        ]
      }
    }
  }
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDirectories](#)中的。

## describe-trusts

以下代码示例演示了如何使用 describe-trusts。

### AWS CLI

获取有关您的信任关系的详细信息

以下describe-trusts示例显示有关指定目录的信任关系的详细信息。

```
aws ds describe-trusts \
  --directory-id d-a1b2c3d4e5
```

输出：

```
{
  "Trusts": [
    {
      "DirectoryId": "d-a1b2c3d4e5",
      "TrustId": "t-9a8b7c6d5e",
      "RemoteDomainName": "other.example.com",
      "TrustType": "Forest",
      "TrustDirection": "Two-Way",
      "TrustState": "Verified",
      "CreatedDateTime": "2017-06-20T18:08:45.614000-07:00",
      "LastUpdatedDateTime": "2019-06-04T10:52:12.410000-07:00",
      "StateLastUpdatedDateTime": "2019-06-04T10:52:12.410000-07:00",
      "SelectiveAuth": "Disabled"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTrusts](#)中的。

## AWS DMS 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS DMS。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### add-tags-to-resource

以下代码示例演示了如何使用 `add-tags-to-resource`。

#### AWS CLI

为资源添加标签

以下 `add-tags-to-resource` 示例为复制实例添加标签。

```
aws dms add-tags-to-resource \  
  --resource-arn arn:aws:dms:us-east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE \  
  \  
  --tags Key=Environment,Value=PROD Key=Project,Value=dbMigration
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Database Migration Service 用户指南》中的[标记资源](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AddTagsToResource](#) 中的。

### create-endpoint

以下代码示例演示了如何使用 `create-endpoint`。

#### AWS CLI

创建终端节点

以下 `create-endpoint` 示例为 Amazon S3 源创建了一个终端节点。

```
aws dms create-endpoint \  
  --endpoint-type source \  
  --engine-name s3 \  
  --endpoint-identifier src-endpoint \  
  --s3-settings file://s3-settings.json
```

`s3-settings.json` 的内容：

```
{  
  "BucketName": "my-corp-data",
```

```

    "BucketFolder": "sourcedata",
    "ServiceAccessRoleArn": "arn:aws:iam::123456789012:role/my-s3-access-role"
  }

```

输出：

```

{
  "Endpoint": {
    "EndpointIdentifier": "src-endpoint",
    "EndpointType": "SOURCE",
    "EngineName": "s3",
    "EngineDisplayName": "Amazon S3",
    "ExtraConnectionAttributes": "bucketFolder=sourcedata;bucketName=my-corp-
data;compressionType=NONE;csvDelimiter=,;csvRowDelimiter=\\n;",
    "Status": "active",
    "EndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:GUVAFG34EECU0J6QVZ56DAHT3U",
    "SslMode": "none",
    "ServiceAccessRoleArn": "arn:aws:iam::123456789012:role/my-s3-access-role",
    "S3Settings": {
      "ServiceAccessRoleArn": "arn:aws:iam::123456789012:role/my-s3-access-
role",
      "CsvRowDelimiter": "\\n",
      "CsvDelimiter": ",",
      "BucketFolder": "sourcedata",
      "BucketName": "my-corp-data",
      "CompressionType": "NONE",
      "EnableStatistics": true
    }
  }
}

```

有关更多信息，请参阅 [AWS DMS atabase Migration Service 用户指南中的使用 DMS 端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateEndpoint](#) 中的。

## create-event-subscription

以下代码示例演示了如何使用 create-event-subscription。

AWS CLI

列出活动订阅

以下 `create-event-subscription` 示例创建了对 Amazon SNS 主题的事件订阅 () `my-sns-topic`。

```
aws dms create-event-subscription \  
  --subscription-name my-dms-events \  
  --sns-topic-arn arn:aws:sns:us-east-1:123456789012:my-sns-topic
```

输出：

```
{  
  "EventSubscription": {  
    "CustomerAwsId": "123456789012",  
    "CustSubscriptionId": "my-dms-events",  
    "SnsTopicArn": "arn:aws:sns:us-east-1:123456789012:my-sns-topic",  
    "Status": "creating",  
    "SubscriptionCreationTime": "2020-05-21 21:58:38.598",  
    "Enabled": true  
  }  
}
```

有关更多信息，请参阅 [AWS Database Migration Service 用户指南中的处理事件和通知](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `CreateEventSubscription`](#) 中的。

## create-replication-instance

以下代码示例演示了如何使用 `create-replication-instance`。

### AWS CLI

#### 创建复制实例

以下 `create-replication-instance` 示例创建了一个复制实例。

```
aws dms create-replication-instance \  
  --replication-instance-identifier my-repl-instance \  
  --replication-instance-class dms.t2.micro \  
  --allocated-storage 5
```

输出：

```
{
```

```
"ReplicationInstance": {
  "ReplicationInstanceIdentifier": "my-repl-instance",
  "ReplicationInstanceClass": "dms.t2.micro",
  "ReplicationInstanceStatus": "creating",
  "AllocatedStorage": 5,
  "VpcSecurityGroups": [
    {
      "VpcSecurityGroupId": "sg-f839b688",
      "Status": "active"
    }
  ],
  "ReplicationSubnetGroup": {
    "ReplicationSubnetGroupIdentifier": "default",
    "ReplicationSubnetGroupDescription": "default",
    "VpcId": "vpc-136a4c6a",
    "SubnetGroupStatus": "Complete",
    "Subnets": [
      {
        "SubnetIdentifier": "subnet-da327bf6",
        "SubnetAvailabilityZone": {
          "Name": "us-east-1a"
        },
        "SubnetStatus": "Active"
      },
      {
        "SubnetIdentifier": "subnet-42599426",
        "SubnetAvailabilityZone": {
          "Name": "us-east-1d"
        },
        "SubnetStatus": "Active"
      },
      {
        "SubnetIdentifier": "subnet-bac383e0",
        "SubnetAvailabilityZone": {
          "Name": "us-east-1c"
        },
        "SubnetStatus": "Active"
      },
      {
        "SubnetIdentifier": "subnet-6746046b",
        "SubnetAvailabilityZone": {
          "Name": "us-east-1f"
        },
        "SubnetStatus": "Active"
      }
    ]
  }
}
```

```
    },
    {
      "SubnetIdentifier": "subnet-d7c825e8",
      "SubnetAvailabilityZone": {
        "Name": "us-east-1e"
      },
      "SubnetStatus": "Active"
    },
    {
      "SubnetIdentifier": "subnet-cbfff283",
      "SubnetAvailabilityZone": {
        "Name": "us-east-1b"
      },
      "SubnetStatus": "Active"
    }
  ]
},
"PreferredMaintenanceWindow": "sat:12:35-sat:13:05",
"PendingModifiedValues": {},
"MultiAZ": false,
"EngineVersion": "3.3.2",
"AutoMinorVersionUpgrade": true,
"KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/f7bc0f8e-1a3a-4ace-9faa-
e8494fa3921a",
"ReplicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:ZK2VQBUWFDBAWHIXHAYG5G2PKY",
"PubliclyAccessible": true
}
}
```

有关更多信息，请参阅 [AWS Data AWS abase Migration Service 用户指南中的使用 DMS 复制实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateReplicationInstance](#) 中的。

## create-replication-subnet-group

以下代码示例演示了如何使用 create-replication-subnet-group。

### AWS CLI

#### 创建子网组

以下`create-replication-subnet-group`示例创建了一个由 3 个子网组成的组。

```
aws dms create-replication-subnet-group \  
  --replication-subnet-group-identifier my-subnet-group \  
  --replication-subnet-group-description "my subnet group" \  
  --subnet-ids subnet-da327bf6 subnet-bac383e0 subnet-d7c825e8
```

输出：

```
{  
  "ReplicationSubnetGroup": {  
    "ReplicationSubnetGroupIdentifier": "my-subnet-group",  
    "ReplicationSubnetGroupDescription": "my subnet group",  
    "VpcId": "vpc-136a4c6a",  
    "SubnetGroupStatus": "Complete",  
    "Subnets": [  
      {  
        "SubnetIdentifier": "subnet-da327bf6",  
        "SubnetAvailabilityZone": {  
          "Name": "us-east-1a"  
        },  
        "SubnetStatus": "Active"  
      },  
      {  
        "SubnetIdentifier": "subnet-bac383e0",  
        "SubnetAvailabilityZone": {  
          "Name": "us-east-1c"  
        },  
        "SubnetStatus": "Active"  
      },  
      {  
        "SubnetIdentifier": "subnet-d7c825e8",  
        "SubnetAvailabilityZone": {  
          "Name": "us-east-1e"  
        },  
        "SubnetStatus": "Active"  
      }  
    ]  
  }  
}
```

有关更多信息，请参阅《[Dat AWS abase Migration Service 用户指南](#)》中的[为复制实例设置网络](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateReplicationSubnetGroup](#)中的。

## create-replication-task

以下代码示例演示了如何使用 create-replication-task。

### AWS CLI

#### 创建复制任务

以下create-replication-task示例创建了一个复制任务。

```
aws dms create-replication-task \  
  --replication-task-identifier movedata \  
  --source-endpoint-arn arn:aws:dms:us-  
east-1:123456789012:endpoint:6GGI6YPWGWAYUVLKIB732KEVWA \  
  --target-endpoint-arn arn:aws:dms:us-  
east-1:123456789012:endpoint:E0M4SFKCZEYHZBFGAGZT3QEC5U \  
  --replication-instance-arn $RI_ARN \  
  --migration-type full-load \  
  --table-mappings file://table-mappings.json
```

table-mappings.json 的内容：

```
{  
  "rules": [  
    {  
      "rule-type": "selection",  
      "rule-id": "1",  
      "rule-name": "1",  
      "object-locator": {  
        "schema-name": "prodrep",  
        "table-name": "%"  
      },  
      "rule-action": "include",  
      "filters": []  
    }  
  ]  
}
```

输出：

```
{
  "ReplicationTask": {
    "ReplicationTaskIdentifier": "moveit2",
    "SourceEndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWGWAYUVLKIB732KEVWA",
    "TargetEndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:E0M4SFKCZEYHZBFGAGZT3QEC5U",
    "ReplicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE",
    "MigrationType": "full-load",
    "TableMappings": ...output omitted... ,
    "ReplicationTaskSettings": ...output omitted... ,
    "Status": "creating",
    "ReplicationTaskCreationDate": 1590524772.505,
    "ReplicationTaskArn": "arn:aws:dms:us-
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII"
  }
}
```

有关更多信息，请参阅 [AWS DMS AWS atabase Migration Service 用户指南中的处理 DMS 任务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateReplicationTask](#) 中的。

## delete-connection

以下代码示例演示了如何使用 delete-connection。

### AWS CLI

#### 删除连接

以下 delete-connection 示例取消终端节点与复制实例的关联。

```
aws dms delete-connection \
  --endpoint-arn arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWGWAYUVLKIB732KEVWA \
  --replication-instance-arn arn:aws:dms:us-
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE
```

输出：

```
{
  "Connection": {
```

```

    "ReplicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE",
    "EndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWGWAYUVLKIB732KEVWA",
    "Status": "deleting",
    "EndpointIdentifier": "src-database-1",
    "ReplicationInstanceIdentifier": "my-repl-instance"
  }
}

```

有关更多信息，请参阅《AWS 数据库迁移服务用户指南》中的 [https://docs.aws.amazon.com/dms/latest/userguide/CHAP\\_Endpoints.Creating.html](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Endpoints.Creating.html)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteConnection](#) 中的。

## delete-endpoint

以下代码示例演示了如何使用 delete-endpoint。

### AWS CLI

#### 删除终端节点

以下 delete-endpoint 示例删除终端节点。

```

aws dms delete-endpoint \
  --endpoint-arn arn:aws:dms:us-
east-1:123456789012:endpoint:0UJJVX04XZ4CYTSEG5XGMN2R3Y

```

输出：

```

{
  "Endpoint": {
    "EndpointIdentifier": "src-endpoint",
    "EndpointType": "SOURCE",
    "EngineName": "s3",
    "EngineDisplayName": "Amazon S3",
    "ExtraConnectionAttributes": "bucketFolder=sourcedata;bucketName=my-corp-
data;compressionType=NONE;csvDelimiter=,;csvRowDelimiter=\n;",
    "Status": "deleting",
    "EndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:0UJJVX04XZ4CYTSEG5XGMN2R3Y",
    "SslMode": "none",
  }
}

```

```

    "ServiceAccessRoleArn": "arn:aws:iam::123456789012:role/my-s3-access-role",
    "S3Settings": {
      "ServiceAccessRoleArn": "arn:aws:iam::123456789012:role/my-s3-access-
role",
      "CsvRowDelimiter": "\\n",
      "CsvDelimiter": ",",
      "BucketFolder": "sourcedata",
      "BucketName": "my-corp-data",
      "CompressionType": "NONE",
      "EnableStatistics": true
    }
  }
}

```

有关更多信息，请参阅 [AWS DMS atabase Migration Service 用户指南中的使用 DMS 端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteEndpoint](#) 中的。

## delete-event-subscription

以下代码示例演示了如何使用 delete-event-subscription。

### AWS CLI

#### 删除活动订阅

以下 delete-event-subscription 示例删除了对 Amazon SNS 主题的订阅。

```
aws dms delete-event-subscription \
  --subscription-name "my-dms-events"
```

输出：

```

{
  "EventSubscription": {
    "CustomerAwsId": "123456789012",
    "CustSubscriptionId": "my-dms-events",
    "SnsTopicArn": "arn:aws:sns:us-east-1:123456789012:my-sns-topic",
    "Status": "deleting",
    "SubscriptionCreationTime": "2020-05-21 21:58:38.598",
    "Enabled": true
  }
}

```

有关更多信息，请参阅 D AWS atabase Migration Service 用户指南中的[处理事件和通知](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteEventSubscription](#)中的。

## delete-replication-instance

以下代码示例演示了如何使用 delete-replication-instance。

### AWS CLI

#### 删除复制实例

以下 delete-replication-instance 示例删除复制实例。

```
aws dms delete-replication-instance \  
  --replication-instance-arn arn:aws:dms:us-  
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNU
```

输出：

```
{  
  "ReplicationInstance": {  
    "ReplicationInstanceIdentifier": "my-repl-instance",  
    "ReplicationInstanceClass": "dms.t2.micro",  
    "ReplicationInstanceStatus": "deleting",  
    "AllocatedStorage": 5,  
    "InstanceCreateTime": 1590011235.952,  
    "VpcSecurityGroups": [  
      {  
        "VpcSecurityGroupId": "sg-f839b688",  
        "Status": "active"  
      }  
    ],  
    "AvailabilityZone": "us-east-1e",  
    "ReplicationSubnetGroup": {  
      "ReplicationSubnetGroupIdentifier": "default",  
      "ReplicationSubnetGroupDescription": "default",  
      "VpcId": "vpc-136a4c6a",  
      "SubnetGroupStatus": "Complete",  
      "Subnets": [  
        {  
          "SubnetIdentifier": "subnet-da327bf6",  
          "SubnetAvailabilityZone": {
```

```
        "Name": "us-east-1a"
      },
      "SubnetStatus": "Active"
    },
    {
      "SubnetIdentifier": "subnet-42599426",
      "SubnetAvailabilityZone": {
        "Name": "us-east-1d"
      },
      "SubnetStatus": "Active"
    },
    {
      "SubnetIdentifier": "subnet-bac383e0",
      "SubnetAvailabilityZone": {
        "Name": "us-east-1c"
      },
      "SubnetStatus": "Active"
    },
    {
      "SubnetIdentifier": "subnet-6746046b",
      "SubnetAvailabilityZone": {
        "Name": "us-east-1f"
      },
      "SubnetStatus": "Active"
    },
    {
      "SubnetIdentifier": "subnet-d7c825e8",
      "SubnetAvailabilityZone": {
        "Name": "us-east-1e"
      },
      "SubnetStatus": "Active"
    },
    {
      "SubnetIdentifier": "subnet-cbfff283",
      "SubnetAvailabilityZone": {
        "Name": "us-east-1b"
      },
      "SubnetStatus": "Active"
    }
  ]
},
"PreferredMaintenanceWindow": "wed:11:42-wed:12:12",
"PendingModifiedValues": {},
"MultiAZ": true,
```

```
    "EngineVersion": "3.3.2",
    "AutoMinorVersionUpgrade": true,
    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/f7bc0f8e-1a3a-4ace-9faa-
e8494fa3921a",
    "ReplicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE",
    "ReplicationInstancePublicIpAddress": "54.225.120.92",
    "ReplicationInstancePrivateIpAddress": "172.31.30.121",
    "ReplicationInstancePublicIpAddresses": [
      "54.225.120.92",
      "3.230.18.248"
    ],
    "ReplicationInstancePrivateIpAddresses": [
      "172.31.30.121",
      "172.31.75.90"
    ],
    "PubliclyAccessible": true,
    "SecondaryAvailabilityZone": "us-east-1b"
  }
}
```

有关更多信息，请参阅 [AWS Data AWS abase Migration Service 用户指南中的使用 DMS 复制实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteReplicationInstance](#) 中的。

## delete-replication-subnet-group

以下代码示例演示了如何使用 delete-replication-subnet-group。

### AWS CLI

#### 删除子网组

以下 delete-replication-subnet-group 示例删除子网组。

```
aws dms delete-replication-subnet-group \
--replication-subnet-group-identifier my-subnet-group
```

#### 输出：

```
(none)
```

有关更多信息，请参阅《[Dat AWS abase Migration Service 用户指南](#)》中的[为复制实例设置网络](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteReplicationSubnetGroup](#)中的。

## delete-replication-task

以下代码示例演示了如何使用 delete-replication-task。

### AWS CLI

#### 删除复制任务

以下delete-replication-task示例删除了复制任务。

```
aws dms delete-replication-task \  
  --replication-task-arn arn:aws:dms:us-  
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII
```

输出：

```
{  
  "ReplicationTask": {  
    "ReplicationTaskIdentifier": "moveit2",  
    "SourceEndpointArn": "arn:aws:dms:us-  
east-1:123456789012:endpoint:6GGI6YPWYGAYUVLKIB732KEVWA",  
    "TargetEndpointArn": "arn:aws:dms:us-  
east-1:123456789012:endpoint:E0M4SFKCZEYHZBFGAGZT3QEC5U",  
    "ReplicationInstanceArn": "arn:aws:dms:us-  
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE",  
    "MigrationType": "full-load",  
    "TableMappings": "...output omitted...",  
    "ReplicationTaskSettings": "...output omitted...",  
    "Status": "deleting",  
    "StopReason": "Stop Reason FULL_LOAD_ONLY_FINISHED",  
    "ReplicationTaskCreationDate": 1590524772.505,  
    "ReplicationTaskStartDate": 1590789988.677,  
    "ReplicationTaskArn": "arn:aws:dms:us-  
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII"  
  }  
}
```

有关更多信息，请参阅 [AWS D AWS atabase Migration Service 用户指南](#)中的[处理 DMS 任务](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteReplicationTask](#)中的。

## describe-account-attributes

以下代码示例演示了如何使用 describe-account-attributes。

### AWS CLI

#### 描述账户属性

以下describe-account-attributes示例列出了您 AWS 账户的属性。

```
aws dms describe-account-attributes
```

输出：

```
{
  "AccountQuotas": [
    {
      "AccountQuotaName": "ReplicationInstances",
      "Used": 1,
      "Max": 20
    },
    {
      "AccountQuotaName": "AllocatedStorage",
      "Used": 5,
      "Max": 10000
    },
    ...remaining output omitted...
  ],
  "UniqueAccountIdentifier": "cqahfbfy5xee"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAccountAttributes](#)中的。

## describe-certificates

以下代码示例演示了如何使用 describe-certificates。

## AWS CLI

### 列出可用证书

以下describe-certificates示例列出了您 AWS 账户中的可用证书。

```
aws dms describe-certificates
```

输出：

```
{
  "Certificates": [
    {
      "CertificateIdentifier": "my-cert",
      "CertificateCreationDate": 1543259542.506,
      "CertificatePem": "-----BEGIN CERTIFICATE-----
\nMIID9DCCAtygAwIBAgIBQjANBgkqhkiG9w0BAQ ...U"
      ... remaining output omitted ...
    }
  ]
}
```

有关更多信息，请参阅 [D AWS atabase Migration Service 用户指南中的使用 SSL](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCertificates](#)中的。

## describe-connections

以下代码示例演示了如何使用 describe-connections。

## AWS CLI

### 描述连接

以下describe-connections示例列出了您在复制实例和终端节点之间测试的连接。

```
aws dms describe-connections
```

输出：

```
{
```

```

    "Connections": [
      {
        "Status": "successful",
        "ReplicationInstanceIdentifier": "test",
        "EndpointArn": "arn:aws:dms:us-east-1:123456789012:endpoint:ZW5UAN6P4E77EC7YWHK4RZZ3BE",
        "EndpointIdentifier": "testsrc1",
        "ReplicationInstanceArn": "arn:aws:dms:us-east-1:123456789012:rep:6UTDJGB0US3VI3SUWA66XFJCJQ"
      }
    ]
  }
}

```

有关更多信息，请参阅 D AWS atabase Migration Service 用户指南中的[创建源端点和目标端点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeConnections](#)中的。

## describe-endpoint-types

以下代码示例演示了如何使用 describe-endpoint-types。

### AWS CLI

列出可用的端点类型

以下describe-endpoint-types示例列出了可用的 MySQL 端点类型。

```

aws dms describe-endpoint-types \
  --filters "Name=engine-name,Values=mysql"

```

输出：

```

{
  "SupportedEndpointTypes": [
    {
      "EngineName": "mysql",
      "SupportsCDC": true,
      "EndpointType": "source",
      "EngineDisplayName": "MySQL"
    },
    {
      "EngineName": "mysql",
      "SupportsCDC": true,

```

```
        "EndpointType": "target",
        "EngineDisplayName": "MySQL"
    }
]
}
```

有关更多信息，请参阅 AWS Databases AWS e Migration Service 用户指南中的使用 DMS 端点`\_\_`。  
< [https://docs.aws.amazon.com/dms/latest/userguide/CHAP\\_Endpoints.html](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Endpoints.html)>

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEndpointTypes](#) 中的。

## describe-endpoints

以下代码示例演示了如何使用 describe-endpoints。

### AWS CLI

#### 描述端点

以下 describe-endpoints 示例列出了您 AWS 账户中的终端节点。

```
aws dms describe-endpoints
```

#### 输出：

```
{
  "Endpoints": [
    {
      "Username": "dms",
      "Status": "active",
      "EndpointArn": "arn:aws:dms:us-east-1:123456789012:endpoint:SF2W0FLWYWKVE0HID2EKLP3SJI",
      "ServerName": "ec2-52-32-48-61.us-west-2.compute.amazonaws.com",
      "EndpointType": "SOURCE",
      "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/94d5c4e7-4e4c-44be-b58a-c8da7adf57cd",
      "DatabaseName": "test",
      "EngineName": "mysql",
      "EndpointIdentifier": "pri100",
      "Port": 8193
    },
    {
      "Username": "admin",
```

```
    "Status": "active",
    "EndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:TJJZCIH3CJ24TJRU4VC32WEWFR",
    "ServerName": "test.example.com",
    "EndpointType": "SOURCE",
    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/2431021b-1cf2-
a2d4-77b2-59a9e4bce323",
    "DatabaseName": "EMPL",
    "EngineName": "oracle",
    "EndpointIdentifier": "test",
    "Port": 1521
  }
]
}
```

有关更多信息，请参阅 [AWS DMS atabase Migration Service 用户指南中的使用 DMS 端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEndpoints](#) 中的。

## describe-event-categories

以下代码示例演示了如何使用 `describe-event-categories`。

### AWS CLI

#### 描述事件类别

以下 `describe-event-categories` 示例列出了可用的事件类别。

```
aws dms describe-event-categories
```

输出：

```
{
  "EventCategoryGroupList": [
    {
      "SourceType": "replication-instance",
      "EventCategories": [
        "low storage",
        "configuration change",
        "maintenance",
        "deletion",
        "creation",

```

```

        "failover",
        "failure"
    ]
},
{
    "SourceType": "replication-task",
    "EventCategories": [
        "configuration change",
        "state change",
        "deletion",
        "creation",
        "failure"
    ]
}
]
}

```

有关更多信息，请参阅 D AWS atabase Migration Service 用户指南 [中的处理事件和通知](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEventCategories](#) 中的。

## describe-event-subscriptions

以下代码示例演示了如何使用 describe-event-subscriptions。

### AWS CLI

#### 描述活动订阅

以下 describe-event-subscriptions 示例列出了对 Amazon SNS 主题的事件订阅。

```
aws dms describe-event-subscriptions
```

输出：

```

{
  "EventSubscriptionsList": [
    {
      "CustomerAwsId": "123456789012",
      "CustSubscriptionId": "my-dms-events",
      "SnsTopicArn": "arn:aws:sns:us-east-1:123456789012:my-sns-topic",
      "Status": "deleting",
      "SubscriptionCreationTime": "2020-05-21 22:28:51.924",
    }
  ]
}

```

```

        "Enabled": true
      }
    ]
  }

```

有关更多信息，请参阅 D AWS atabase Migration Service 用户指南 [中的处理事件和通知](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEventSubscriptions](#) 中的。

## describe-events

以下代码示例演示了如何使用 describe-events。

### AWS CLI

列出 DMS 事件

以下 describe-events 示例列出了源自复制实例的事件。

```

aws dms describe-events \
  --source-type "replication-instance"

```

输出：

```

{
  "Events": [
    {
      "SourceIdentifier": "my-repl-instance",
      "SourceType": "replication-instance",
      "Message": "Replication application shutdown",
      "EventCategories": [],
      "Date": 1590771645.776
    }
  ]
}

```

有关更多信息，请参阅 D AWS atabase Migration Service 用户指南 [中的处理事件和通知](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEvents](#) 中的。

## describe-orderable-replication-instances

以下代码示例演示了如何使用 describe-orderable-replication-instances。

## AWS CLI

### 描述可订购的复制实例

以下describe-orderable-replication-instances示例列出了您可以订购的复制实例类型。

```
aws dms describe-orderable-replication-instances
```

输出：

```
{
  "OrderableReplicationInstances": [
    {
      "EngineVersion": "3.3.2",
      "ReplicationInstanceClass": "dms.c4.2xlarge",
      "StorageType": "gp2",
      "MinAllocatedStorage": 5,
      "MaxAllocatedStorage": 6144,
      "DefaultAllocatedStorage": 100,
      "IncludedAllocatedStorage": 100,
      "AvailabilityZones": [
        "us-east-1a",
        "us-east-1b",
        "us-east-1c",
        "us-east-1d",
        "us-east-1e",
        "us-east-1f"
      ]
    },
    {
      "EngineVersion": "3.3.2",
      "ReplicationInstanceClass": "dms.c4.4xlarge",
      "StorageType": "gp2",
      "MinAllocatedStorage": 5,
      "MaxAllocatedStorage": 6144,
      "DefaultAllocatedStorage": 100,
      "IncludedAllocatedStorage": 100,
      "AvailabilityZones": [
        "us-east-1a",
        "us-east-1b",
        "us-east-1c",
        "us-east-1d",

```



```

        "us-east-1e",
        "us-east-1f"
    ]
},
...remaining output omitted...
}

```

有关更多信息，请参阅 [AWS Data AWS abase Migration Service 用户指南中的使用 DMS 复制实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeOrderableReplicationInstances](#) 中的。

## describe-refresh-schemas-status

以下代码示例演示了如何使用 `describe-refresh-schemas-status`。

### AWS CLI

列出终端节点的刷新状态

以下 `describe-refresh-schemas-status` 示例返回先前刷新请求的状态。

```

aws dms describe-refresh-schemas-status \
  --endpoint-arn arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA

```

输出：

```

{
  "RefreshSchemasStatus": {
    "EndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA",
    "ReplicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE",
    "Status": "successful",
    "LastRefreshDate": 1590786544.605
  }
}

```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeRefreshSchemasStatus](#) 中的。

## describe-replication-instances

以下代码示例演示了如何使用 `describe-replication-instances`。

### AWS CLI

#### 描述复制实例

以下 `describe-replication-instances` 示例列出了您 AWS 账户中的复制实例。

```
aws dms describe-replication-instances
```

输出：

```
{
  "ReplicationInstances": [
    {
      "ReplicationInstanceIdentifier": "my-repl-instance",
      "ReplicationInstanceClass": "dms.t2.micro",
      "ReplicationInstanceStatus": "available",
      "AllocatedStorage": 5,
      "InstanceCreateTime": 1590011235.952,
      "VpcSecurityGroups": [
        {
          "VpcSecurityGroupId": "sg-f839b688",
          "Status": "active"
        }
      ],
      "AvailabilityZone": "us-east-1e",
      "ReplicationSubnetGroup": {
        "ReplicationSubnetGroupIdentifier": "default",
        "ReplicationSubnetGroupDescription": "default",
        "VpcId": "vpc-136a4c6a",
        "SubnetGroupStatus": "Complete",
        "Subnets": [
          {
            "SubnetIdentifier": "subnet-da327bf6",
            "SubnetAvailabilityZone": {
              "Name": "us-east-1a"
            }
          },
          {
            "SubnetIdentifier": "subnet-42599426",
```

```
        "SubnetAvailabilityZone": {
            "Name": "us-east-1d"
        },
        "SubnetStatus": "Active"
    },
    {
        "SubnetIdentifier": "subnet-bac383e0",
        "SubnetAvailabilityZone": {
            "Name": "us-east-1c"
        },
        "SubnetStatus": "Active"
    },
    {
        "SubnetIdentifier": "subnet-6746046b",
        "SubnetAvailabilityZone": {
            "Name": "us-east-1f"
        },
        "SubnetStatus": "Active"
    },
    {
        "SubnetIdentifier": "subnet-d7c825e8",
        "SubnetAvailabilityZone": {
            "Name": "us-east-1e"
        },
        "SubnetStatus": "Active"
    },
    {
        "SubnetIdentifier": "subnet-cbfff283",
        "SubnetAvailabilityZone": {
            "Name": "us-east-1b"
        },
        "SubnetStatus": "Active"
    }
]
},
"PreferredMaintenanceWindow": "wed:11:42-wed:12:12",
"PendingModifiedValues": {
    "MultiAZ": true
},
"MultiAZ": false,
"EngineVersion": "3.3.2",
"AutoMinorVersionUpgrade": true,
"KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
f7bc0f8e-1a3a-4ace-9faa-e8494fa3921a",
```

```

    "ReplicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE",
    "ReplicationInstancePublicIpAddress": "3.230.18.248",
    "ReplicationInstancePrivateIpAddress": "172.31.75.90",
    "ReplicationInstancePublicIpAddresses": [
        "3.230.18.248"
    ],
    "ReplicationInstancePrivateIpAddresses": [
        "172.31.75.90"
    ],
    "PubliclyAccessible": true,
    "FreeUntil": 1590194829.267
  }
]
}

```

有关更多信息，请参阅 [AWS Data AWS abase Migration Service 用户指南中的使用 DMS 复制实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeReplicationInstances](#) 中的。

## describe-replication-subnet-groups

以下代码示例演示了如何使用 describe-replication-subnet-groups。

### AWS CLI

显示可用的子网组

以下 describe-replication-subnet-groups 示例列出了可用的子网组。

```

aws dms describe-replication-subnet-groups \
  --filter "Name=replication-subnet-group-id,Values=my-subnet-group"

```

输出：

```

{
  "ReplicationSubnetGroups": [
    {
      "ReplicationSubnetGroupIdentifier": "my-subnet-group",
      "ReplicationSubnetGroupDescription": "my subnet group",
      "VpcId": "vpc-136a4c6a",

```

```
    "SubnetGroupStatus": "Complete",
    "Subnets": [
      {
        "SubnetIdentifier": "subnet-da327bf6",
        "SubnetAvailabilityZone": {
          "Name": "us-east-1a"
        },
        "SubnetStatus": "Active"
      },
      {
        "SubnetIdentifier": "subnet-bac383e0",
        "SubnetAvailabilityZone": {
          "Name": "us-east-1c"
        },
        "SubnetStatus": "Active"
      },
      {
        "SubnetIdentifier": "subnet-d7c825e8",
        "SubnetAvailabilityZone": {
          "Name": "us-east-1e"
        },
        "SubnetStatus": "Active"
      }
    ]
  }
]
```

有关更多信息，请参阅《Dat AWS abase Migration Service 用户指南》中的[为复制实例设置网络](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeReplicationSubnetGroups](#)中的。

## describe-replication-task-assessment-results

以下代码示例演示了如何使用 describe-replication-task-assessment-results。

### AWS CLI

列出复制任务评估的结果

以下describe-replication-task-assessment-results示例列出了先前任务评估的结果。

```
aws dms describe-replication-task-assessment-results
```

输出：

```
{
  "ReplicationTaskAssessmentResults": [
    {
      "ReplicationTaskIdentifier": "moveit2",
      "ReplicationTaskArn": "arn:aws:dms:us-
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII",
      "ReplicationTaskLastAssessmentDate": 1590790230.0,
      "AssessmentStatus": "No issues found",
      "AssessmentResultsFile": "moveit2/2020-05-29-22-10"
    }
  ]
}
```

有关更多信息，请参阅《D AWS atabase Migration Service 用户指南》中的[创建任务评估报告](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeReplicationTaskAssessmentResults](#)中的。

## describe-replication-tasks

以下代码示例演示了如何使用 describe-replication-tasks。

AWS CLI

描述复制任务

以下describe-replication-tasks示例描述了当前的复制任务。

```
aws dms describe-replication-tasks
```

输出：

```
{
  "ReplicationTasks": [
    {
      "ReplicationTaskIdentifier": "moveit2",
      "SourceEndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA",
      "TargetEndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:E0M4SFKCZEYHZBFGAGZT3QEC5U",
```

```

    "ReplicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE",
    "MigrationType": "full-load",
    "TableMappings": ...output omitted... ,
    "ReplicationTaskSettings": ...output omitted... ,
    "Status": "stopped",
    "StopReason": "Stop Reason FULL_LOAD_ONLY_FINISHED",
    "ReplicationTaskCreationDate": 1590524772.505,
    "ReplicationTaskStartDate": 1590619805.212,
    "ReplicationTaskArn": "arn:aws:dms:us-
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII",
    "ReplicationTaskStats": {
      "FullLoadProgressPercent": 100,
      "ElapsedTimeMillis": 0,
      "TablesLoaded": 0,
      "TablesLoading": 0,
      "TablesQueued": 0,
      "TablesErrored": 0,
      "FreshStartDate": 1590619811.528,
      "StartDate": 1590619811.528,
      "StopDate": 1590619842.068
    }
  }
]
}

```

有关更多信息，请参阅 [AWS DMS atabase Migration Service 用户指南中的处理 DMS 任务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeReplicationTasks](#) 中的。

## describe-schemas

以下代码示例演示了如何使用 describe-schemas。

### AWS CLI

#### 描述数据库架构

以下 describe-schemas 示例列出了终端节点上的可用表。

```

aws dms describe-schemas \
  --endpoint-arn "arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA"

```

输出：

```
{
  "Schemas": [
    "prodrep"
  ]
}
```

有关更多信息，请参阅《AWS Database Migration Service 用户指南》中的[主题标题](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSchemas](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

AWS CLI

列出资源的标签

以下`list-tags-for-resource`示例列出了复制实例的标签。

```
aws dms list-tags-for-resource \
  --resource-arn arn:aws:dms:us-east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE
```

输出：

```
{
  "TagList": [
    {
      "Key": "Project",
      "Value": "dbMigration"
    },
    {
      "Key": "Environment",
      "Value": "PROD"
    }
  ]
}
```

有关更多信息，请参阅《AWS Database Migration Service 用户指南》中的[标记资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。



## modify-endpoint

以下代码示例演示了如何使用 modify-endpoint。

### AWS CLI

#### 修改端点

以下 modify-endpoint 示例向终端节点添加了额外的连接属性。

```
aws dms modify-endpoint \  
  --endpoint-arn "arn:aws:dms:us-  
east-1:123456789012:endpoint:GUVAFG34EECU0J6QVZ56DAHT3U" \  
  --extra-connection-attributes "compressionType=GZIP"
```

输出：

```
{  
  "Endpoint": {  
    "EndpointIdentifier": "src-endpoint",  
    "EndpointType": "SOURCE",  
    "EngineName": "s3",  
    "EngineDisplayName": "Amazon S3",  
    "ExtraConnectionAttributes":  
"compressionType=GZIP;csvDelimiter=,;csvRowDelimiter=\\n;",  
    "Status": "active",  
    "EndpointArn": "arn:aws:dms:us-  
east-1:123456789012:endpoint:GUVAFG34EECU0J6QVZ56DAHT3U",  
    "SslMode": "none",  
    "ServiceAccessRoleArn": "arn:aws:iam::123456789012:role/my-s3-access-role",  
    "S3Settings": {  
      "ServiceAccessRoleArn": "arn:aws:iam::123456789012:role/my-s3-access-  
role",  
      "CsvRowDelimiter": "\\n",  
      "CsvDelimiter": ",",  
      "BucketFolder": "",  
      "BucketName": "",  
      "CompressionType": "GZIP",  
      "EnableStatistics": true  
    }  
  }  
}
```

有关更多信息，请参阅 AWS Databases AWS e Migration Service 用户指南中的使用 DMS 端点`\_\_`。  
< [https://docs.aws.amazon.com/dms/latest/userguide/CHAP\\_Endpoints.html](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Endpoints.html)>

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyEndpoint](#)中的。

## modify-event-subscription

以下代码示例演示了如何使用 modify-event-subscription。

### AWS CLI

#### 修改活动订阅

以下modify-event-subscription示例更改了事件订阅的来源类型。

```
aws dms modify-event-subscription \
  --subscription-name "my-dms-events" \
  --source-type replication-task
```

输出：

```
{
  "EventSubscription": {
    "CustomerAwsId": "123456789012",
    "CustSubscriptionId": "my-dms-events",
    "SnsTopicArn": "arn:aws:sns:us-east-1:123456789012:my-sns-topic",
    "Status": "modifying",
    "SubscriptionCreationTime": "2020-05-29 17:04:40.262",
    "SourceType": "replication-task",
    "Enabled": true
  }
}
```

有关更多信息，请参阅 Databases AWS Migration Service 用户指南中的[处理事件和通知](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyEventSubscription](#)中的。

## modify-replication-instance

以下代码示例演示了如何使用 modify-replication-instance。

## AWS CLI

### 修改复制实例

以下`modify-replication-instance`示例修改复制实例，使其使用多可用区部署。

```
aws dms modify-replication-instance \  
  --replication-instance-arn arn:aws:dms:us-  
east-1:123456789012:rep:T3OM70UB5NM2LCVZF7JPGJRNUE \  
  --multi-az
```

输出：

```
{  
  "ReplicationInstance": {  
    "ReplicationInstanceIdentifier": "my-repl-instance",  
    "ReplicationInstanceClass": "dms.t2.micro",  
    "ReplicationInstanceStatus": "available",  
    "AllocatedStorage": 5,  
    "InstanceCreateTime": 1590011235.952,  
  
    ...output omitted...  
  
    "PendingModifiedValues": {  
      "MultiAZ": true  
    },  
    "MultiAZ": false,  
    "EngineVersion": "3.3.2",  
    "AutoMinorVersionUpgrade": true,  
    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/f7bc0f8e-1a3a-4ace-9faa-  
e8494fa3921a",  
  
    ...output omitted...  
  }  
}
```

有关更多信息，请参阅 [AWS Data Migration Service 用户指南中的使用 DMS 复制实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyReplicationInstance](#) 中的。

## modify-replication-subnet-group

以下代码示例演示了如何使用 modify-replication-subnet-group。

### AWS CLI

#### 修改子网组

以下modify-replication-subnet-group示例更改了与子网组关联的子网列表。

```
aws dms modify-replication-subnet-group \  
  --replication-subnet-group-identifier my-subnet-group \  
  --subnet-id subnet-da327bf6 subnet-bac383e0
```

输出：

```
{  
  "ReplicationSubnetGroup": {  
    "ReplicationSubnetGroupIdentifier": "my-subnet-group",  
    "ReplicationSubnetGroupDescription": "my subnet group",  
    "VpcId": "vpc-136a4c6a",  
    "SubnetGroupStatus": "Complete",  
    "Subnets": [  
      {  
        "SubnetIdentifier": "subnet-da327bf6",  
        "SubnetAvailabilityZone": {  
          "Name": "us-east-1a"  
        },  
        "SubnetStatus": "Active"  
      },  
      {  
        "SubnetIdentifier": "subnet-bac383e0",  
        "SubnetAvailabilityZone": {  
          "Name": "us-east-1c"  
        },  
        "SubnetStatus": "Active"  
      }  
    ]  
  }  
}
```

有关更多信息，请参阅《Dat AWS abase Migration Service 用户指南》中的[为复制实例设置网络](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyReplicationSubnetGroup](#)中的。

## modify-replication-task

以下代码示例演示了如何使用 `modify-replication-task`。

### AWS CLI

#### 修改复制任务

以下 `modify-replication-task` 示例更改了任务的表映射。

```
aws dms modify-replication-task \  
  --replication-task-arn "arn:aws:dms:us-  
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII" \  
  --table-mappings file://table-mappings.json
```

`table-mappings.json` 的内容：

```
{  
  "rules": [  
    {  
      "rule-type": "selection",  
      "rule-id": "1",  
      "rule-name": "1",  
      "object-locator": {  
        "schema-name": "prodrep",  
        "table-name": "ACCT_%"  
      },  
      "rule-action": "include",  
      "filters": []  
    }  
  ]  
}
```

输出：

```
{  
  "ReplicationTask": {  
    "ReplicationTaskIdentifier": "moveit2",  
    "SourceEndpointArn": "arn:aws:dms:us-  
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA",  
    "TargetEndpointArn": "arn:aws:dms:us-  
east-1:123456789012:endpoint:E0M4SFKCZEYHZBFGAGZT3QEC5U",
```

```

    "ReplicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE",
    "MigrationType": "full-load",
    "TableMappings": ...output omitted...,
    "ReplicationTaskSettings": ...output omitted...,
    "Status": "modifying",
    "StopReason": "Stop Reason FULL_LOAD_ONLY_FINISHED",
    "ReplicationTaskCreationDate": 1590524772.505,
    "ReplicationTaskStartDate": 1590789424.653,
    "ReplicationTaskArn": "arn:aws:dms:us-
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII"
  }
}

```

有关更多信息，请参阅 [AWS DMS atabase Migration Service 用户指南中的处理 DMS 任务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyReplicationTask](#) 中的。

## reboot-replication-instance

以下代码示例演示了如何使用 `reboot-replication-instance`。

### AWS CLI

#### 重启复制实例

以下 `reboot-replication-instance` 示例重启复制实例。

```

aws dms reboot-replication-instance \
  --replication-instance-arn arn:aws:dms:us-
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE

```

输出：

```

{
  "ReplicationInstance": {
    "ReplicationInstanceIdentifier": "my-repl-instance",
    "ReplicationInstanceClass": "dms.t2.micro",
    "ReplicationInstanceStatus": "rebooting",
    "AllocatedStorage": 5,
    "InstanceCreateTime": 1590011235.952,
    ... output omitted ...
  }
}

```

```
}
```

有关更多信息，请参阅 [AWS Data Migration Service 用户指南中的使用 DMS 复制实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RebootReplicationInstance](#) 中的。

## refresh-schemas

以下代码示例演示了如何使用 refresh-schemas。

### AWS CLI

#### 刷新数据库架构

以下 refresh-schemas 示例请求 AWS DMS 刷新终端节点上的架构列表。

```
aws dms refresh-schemas \
  --replication-instance-arn arn:aws:dms:us-
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE \
  --endpoint-arn "arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA"
```

输出：

```
{
  "RefreshSchemasStatus": {
    "EndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA",
    "ReplicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE",
    "Status": "refreshing",
    "LastRefreshDate": 1590019949.103
  }
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RefreshSchemas](#) 中的。

## reload-tables

以下代码示例演示了如何使用 reload-tables。

## AWS CLI

刷新终端节点上可用表的列表

以下`reload-tables`示例重新加载终端节点上的可用表列表。

```
aws dms reload-tables \  
  --replication-task-arn "arn:aws:dms:us-  
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII" \  
  --tables-to-reload "SchemaName=prodrep,TableName=ACCT_BAL"
```

输出：

```
{  
  "ReplicationTaskArn": "arn:aws:dms:us-  
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReloadTables](#)中的。

## `remove-tags-from-resource`

以下代码示例演示了如何使用 `remove-tags-from-resource`。

## AWS CLI

从复制实例中移除标签

以下`remove-tags-from-resource`示例从复制实例中删除标签。

```
aws dms remove-tags-from-resource \  
  --resource-arn arn:aws:dms:us-east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE  
 \  
  --tag-keys Environment Project
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Database Migration Service 用户指南》中的[标记资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveTagsFromResource](#)中的。



## start-replication-task-assessment

以下代码示例演示了如何使用 start-replication-task-assessment。

### AWS CLI

#### 开始任务评估

以下start-replication-task-assessment示例启动复制任务评估。

```
aws dms start-replication-task-assessment \  
  --replication-task-arn arn:aws:dms:us-  
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII
```

输出：

```
{  
  "ReplicationTask": {  
    "ReplicationTaskIdentifier": "moveit2",  
    "SourceEndpointArn": "arn:aws:dms:us-  
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA",  
    "TargetEndpointArn": "arn:aws:dms:us-  
east-1:123456789012:endpoint:E0M4SFKCZEYHZBFGAGZT3QEC5U",  
    "ReplicationInstanceArn": "arn:aws:dms:us-  
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE",  
    "MigrationType": "full-load",  
    "TableMappings": "...output omitted...",  
    "ReplicationTaskSettings": "...output omitted...",  
    "Status": "testing",  
    "StopReason": "Stop Reason FULL_LOAD_ONLY_FINISHED",  
    "ReplicationTaskCreationDate": 1590524772.505,  
    "ReplicationTaskStartDate": 1590789988.677,  
    "ReplicationTaskArn": "arn:aws:dms:us-  
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII"  
  }  
}
```

有关更多信息，请参阅《D AWS atabase Migration Service 用户指南》中的[创建任务评估报告](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartReplicationTaskAssessment](#)中的。

## start-replication-task

以下代码示例演示了如何使用 start-replication-task。

### AWS CLI

#### 启动复制任务

以下command-name示例列出了您 AWS 账户中可用的微件。

```
aws dms start-replication-task \  
  --replication-task-arn arn:aws:dms:us-  
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII \  
  --start-replication-task-type reload-target
```

输出：

```
{  
  "ReplicationTask": {  
    "ReplicationTaskIdentifier": "moveit2",  
    "SourceEndpointArn": "arn:aws:dms:us-  
east-1:123456789012:endpoint:6GGI6YPWGWAYUVLKIB732KEVWA",  
    "TargetEndpointArn": "arn:aws:dms:us-  
east-1:123456789012:endpoint:E0M4SFKCZEYHZBFGAGZT3QEC5U",  
    "ReplicationInstanceArn": "arn:aws:dms:us-  
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE",  
    "MigrationType": "full-load",  
    "TableMappings": "...output omitted... ",  
    "ReplicationTaskSettings": "...output omitted... ",  
    "Status": "starting",  
    "ReplicationTaskCreationDate": 1590524772.505,  
    "ReplicationTaskStartDate": 1590619805.212,  
    "ReplicationTaskArn": "arn:aws:dms:us-  
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII"  
  }  
}
```

有关更多信息，请参阅 [AWS DMS atabase Migration Service 用户指南中的处理 DMS 任务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartReplicationTask](#)中的。

## stop-replication-task

以下代码示例演示了如何使用 stop-replication-task。

### AWS CLI

#### 停止任务

以下stop-replication-task示例停止任务。

```
aws dms stop-replication-task \  
  --replication-task-arn arn:aws:dms:us-  
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII
```

输出：

```
{  
  "ReplicationTask": {  
    "ReplicationTaskIdentifier": "moveit2",  
    "SourceEndpointArn": "arn:aws:dms:us-  
east-1:123456789012:endpoint:6GGI6YPWYGAYUVLKIB732KEVWA",  
    "TargetEndpointArn": "arn:aws:dms:us-  
east-1:123456789012:endpoint:E0M4SFKCZEYHZBFGAGZT3QEC5U",  
    "ReplicationInstanceArn": "arn:aws:dms:us-  
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE",  
    "MigrationType": "full-load",  
    "TableMappings": "...output omitted...",  
    "ReplicationTaskSettings": "...output omitted...",  
    "Status": "stopping",  
    "ReplicationTaskCreationDate": 1590524772.505,  
    "ReplicationTaskStartDate": 1590789424.653,  
    "ReplicationTaskArn": "arn:aws:dms:us-  
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII"  
  }  
}
```

有关更多信息，请参阅 [AWS DMS atabase Migration Service 用户指南中的处理 DMS 任务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopReplicationTask](#)中的。

## test-connection

以下代码示例演示了如何使用 test-connection。

## AWS CLI

### 测试与端点的连接

以下test-connection示例测试是否可以从复制实例访问终端节点。

```
aws dms test-connection \  
  --replication-instance-arn arn:aws:dms:us-  
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE \  
  --endpoint-arn arn:aws:dms:us-  
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA
```

输出：

```
{  
  "Connection": {  
    "ReplicationInstanceArn": "arn:aws:dms:us-  
east-1:123456789012:rep:T30M70UB5NM2LCVZF7JPGJRNUE",  
    "EndpointArn": "arn:aws:dms:us-  
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA",  
    "Status": "testing",  
    "EndpointIdentifier": "src-database-1",  
    "ReplicationInstanceIdentifier": "my-repl-instance"  
  }  
}
```

有关更多信息，请参阅 D AWS atabase Migration Service 用户指南中的[创建源端点和目标端点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TestConnection](#)中的。

## 使用 Amazon DocumentDB 示例 AWS CLI

以下代码示例向您展示了如何在 Amazon DocumentDB 中 AWS Command Line Interface 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### **add-tags-to-resource**

以下代码示例演示了如何使用 `add-tags-to-resource`。

#### AWS CLI

向指定资源添加一个或多个标签

以下 `add-tags-to-resource` 示例向添加了三个标签 `sample-cluster`。一个标签 (`CropB`) 有密钥名称但没有值。

```
aws docdb add-tags-to-resource \  
  --resource-name arn:aws:rds:us-west-2:123456789012:cluster:sample-cluster \  
  --tags Key="CropA",Value="Apple" Key="CropB" Key="CropC",Value="Corn"
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 DocumentDB 开发者指南中的为 [亚马逊 DocumentDB 资源添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AddTagsToResource](#) 中的。

### **apply-pending-maintenance-action**

以下代码示例演示了如何使用 `apply-pending-maintenance-action`。

#### AWS CLI

在下一个维护时段内执行待处理的维护操作

以下 `apply-pending-maintenance-action` 示例导致在下一个计划维护时段内执行所有系统更新操作。

```
aws docdb apply-pending-maintenance-action \  
  --resource-identifier arn:aws:rds:us-west-2:123456789012:cluster:sample-cluster \  
  --apply-action system-update \  
  --opt-in-type next-maintenance
```

此命令不生成任何输出。

有关更多信息，请参阅[亚马逊 DocumentDB 开发者指南中的应用亚马逊 DocumentDB 更新](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ApplyPendingMaintenanceAction](#)中的。

## copy-db-cluster-parameter-group

以下代码示例演示了如何使用 copy-db-cluster-parameter-group。

### AWS CLI

复制现有的数据库集群参数组

以下copy-db-cluster-parameter-group示例创建custom-docdb3-6名为的参数组的副本custom-docdb3-6-copy。制作副本时，它会向新的参数组添加标签。

```
aws docdb copy-db-cluster-parameter-group \  
  --source-db-cluster-parameter-group-identifier custom-docdb3-6 \  
  --target-db-cluster-parameter-group-identifier custom-docdb3-6-copy \  
  --target-db-cluster-parameter-group-description "Copy of custom-docdb3-6" \  
  --tags Key="CopyNumber",Value="1" Key="Modifiable",Value="Yes"
```

输出：

```
{  
  "DBClusterParameterGroup": {  
    "DBParameterGroupFamily": "docdb3.6",  
    "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:12345678901:cluster-  
pg:custom-docdb3-6-copy",  
    "DBClusterParameterGroupName": "custom-docdb3-6-copy",  
    "Description": "Copy of custom-docdb3-6"  
  }  
}
```

有关更多信息，请参阅[亚马逊 DocumentDB 开发者指南中的复制亚马逊 DocumentDB 集群参数组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CopyDbClusterParameterGroup](#)中的。

## copy-db-cluster-snapshot

以下代码示例演示了如何使用 copy-db-cluster-snapshot。

## AWS CLI

### 创建快照副本

以下 `copy-db-cluster-snapshot` 示例将创建 `sample-cluster-snapshot` 的副本，名为 `sample-cluster-snapshot-copy`。副本包含原始标签的所有标签以及带有密钥名称的新标签 `CopyNumber`。

```
aws docdb copy-db-cluster-snapshot \  
  --source-db-cluster-snapshot-identifier sample-cluster-snapshot \  
  --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy \  
  --copy-tags \  
  --tags Key="CopyNumber",Value="1"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon DocumentDB 开发者[指南中的复制集群快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CopyDbClusterSnapshot](#) 中的。

## **create-db-cluster-parameter-group**

以下代码示例演示了如何使用 `create-db-cluster-parameter-group`。

## AWS CLI

### 创建 Amazon DocumentDB 集群参数组

以下 `create-db-cluster-parameter-group` 示例 `sample-parameter-group` 使用 `docdb3.6` 系列创建数据库集群参数组。

```
aws docdb create-db-cluster-parameter-group \  
  --db-cluster-parameter-group-name sample-parameter-group \  
  --db-parameter-group-family docdb3.6 \  
  --description "Sample parameter group based on docdb3.6"
```

输出：

```
{  
  "DBClusterParameterGroup": {  
    "Description": "Sample parameter group based on docdb3.6",  
    "DBParameterGroupFamily": "docdb3.6",
```

```

    "DBClusterParameterGroupArn": "arn:aws:rds:us-west-2:123456789012:cluster-
pg:sample-parameter-group",
    "DBClusterParameterGroupName": "sample-parameter-group"
  }
}

```

有关更多信息，请参阅亚马逊 DocumentDB 开发者指南中的[创建亚马逊 DocumentDB 集群参数组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateDbClusterParameterGroup](#)中的。

## create-db-cluster-snapshot

以下代码示例演示了如何使用 create-db-cluster-snapshot。

### AWS CLI

创建手动的 Amazon DocumentDB 集群快照

以下 create-db-cluster-snapshot 示例创建名为的 Amazon 数据库集群快照 sample-cluster-snapshot。

```

aws docdb create-db-cluster-snapshot \
  --db-cluster-identifier sample-cluster \
  --db-cluster-snapshot-identifier sample-cluster-snapshot

```

输出：

```

{
  "DBClusterSnapshot": {
    "MasterUsername": "master-user",
    "SnapshotCreateTime": "2019-03-18T18:27:14.794Z",
    "AvailabilityZones": [
      "us-west-2a",
      "us-west-2b",
      "us-west-2c",
      "us-west-2d",
      "us-west-2e",
      "us-west-2f"
    ],
    "SnapshotType": "manual",
    "DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:sample-cluster-snapshot",
  }
}

```



```
    "EngineVersion": "3.6.0",
    "PercentProgress": 0,
    "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
    "Engine": "docdb",
    "DBClusterIdentifier": "sample-cluster",
    "Status": "creating",
    "ClusterCreateTime": "2019-03-15T20:29:58.836Z",
    "Port": 0,
    "StorageEncrypted": false,
    "VpcId": "vpc-91280df6"
  }
}
```

有关更多信息，请参阅 Amazon DocumentDB 开发人员指南中的[创建手动集群快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateDbClusterSnapshot](#)中的。

## create-db-cluster

以下代码示例演示了如何使用 create-db-cluster。

### AWS CLI

#### 创建 Amazon DocumentDB 集群

以下 create-db-cluster 示例创建了一个名为 sample-cluster 的 Amazon DocumentDB 集群，其首选维护时段为星期日 20:30 到 11:00 之间。

```
aws docdb create-db-cluster \
  --db-cluster-identifier sample-cluster \
  --engine docdb \
  --master-username master-user \
  --master-user-password password \
  --preferred-maintenance-window Sun:20:30-Sun:21:00
```

输出：

```
{
  "DBCluster": {
    "DBClusterParameterGroup": "default.docdb3.6",
    "AssociatedRoles": [],
    "DBSubnetGroup": "default",
    "ClusterCreateTime": "2019-03-18T18:06:34.616Z",
```

```

    "Status": "creating",
    "Port": 27017,
    "PreferredMaintenanceWindow": "sun:20:30-sun:21:00",
    "HostedZoneId": "ZNKXH85TT8WVW",
    "DBClusterMembers": [],
    "Engine": "docdb",
    "DBClusterIdentifier": "sample-cluster",
    "PreferredBackupWindow": "10:12-10:42",
    "AvailabilityZones": [
      "us-west-2d",
      "us-west-2f",
      "us-west-2e"
    ],
    "MasterUsername": "master-user",
    "BackupRetentionPeriod": 1,
    "ReaderEndpoint": "sample-cluster.cluster-ro-corcjozrlsfc.us-
west-2.docdb.amazonaws.com",
    "VpcSecurityGroups": [
      {
        "VpcSecurityGroupId": "sg-77186e0d",
        "Status": "active"
      }
    ],
    "StorageEncrypted": false,
    "DBClusterArn": "arn:aws:rds:us-west-2:123456789012:cluster:sample-cluster",
    "DbClusterResourceId": "cluster-L3R4YRSBUYDP4GLMTJ2WF5GH5Q",
    "MultiAZ": false,
    "Endpoint": "sample-cluster.cluster-corcjozrlsfc.us-
west-2.docdb.amazonaws.com",
    "EngineVersion": "3.6.0"
  }
}

```

有关更多信息，请参阅亚马逊 Document [tDB 开发者指南中的创建亚马逊 DocumentDB 集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDbCluster](#) 中的。

## create-db-instance

以下代码示例演示了如何使用 create-db-instance。

### AWS CLI

#### 创建 Amazon DocumentDB 集群实例

以下create-db-instance示例代码在 Amazon DocumentDB 集群sample-cluster-instance-2中创建实例。sample-cluster

```
aws docdb create-db-instance \  
  --db-cluster-identifier sample-cluster \  
  --db-instance-class db.r4.xlarge \  
  --db-instance-identifier sample-cluster-instance-2 \  
  --engine docdb
```

输出：

```
{  
  "DBInstance": {  
    "DBInstanceStatus": "creating",  
    "PendingModifiedValues": {  
      "PendingCloudwatchLogsExports": {  
        "LogTypesToEnable": [  
          "audit"  
        ]  
      }  
    },  
    "PubliclyAccessible": false,  
    "PreferredBackupWindow": "00:00-00:30",  
    "PromotionTier": 1,  
    "EngineVersion": "3.6.0",  
    "BackupRetentionPeriod": 3,  
    "DBInstanceIdentifier": "sample-cluster-instance-2",  
    "PreferredMaintenanceWindow": "tue:10:28-tue:10:58",  
    "StorageEncrypted": false,  
    "Engine": "docdb",  
    "DBClusterIdentifier": "sample-cluster",  
    "DBSubnetGroup": {  
      "Subnets": [  
        {  
          "SubnetAvailabilityZone": {  
            "Name": "us-west-2a"  
          },  
          "SubnetStatus": "Active",  
          "SubnetIdentifier": "subnet-4e26d263"  
        },  
        {  
          "SubnetAvailabilityZone": {  
            "Name": "us-west-2c"  
          }  
        }  
      ]  
    }  
  }  
}
```

```
    },
    "SubnetStatus": "Active",
    "SubnetIdentifier": "subnet-afc329f4"
  },
  {
    "SubnetAvailabilityZone": {
      "Name": "us-west-2d"
    },
    "SubnetStatus": "Active",
    "SubnetIdentifier": "subnet-53ab3636"
  },
  {
    "SubnetAvailabilityZone": {
      "Name": "us-west-2b"
    },
    "SubnetStatus": "Active",
    "SubnetIdentifier": "subnet-991cb8d0"
  }
],
"DBSubnetGroupDescription": "default",
"SubnetGroupStatus": "Complete",
"VpcId": "vpc-91280df6",
"DBSubnetGroupName": "default"
},
"DBInstanceClass": "db.r4.xlarge",
"VpcSecurityGroups": [
  {
    "Status": "active",
    "VpcSecurityGroupId": "sg-77186e0d"
  }
],
"DBInstanceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster-
instance-2",
"DbiResourceId": "db-XEKJLEMGRV5ZKCARUVA4H03ITE"
}
}
```

有关更多信息，请参阅[亚马逊 DocumentDB 开发者指南中的向集群添加亚马逊文档数据库实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDbInstance](#) 中的。

## create-db-subnet-group

以下代码示例演示了如何使用 create-db-subnet-group。

### AWS CLI

#### 创建 Amazon DocumentDB 子网组

以下 create-db-subnet-group 示例创建了一个名为的 Amazon DocumentDB 子网组。sample-subnet-group

```
aws docdb create-db-subnet-group \  
  --db-subnet-group-description "a sample subnet group" \  
  --db-subnet-group-name sample-subnet-group \  
  --subnet-ids "subnet-29ab1025" "subnet-991cb8d0" "subnet-53ab3636"
```

输出：

```
{  
  "DBSubnetGroup": {  
    "SubnetGroupStatus": "Complete",  
    "DBSubnetGroupName": "sample-subnet-group",  
    "DBSubnetGroupDescription": "a sample subnet group",  
    "VpcId": "vpc-91280df6",  
    "DBSubnetGroupArn": "arn:aws:rds:us-west-2:123456789012:subgrp:sample-subnet-group",  
    "Subnets": [  
      {  
        "SubnetStatus": "Active",  
        "SubnetIdentifier": "subnet-53ab3636",  
        "SubnetAvailabilityZone": {  
          "Name": "us-west-2d"  
        }  
      },  
      {  
        "SubnetStatus": "Active",  
        "SubnetIdentifier": "subnet-991cb8d0",  
        "SubnetAvailabilityZone": {  
          "Name": "us-west-2b"  
        }  
      },  
      {  
        "SubnetStatus": "Active",
```

```
        "SubnetIdentifier": "subnet-29ab1025",
        "SubnetAvailabilityZone": {
            "Name": "us-west-2c"
        }
    ]
}
```

有关更多信息，请参阅亚马逊 DocumentDB [开发者指南中的创建亚马逊 DocumentDB 子网组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDbSubnetGroup](#)中的。

## delete-db-cluster-parameter-group

以下代码示例演示了如何使用 delete-db-cluster-parameter-group。

### AWS CLI

删除 Amazon DocumentDB 集群参数组

以下delete-db-cluster-parameter-group示例删除了 Amazon DocumentDB 参数组。sample-parameter-group

```
aws docdb delete-db-cluster-parameter-group \
  --db-cluster-parameter-group-name sample-parameter-group
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 DocumentDB [开发者指南中的删除亚马逊 DocumentDB 集群参数组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDbClusterParameterGroup](#)中的。

## delete-db-cluster-snapshot

以下代码示例演示了如何使用 delete-db-cluster-snapshot。

### AWS CLI

删除 Amazon DocumentDB 集群快照

以下delete-db-cluster-snapshot示例删除了 Amazon DocumentDB 集群快照。sample-cluster-snapshot

```
aws docdb delete-db-cluster-snapshot \  
  --db-cluster-snapshot-identifier sample-cluster-snapshot
```

输出：

```
{  
  "DBClusterSnapshot": {  
    "DBClusterIdentifier": "sample-cluster",  
    "AvailabilityZones": [  
      "us-west-2a",  
      "us-west-2b",  
      "us-west-2c",  
      "us-west-2d"  
    ],  
    "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",  
    "VpcId": "vpc-91280df6",  
    "DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-snapshot:sample-cluster-snapshot",  
    "EngineVersion": "3.6.0",  
    "Engine": "docdb",  
    "SnapshotCreateTime": "2019-03-18T18:27:14.794Z",  
    "Status": "available",  
    "MasterUsername": "master-user",  
    "ClusterCreateTime": "2019-03-15T20:29:58.836Z",  
    "PercentProgress": 100,  
    "StorageEncrypted": false,  
    "SnapshotType": "manual",  
    "Port": 0  
  }  
}
```

有关更多信息，请参阅 Amazon DocumentDB 开发者[指南中的删除集群快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDbClusterSnapshot](#)中的。

## delete-db-cluster

以下代码示例演示了如何使用 delete-db-cluster。

## AWS CLI

### 删除亚马逊 DocumentDB 集群

以下`delete-db-cluster`示例删除了亚马逊文档数据库集群。`sample-cluster`在删除集群之前，不会对其进行备份。注意：必须先删除与集群关联的所有实例，然后才能将其删除。

```
aws docdb delete-db-cluster \  
  --db-cluster-identifier sample-cluster \  
  --skip-final-snapshot
```

输出：

```
{  
  "DBCluster": {  
    "DBClusterIdentifier": "sample-cluster",  
    "DBSubnetGroup": "default",  
    "EngineVersion": "3.6.0",  
    "Engine": "docdb",  
    "LatestRestorableTime": "2019-03-18T18:07:24.610Z",  
    "PreferredMaintenanceWindow": "sun:20:30-sun:21:00",  
    "StorageEncrypted": false,  
    "EarliestRestorableTime": "2019-03-18T18:07:24.610Z",  
    "Port": 27017,  
    "VpcSecurityGroups": [  
      {  
        "Status": "active",  
        "VpcSecurityGroupId": "sg-77186e0d"  
      }  
    ],  
    "MultiAZ": false,  
    "MasterUsername": "master-user",  
    "DBClusterArn": "arn:aws:rds:us-west-2:123456789012:cluster:sample-cluster",  
    "Status": "available",  
    "PreferredBackupWindow": "10:12-10:42",  
    "ReaderEndpoint": "sample-cluster.cluster-ro-corcjozrlsfc.us-west-2.docdb.amazonaws.com",  
    "AvailabilityZones": [  
      "us-west-2c",  
      "us-west-2b",  
      "us-west-2a"  
    ],  
  },  
}
```



```

    "Endpoint": "sample-cluster.cluster-corcjozrlsfc.us-
west-2.docdb.amazonaws.com",
    "DbClusterResourceId": "cluster-L3R4YRSBUYDP4GLMTJ2WF5GH5Q",
    "ClusterCreateTime": "2019-03-18T18:06:34.616Z",
    "AssociatedRoles": [],
    "DBClusterParameterGroup": "default.docdb3.6",
    "HostedZoneId": "ZNKXH85TT8WVW",
    "BackupRetentionPeriod": 1,
    "DBClusterMembers": []
  }
}

```

有关更多信息，请参阅亚马逊 Document [tDB 开发者指南中的删除亚马逊 DocumentDB 集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteDbCluster](#) 中的。

## delete-db-instance

以下代码示例演示了如何使用 delete-db-instance。

### AWS CLI

#### 删除亚马逊文档数据库实例

以下 delete-db-instance 示例删除了亚马逊文档数据库实例。sample-cluster-instance-2

```
aws docdb delete-db-instance \
  --db-instance-identifier sample-cluster-instance-2
```

输出：

```

{
  "DBInstance": {
    "DBSubnetGroup": {
      "Subnets": [
        {
          "SubnetAvailabilityZone": {
            "Name": "us-west-2a"
          },
          "SubnetStatus": "Active",
          "SubnetIdentifier": "subnet-4e26d263"
        }
      ],
    }
  }
}

```

```
    {
      "SubnetAvailabilityZone": {
        "Name": "us-west-2c"
      },
      "SubnetStatus": "Active",
      "SubnetIdentifier": "subnet-afc329f4"
    },
    {
      "SubnetAvailabilityZone": {
        "Name": "us-west-2d"
      },
      "SubnetStatus": "Active",
      "SubnetIdentifier": "subnet-53ab3636"
    },
    {
      "SubnetAvailabilityZone": {
        "Name": "us-west-2b"
      },
      "SubnetStatus": "Active",
      "SubnetIdentifier": "subnet-991cb8d0"
    }
  ],
  "DBSubnetGroupName": "default",
  "DBSubnetGroupDescription": "default",
  "VpcId": "vpc-91280df6",
  "SubnetGroupStatus": "Complete"
},
"PreferredBackupWindow": "00:00-00:30",
"InstanceCreateTime": "2019-03-18T18:37:33.709Z",
"DBInstanceClass": "db.r4.xlarge",
"DbiResourceId": "db-XEKJLEMGRV5ZKCARUVA4H03ITE",
"BackupRetentionPeriod": 3,
"Engine": "docdb",
"VpcSecurityGroups": [
  {
    "Status": "active",
    "VpcSecurityGroupId": "sg-77186e0d"
  }
],
"AutoMinorVersionUpgrade": true,
"PromotionTier": 1,
"EngineVersion": "3.6.0",
"Endpoint": {
```

```
        "Address": "sample-cluster-instance-2.corcjozrlsfc.us-
west-2.docdb.amazonaws.com",
        "HostedZoneId": "ZNKXH85TT8WVW",
        "Port": 27017
    },
    "DBInstanceIdentifier": "sample-cluster-instance-2",
    "PreferredMaintenanceWindow": "tue:10:28-tue:10:58",
    "EnabledCloudwatchLogsExports": [
        "audit"
    ],
    "PendingModifiedValues": {},
    "DBInstanceStatus": "deleting",
    "PubliclyAccessible": false,
    "DBInstanceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster-
instance-2",
    "DBClusterIdentifier": "sample-cluster",
    "AvailabilityZone": "us-west-2c",
    "StorageEncrypted": false
}
}
```

有关更多信息，请参阅[亚马逊 DocumentDB 开发者指南中的删除亚马逊文档数据库实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDbInstance](#)中的。

## delete-db-subnet-group

以下代码示例演示了如何使用 delete-db-subnet-group。

### AWS CLI

#### 删除 Amazon DocumentDB 子网组

以下delete-db-subnet-group示例删除了 Amazon DocumentDB 子网组。sample-subnet-group

```
aws docdb delete-db-subnet-group \
    --db-subnet-group-name sample-subnet-group
```

此命令不生成任何输出。

有关更多信息，请参阅[亚马逊 DocumentDB 开发者指南中的删除亚马逊 DocumentDB 子网组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDbSubnetGroup](#)中的。

## describe-db-cluster-parameter-groups

以下代码示例演示了如何使用 describe-db-cluster-parameter-groups。

### AWS CLI

查看一个或多个 Amazon DocumentDB 集群参数组的详细信息

以下describe-db-cluster-parameter-groups示例显示了 Amazon DocumentDB 集群参数组的详细信息。custom3-6-param-grp

```
aws docdb describe-db-cluster-parameter-groups \  
  --db-cluster-parameter-group-name custom3-6-param-grp
```

输出：

```
{  
  "DBClusterParameterGroups": [  
    {  
      "DBParameterGroupFamily": "docdb3.6",  
      "DBClusterParameterGroupArn": "arn:aws:rds:us-  
east-1:123456789012:cluster-pg:custom3-6-param-grp",  
      "Description": "Custom docdb3.6 parameter group",  
      "DBClusterParameterGroupName": "custom3-6-param-grp"  
    }  
  ]  
}
```

有关更多信息，请参阅亚马逊 D [ocumentDB 开发者指南中的查看亚马逊 Docum entDB 集群参数组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbClusterParameterGroups](#)中的。

## describe-db-cluster-parameters

以下代码示例演示了如何使用 describe-db-cluster-parameters。

### AWS CLI

查看 Amazon DocumentDB 集群参数组的详细参数列表。

以下describe-db-cluster-parameters示例列出了亚马逊 DocumentDB 参数组 custom3-6-param-grp 的参数。

```
aws docdb describe-db-cluster-parameters \  
  --db-cluster-parameter-group-name custom3-6-param-grp
```

输出：

```
{  
  "Parameters": [  
    {  
      "DataType": "string",  
      "ParameterName": "audit_logs",  
      "IsModifiable": true,  
      "ApplyMethod": "pending-reboot",  
      "Source": "system",  
      "ApplyType": "dynamic",  
      "AllowedValues": "enabled,disabled",  
      "Description": "Enables auditing on cluster.",  
      "ParameterValue": "disabled"  
    },  
    {  
      "DataType": "string",  
      "ParameterName": "tls",  
      "IsModifiable": true,  
      "ApplyMethod": "pending-reboot",  
      "Source": "system",  
      "ApplyType": "static",  
      "AllowedValues": "disabled,enabled",  
      "Description": "Config to enable/disable TLS",  
      "ParameterValue": "enabled"  
    },  
    {  
      "DataType": "string",  
      "ParameterName": "ttl_monitor",  
      "IsModifiable": true,  
      "ApplyMethod": "pending-reboot",  
      "Source": "user",  
      "ApplyType": "dynamic",  
      "AllowedValues": "disabled,enabled",  
      "Description": "Enables TTL Monitoring",  
      "ParameterValue": "enabled"  
    }  
  ]  
}
```

```
]
}
```

有关更多信息，请参阅亚马逊 DocumentDB [DB 开发者指南中的查看亚马逊 DocumentDB 集群参数](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbClusterParameters](#)中的。

## describe-db-cluster-snapshot-attributes

以下代码示例演示了如何使用 describe-db-cluster-snapshot-attributes。

### AWS CLI

列出 Amazon DocumentDB 快照属性名称和值

以下describe-db-cluster-snapshot-attributes示例列出了 Amazon DocumentDB 快照的属性名称和值。sample-cluster-snapshot

```
aws docdb describe-db-cluster-snapshot-attributes \
  --db-cluster-snapshot-identifier sample-cluster-snapshot
```

输出：

```
{
  "DBClusterSnapshotAttributesResult": {
    "DBClusterSnapshotAttributes": [
      {
        "AttributeName": "restore",
        "AttributeValues": []
      }
    ],
    "DBClusterSnapshotIdentifier": "sample-cluster-snapshot"
  }
}
```

有关更多信息，请参阅亚马逊 DocumentDB [DB 开发者指南ClusterSnapshotAttributes中的 Described B](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbClusterSnapshotAttributes](#)中的。

## describe-db-cluster-snapshots

以下代码示例演示了如何使用 describe-db-cluster-snapshots。

### AWS CLI

描述亚马逊 DocumentDB 快照

以下 describe-db-cluster-snapshots 示例显示了 Amazon DocumentDB 快照的详细信息。sample-cluster-snapshot

```
aws docdb describe-db-cluster-snapshots \  
  --db-cluster-snapshot-identifier sample-cluster-snapshot
```

输出：

```
{  
  "DBClusterSnapshots": [  
    {  
      "AvailabilityZones": [  
        "us-west-2a",  
        "us-west-2b",  
        "us-west-2c",  
        "us-west-2d"  
      ],  
      "Status": "available",  
      "DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-  
snapshot:sample-cluster-snapshot",  
      "SnapshotCreateTime": "2019-03-15T20:41:26.515Z",  
      "SnapshotType": "manual",  
      "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",  
      "DBClusterIdentifier": "sample-cluster",  
      "MasterUsername": "master-user",  
      "StorageEncrypted": false,  
      "VpcId": "vpc-91280df6",  
      "EngineVersion": "3.6.0",  
      "PercentProgress": 100,  
      "Port": 0,  
      "Engine": "docdb",  
      "ClusterCreateTime": "2019-03-15T20:29:58.836Z"  
    }  
  ]  
}
```

有关更多信息，请参阅亚马逊 DocumentDB [B 开发者指南ClusterSnapshots](#)中的 [Described B](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbClusterSnapshots](#)中的。

## describe-db-clusters

以下代码示例演示了如何使用 describe-db-clusters。

### AWS CLI

获取有关一个或多个 Amazon DocumentDB 集群的详细信息。

以下describe-db-clusters示例显示了 Amazon DocumentDB 集群的详细信息。sample-cluster通过省略该--db-cluster-identifier参数，您最多可以获得 100 个集群的信息。

```
aws docdb describe-db-clusters
  --db-cluster-identifier sample-cluster
```

输出：

```
{
  "DBClusters": [
    {
      "DBClusterParameterGroup": "default.docdb3.6",
      "Endpoint": "sample-cluster.cluster-corcjozrlsfc.us-
west-2.docdb.amazonaws.com",
      "PreferredBackupWindow": "00:00-00:30",
      "DBClusterIdentifier": "sample-cluster",
      "ClusterCreateTime": "2019-03-15T20:29:58.836Z",
      "LatestRestorableTime": "2019-03-18T20:28:03.239Z",
      "MasterUsername": "master-user",
      "DBClusterMembers": [
        {
          "PromotionTier": 1,
          "DBClusterParameterGroupStatus": "in-sync",
          "IsClusterWriter": false,
          "DBInstanceIdentifier": "sample-cluster"
        },
        {
          "PromotionTier": 1,
          "DBClusterParameterGroupStatus": "in-sync",
          "IsClusterWriter": true,

```



```
        "DBInstanceIdentifier": "sample-cluster2"
      }
    ],
    "PreferredMaintenanceWindow": "sat:04:30-sat:05:00",
    "VpcSecurityGroups": [
      {
        "VpcSecurityGroupId": "sg-77186e0d",
        "Status": "active"
      }
    ],
    "Engine": "docdb",
    "ReaderEndpoint": "sample-cluster.cluster-ro-corcjozrlsfc.us-
west-2.docdb.amazonaws.com",
    "DBSubnetGroup": "default",
    "MultiAZ": true,
    "AvailabilityZones": [
      "us-west-2a",
      "us-west-2c",
      "us-west-2b"
    ],
    "EarliestRestorableTime": "2019-03-15T20:30:47.020Z",
    "DbClusterResourceId": "cluster-UP4EF2PVDDFVHHDJQTYDAIGHLE",
    "DBClusterArn": "arn:aws:rds:us-west-2:123456789012:cluster:sample-
cluster",
    "BackupRetentionPeriod": 3,
    "HostedZoneId": "ZNKXH85TT8WVW",
    "StorageEncrypted": false,
    "EnabledCloudwatchLogsExports": [
      "audit"
    ],
    "AssociatedRoles": [],
    "EngineVersion": "3.6.0",
    "Port": 27017,
    "Status": "available"
  }
]
}
```

有关更多信息，请参阅亚马逊 Document [DB 开发者指南中的描述亚马逊 DocumentDB 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbClusters](#)中的。

## describe-db-engine-versions

以下代码示例演示了如何使用 `describe-db-engine-versions`。

### AWS CLI

列出可用的亚马逊 DocumentDB 引擎版本

以下 `describe-db-engine-versions` 示例列出了所有可用的亚马逊 DocumentDB 引擎版本。

```
aws docdb describe-db-engine-versions \
  --engine docdb
```

输出：

```
{
  "DBEngineVersions": [
    {
      "DBEngineVersionDescription": "DocDB version 1.0.200837",
      "DBParameterGroupFamily": "docdb3.6",
      "EngineVersion": "3.6.0",
      "ValidUpgradeTarget": [],
      "DBEngineDescription": "Amazon DocumentDB (with MongoDB compatibility)",
      "SupportsLogExportsToCloudwatchLogs": true,
      "Engine": "docdb",
      "ExportableLogTypes": [
        "audit"
      ]
    }
  ]
}
```

有关更多信息，请参阅亚马逊 DocumentDB [B 开发者指南EngineVersions中的 Described B](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbEngineVersions](#)中的。

## describe-db-instances

以下代码示例演示了如何使用 `describe-db-instances`。

### AWS CLI

要查找有关预配置的 Amazon DocumentDB 实例的信息

以下describe-db-instances示例显示了有关 Amazon DocumentDB 实例的详细信息。sample-cluster-instance通过省略该--db-instance-identifier参数，您可以获得多达 100 个实例的信息。

```
aws docdb describe-db-instances \  
  --db-instance-identifier sample-cluster-instance
```

输出：

```
{  
  "DBInstances": [  
    {  
      "Endpoint": {  
        "HostedZoneId": "ZNKXH85TT8WVW",  
        "Address": "sample-cluster-instance.corcjozrlsfc.us-  
west-2.docdb.amazonaws.com",  
        "Port": 27017  
      },  
      "PreferredBackupWindow": "00:00-00:30",  
      "DBInstanceStatus": "available",  
      "DBInstanceClass": "db.r4.large",  
      "EnabledCloudwatchLogsExports": [  
        "audit"  
      ],  
      "DBInstanceIdentifier": "sample-cluster-instance",  
      "DBSubnetGroup": {  
        "Subnets": [  
          {  
            "SubnetStatus": "Active",  
            "SubnetIdentifier": "subnet-4e26d263",  
            "SubnetAvailabilityZone": {  
              "Name": "us-west-2a"  
            }  
          },  
          {  
            "SubnetStatus": "Active",  
            "SubnetIdentifier": "subnet-afc329f4",  
            "SubnetAvailabilityZone": {  
              "Name": "us-west-2c"  
            }  
          },  
          {  
            "SubnetStatus": "Active",
```

```
        "SubnetIdentifier": "subnet-53ab3636",
        "SubnetAvailabilityZone": {
            "Name": "us-west-2d"
        }
    },
    {
        "SubnetStatus": "Active",
        "SubnetIdentifier": "subnet-991cb8d0",
        "SubnetAvailabilityZone": {
            "Name": "us-west-2b"
        }
    }
],
"DBSubnetGroupName": "default",
"SubnetGroupStatus": "Complete",
"DBSubnetGroupDescription": "default",
"VpcId": "vpc-91280df6"
},
"InstanceCreateTime": "2019-03-15T20:36:06.338Z",
"Engine": "docdb",
"StorageEncrypted": false,
"AutoMinorVersionUpgrade": true,
"DBInstanceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster-
instance",
"PreferredMaintenanceWindow": "tue:08:39-tue:09:09",
"VpcSecurityGroups": [
    {
        "Status": "active",
        "VpcSecurityGroupId": "sg-77186e0d"
    }
],
"DBClusterIdentifier": "sample-cluster",
"PendingModifiedValues": {},
"BackupRetentionPeriod": 3,
"PubliclyAccessible": false,
"EngineVersion": "3.6.0",
"PromotionTier": 1,
"AvailabilityZone": "us-west-2c",
"DbiResourceId": "db-A2GIKUV6KPOHITGGKI2NHVISZA"
}
]
}
```

有关更多信息，请参阅亚马逊 Document [DB 开发者指南中的描述亚马逊 DocumentDB 实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDbInstances](#) 中的。

## describe-db-subnet-groups

以下代码示例演示了如何使用 describe-db-subnet-groups。

### AWS CLI

检索 Amazon DocumentDB 子网描述列表

以下 describe-db-subnet-groups 示例描述了名为的 Amazon DocumentDB 子网的详细信息。default

```
aws docdb describe-db-subnet-groups \
  --db-subnet-group-name default
```

输出：

```
{
  "DBSubnetGroups": [
    {
      "VpcId": "vpc-91280df6",
      "DBSubnetGroupArn": "arn:aws:rds:us-west-2:123456789012:subgrp:default",
      "Subnets": [
        {
          "SubnetIdentifier": "subnet-4e26d263",
          "SubnetStatus": "Active",
          "SubnetAvailabilityZone": {
            "Name": "us-west-2a"
          }
        },
        {
          "SubnetIdentifier": "subnet-afc329f4",
          "SubnetStatus": "Active",
          "SubnetAvailabilityZone": {
            "Name": "us-west-2c"
          }
        },
        {
          "SubnetIdentifier": "subnet-53ab3636",
```

```
        "SubnetStatus": "Active",
        "SubnetAvailabilityZone": {
            "Name": "us-west-2d"
        }
    },
    {
        "SubnetIdentifier": "subnet-991cb8d0",
        "SubnetStatus": "Active",
        "SubnetAvailabilityZone": {
            "Name": "us-west-2b"
        }
    }
],
"DBSubnetGroupName": "default",
"SubnetGroupStatus": "Complete",
"DBSubnetGroupDescription": "default"
}
]
```

有关更多信息，请参阅 Amazon DocumentDB 开发者指南中的[描述子网组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeDbSubnetGroups](#)中的。

## describe-engine-default-cluster-parameters

以下代码示例演示了如何使用 describe-engine-default-cluster-parameters。

### AWS CLI

描述亚马逊 DocumentDB 的默认引擎和系统参数信息

以下 describe-engine-default-cluster-parameters 示例显示了 Amazon DocumentDB 参数组的默认引擎和系统参数信息的详细信息。docdb3.6

```
aws docdb describe-engine-default-cluster-parameters \
  --db-parameter-group-family docdb3.6
```

输出：

```
{
  "EngineDefaults": {
```

```
"DBParameterGroupFamily": "docdb3.6",
"Parameters": [
  {
    "ApplyType": "dynamic",
    "ParameterValue": "disabled",
    "Description": "Enables auditing on cluster.",
    "Source": "system",
    "DataType": "string",
    "MinimumEngineVersion": "3.6.0",
    "AllowedValues": "enabled,disabled",
    "ParameterName": "audit_logs",
    "IsModifiable": true
  },
  {
    "ApplyType": "static",
    "ParameterValue": "enabled",
    "Description": "Config to enable/disable TLS",
    "Source": "system",
    "DataType": "string",
    "MinimumEngineVersion": "3.6.0",
    "AllowedValues": "disabled,enabled",
    "ParameterName": "tls",
    "IsModifiable": true
  },
  {
    "ApplyType": "dynamic",
    "ParameterValue": "enabled",
    "Description": "Enables TTL Monitoring",
    "Source": "system",
    "DataType": "string",
    "MinimumEngineVersion": "3.6.0",
    "AllowedValues": "disabled,enabled",
    "ParameterName": "ttl_monitor",
    "IsModifiable": true
  }
]
}
```

有关更多信息，请参阅亚马逊 DocumentDB 开发者指南 [DescribeEngineDefaultClusterParameters](#) 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEngineDefaultClusterParameters](#) 中的。

## describe-event-categories

以下代码示例演示了如何使用 `describe-event-categories`。

### AWS CLI

描述所有 Amazon DocumentDB 事件类别

以下 `describe-event-categories` 示例列出了 Amazon DocumentDB 事件源类型的所有类别。db-instance

```
aws docdb describe-event-categories \  
  --source-type db-cluster
```

输出：

```
{  
  "EventCategoriesMapList": [  
    {  
      "SourceType": "db-cluster",  
      "EventCategories": [  
        "failover",  
        "maintenance",  
        "notification",  
        "failure"  
      ]  
    }  
  ]  
}
```

有关更多信息，请参阅亚马逊 DocumentDB 开发者指南中的[查看事件类别](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEventCategories](#) 中的。

## describe-events

以下代码示例演示了如何使用 `describe-events`。

### AWS CLI

列出亚马逊 DocumentDB 活动



以下describe-events示例列出了过去 24 小时 ( 1440 分钟 ) 内的所有亚马逊 DocumentDB 事件。

```
aws docdb describe-events \  
  --duration 1440
```

此命令不生成任何输出。输出：

```
{  
  "Events": [  
    {  
      "EventCategories": [  
        "failover"  
      ],  
      "Message": "Started cross AZ failover to DB instance: sample-cluster",  
      "Date": "2019-03-18T21:36:29.807Z",  
      "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:sample-  
cluster",  
      "SourceIdentifier": "sample-cluster",  
      "SourceType": "db-cluster"  
    },  
    {  
      "EventCategories": [  
        "availability"  
      ],  
      "Message": "DB instance restarted",  
      "Date": "2019-03-18T21:36:40.793Z",  
      "SourceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster",  
      "SourceIdentifier": "sample-cluster",  
      "SourceType": "db-instance"  
    },  
    {  
      "EventCategories": [],  
      "Message": "A new writer was promoted. Restarting database as a  
reader.",  
      "Date": "2019-03-18T21:36:43.873Z",  
      "SourceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster2",  
      "SourceIdentifier": "sample-cluster2",  
      "SourceType": "db-instance"  
    },  
    {  
      "EventCategories": [  
        "availability"  
      ],  
      "Message": "DB instance restarted",  
      "Date": "2019-03-18T21:36:40.793Z",  
      "SourceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster",  
      "SourceIdentifier": "sample-cluster",  
      "SourceType": "db-instance"  
    }  
  ]  
}
```

```
    ],
    "Message": "DB instance restarted",
    "Date": "2019-03-18T21:36:51.257Z",
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster2",
    "SourceIdentifier": "sample-cluster2",
    "SourceType": "db-instance"
  },
  {
    "EventCategories": [
      "failover"
    ],
    "Message": "Completed failover to DB instance: sample-cluster",
    "Date": "2019-03-18T21:36:53.462Z",
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:sample-
cluster",
    "SourceIdentifier": "sample-cluster",
    "SourceType": "db-cluster"
  },
  {
    "Date": "2019-03-19T16:51:48.847Z",
    "EventCategories": [
      "configuration change"
    ],
    "Message": "Updated parameter audit_logs to enabled with apply method
pending-reboot",
    "SourceIdentifier": "custom3-6-param-grp",
    "SourceType": "db-parameter-group"
  },
  {
    "EventCategories": [
      "configuration change"
    ],
    "Message": "Applying modification to database instance class",
    "Date": "2019-03-19T17:55:20.095Z",
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster2",
    "SourceIdentifier": "sample-cluster2",
    "SourceType": "db-instance"
  },
  {
    "EventCategories": [
      "availability"
    ],
    "Message": "DB instance shutdown",
    "Date": "2019-03-19T17:56:31.127Z",
```

```
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster2",
    "SourceIdentifier": "sample-cluster2",
    "SourceType": "db-instance"
  },
  {
    "EventCategories": [
      "configuration change"
    ],
    "Message": "Finished applying modification to DB instance class",
    "Date": "2019-03-19T18:00:45.822Z",
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster2",
    "SourceIdentifier": "sample-cluster2",
    "SourceType": "db-instance"
  },
  {
    "EventCategories": [
      "availability"
    ],
    "Message": "DB instance restarted",
    "Date": "2019-03-19T18:00:53.397Z",
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster2",
    "SourceIdentifier": "sample-cluster2",
    "SourceType": "db-instance"
  },
  {
    "EventCategories": [
      "availability"
    ],
    "Message": "DB instance shutdown",
    "Date": "2019-03-19T18:23:36.045Z",
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster2",
    "SourceIdentifier": "sample-cluster2",
    "SourceType": "db-instance"
  },
  {
    "EventCategories": [
      "availability"
    ],
    "Message": "DB instance restarted",
    "Date": "2019-03-19T18:23:46.209Z",
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster2",
    "SourceIdentifier": "sample-cluster2",
    "SourceType": "db-instance"
  },
  },
```

```
{
  "Date": "2019-03-19T18:39:05.822Z",
  "EventCategories": [
    "configuration change"
  ],
  "Message": "Updated parameter ttl_monitor to enabled with apply method
immediate",
  "SourceIdentifier": "custom3-6-param-grp",
  "SourceType": "db-parameter-group"
},
{
  "Date": "2019-03-19T18:39:48.067Z",
  "EventCategories": [
    "configuration change"
  ],
  "Message": "Updated parameter audit_logs to disabled with apply method
immediate",
  "SourceIdentifier": "custom3-6-param-grp",
  "SourceType": "db-parameter-group"
}
]
```

有关更多信息，请参阅亚马逊 [DocumentDB 开发者指南中的查看亚马逊 DocumentDB 事件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEvents](#) 中的。

## describe-orderable-db-instance-options

以下代码示例演示了如何使用 describe-orderable-db-instance-options。

### AWS CLI

要查找 Amazon DocumentDB 实例选项，您可以订购

以下 describe-orderable-db-instance-options 示例列出了某个地区的 Amazon DocumentDB 的所有实例选项。

```
aws docdb describe-orderable-db-instance-options \
  --engine docdb \
  --region us-east-1
```

输出：

```
{
  "OrderableDBInstanceOptions": [
    {
      "Vpc": true,
      "AvailabilityZones": [
        {
          "Name": "us-east-1a"
        },
        {
          "Name": "us-east-1b"
        },
        {
          "Name": "us-east-1c"
        },
        {
          "Name": "us-east-1d"
        }
      ],
      "EngineVersion": "3.6.0",
      "DBInstanceClass": "db.r4.16xlarge",
      "LicenseModel": "na",
      "Engine": "docdb"
    },
    {
      "Vpc": true,
      "AvailabilityZones": [
        {
          "Name": "us-east-1a"
        },
        {
          "Name": "us-east-1b"
        },
        {
          "Name": "us-east-1c"
        },
        {
          "Name": "us-east-1d"
        }
      ],
      "EngineVersion": "3.6.0",
      "DBInstanceClass": "db.r4.2xlarge",
      "LicenseModel": "na",
```

```
    "Engine": "docdb"
  },
  {
    "Vpc": true,
    "AvailabilityZones": [
      {
        "Name": "us-east-1a"
      },
      {
        "Name": "us-east-1b"
      },
      {
        "Name": "us-east-1c"
      },
      {
        "Name": "us-east-1d"
      }
    ],
    "EngineVersion": "3.6.0",
    "DBInstanceClass": "db.r4.4xlarge",
    "LicenseModel": "na",
    "Engine": "docdb"
  },
  {
    "Vpc": true,
    "AvailabilityZones": [
      {
        "Name": "us-east-1a"
      },
      {
        "Name": "us-east-1b"
      },
      {
        "Name": "us-east-1c"
      },
      {
        "Name": "us-east-1d"
      }
    ],
    "EngineVersion": "3.6.0",
    "DBInstanceClass": "db.r4.8xlarge",
    "LicenseModel": "na",
    "Engine": "docdb"
  },
}
```

```
{
  "Vpc": true,
  "AvailabilityZones": [
    {
      "Name": "us-east-1a"
    },
    {
      "Name": "us-east-1b"
    },
    {
      "Name": "us-east-1c"
    },
    {
      "Name": "us-east-1d"
    }
  ],
  "EngineVersion": "3.6.0",
  "DBInstanceClass": "db.r4.large",
  "LicenseModel": "na",
  "Engine": "docdb"
},
{
  "Vpc": true,
  "AvailabilityZones": [
    {
      "Name": "us-east-1a"
    },
    {
      "Name": "us-east-1b"
    },
    {
      "Name": "us-east-1c"
    },
    {
      "Name": "us-east-1d"
    }
  ],
  "EngineVersion": "3.6.0",
  "DBInstanceClass": "db.r4.xlarge",
  "LicenseModel": "na",
  "Engine": "docdb"
}
]
```

```
}
```

有关更多信息，请参阅[亚马逊 DocumentDB 开发者指南中的向集群添加亚马逊文档数据库实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeOrderableDbInstanceOptions](#)中的。

## describe-pending-maintenance-actions

以下代码示例演示了如何使用 describe-pending-maintenance-actions。

### AWS CLI

列出您待处理的 Amazon DocumentDB 维护操作

以下describe-pending-maintenance-actions示例列出了您所有待处理的 Amazon DocumentDB 维护操作。

```
aws docdb describe-pending-maintenance-actions
```

输出：

```
{
  "PendingMaintenanceActions": []
}
```

有关更多信息，请参阅[亚马逊 DocumentDB 开发者指南中的维护亚马逊 DocumentDB](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePendingMaintenanceActions](#)中的。

## failover-db-cluster

以下代码示例演示了如何使用 failover-db-cluster。

### AWS CLI

强制将 Amazon DocumentDB 集群故障转移到副本

以下failover-db-cluster示例导致 Amazon DocumentDB 集群示例集群中的主实例故障转移到副本。

```
aws docdb failover-db-cluster \
  --db-cluster-identifier sample-cluster
```



输出：

```
{
  "DBCluster": {
    "AssociatedRoles": [],
    "DBClusterIdentifier": "sample-cluster",
    "EngineVersion": "3.6.0",
    "DBSubnetGroup": "default",
    "MasterUsername": "master-user",
    "EarliestRestorableTime": "2019-03-15T20:30:47.020Z",
    "Endpoint": "sample-cluster.cluster-corcjozrlsfc.us-
west-2.docdb.amazonaws.com",
    "AvailabilityZones": [
      "us-west-2a",
      "us-west-2c",
      "us-west-2b"
    ],
    "LatestRestorableTime": "2019-03-18T21:35:23.548Z",
    "PreferredMaintenanceWindow": "sat:04:30-sat:05:00",
    "PreferredBackupWindow": "00:00-00:30",
    "Port": 27017,
    "VpcSecurityGroups": [
      {
        "VpcSecurityGroupId": "sg-77186e0d",
        "Status": "active"
      }
    ],
    "StorageEncrypted": false,
    "ClusterCreateTime": "2019-03-15T20:29:58.836Z",
    "MultiAZ": true,
    "Status": "available",
    "DBClusterMembers": [
      {
        "DBClusterParameterGroupStatus": "in-sync",
        "IsClusterWriter": false,
        "DBInstanceIdentifier": "sample-cluster",
        "PromotionTier": 1
      },
      {
        "DBClusterParameterGroupStatus": "in-sync",
        "IsClusterWriter": true,
        "DBInstanceIdentifier": "sample-cluster2",
        "PromotionTier": 2
      }
    ]
  }
}
```

```
    ],
    "EnabledCloudwatchLogsExports": [
      "audit"
    ],
    "DBClusterParameterGroup": "default.docdb3.6",
    "HostedZoneId": "ZNKXH85TT8WW",
    "DBClusterArn": "arn:aws:rds:us-west-2:123456789012:cluster:sample-cluster",
    "BackupRetentionPeriod": 3,
    "DbClusterResourceId": "cluster-UP4EF2PVDDFVHHDJQTYDAIGHLE",
    "ReaderEndpoint": "sample-cluster.cluster-ro-corcjozrlsfc.us-
west-2.docdb.amazonaws.com",
    "Engine": "docdb"
  }
}
```

有关更多信息，请参阅[亚马逊 DocumentDB 开发者指南中的亚马逊 DocumentDB 故障转移](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [FailoverDbCluster](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出 Amazon DocumentDB 资源上的所有标签

以下 `list-tags-for-resource` 示例列出了 Amazon DocumentDB 集群上的所有标签。sample-cluster

```
aws docdb list-tags-for-resource \
  --resource-name arn:aws:rds:us-west-2:123456789012:cluster:sample-cluster
```

输出：

```
{
  "TagList": [
    {
      "Key": "A",
      "Value": "ALPHA"
    },
    {
      "Key": "B",
```

```
        "Value": ""
      },
      {
        "Key": "C",
        "Value": "CHARLIE"
      }
    ]
  }
}
```

有关更多信息，请参阅[亚马逊 DocumentDB 开发者指南中的在亚马逊 Document DB 资源上列出标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## modify-db-cluster-parameter-group

以下代码示例演示了如何使用 `modify-db-cluster-parameter-group`。

### AWS CLI

修改 Amazon DocumentDB 数据库集群参数组

以下`modify-db-cluster-parameter-group`示例通过将两个参数`ttl_monitor`和`audit_logs`设置为启用来修改 Amazon DocumentDB 集群`audit_logs`参数`custom3-6-param-grp`组。更改将在下次重新启动时生效。

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name custom3-6-param-grp \
  --parameters
  ParameterName=audit_logs,ParameterValue=enabled,ApplyMethod=pending-reboot \
  ParameterName=ttl_monitor,ParameterValue=enabled,ApplyMethod=pending-reboot
```

输出：

```
{
  "DBClusterParameterGroupName": "custom3-6-param-grp"
}
```

有关更多信息，请参阅[亚马逊 DocumentDB 开发者指南中的修改亚马逊 DocumentDB 集群参数组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyDbClusterParameterGroup](#)中的。

## modify-db-cluster-snapshot-attribute

以下代码示例演示了如何使用 `modify-db-cluster-snapshot-attribute`。

### AWS CLI

示例 1：向 Amazon DocumentDB 快照添加属性

以下`modify-db-cluster-snapshot-attribute`示例向 Amazon DocumentDB 集群快照添加了四个属性值。

```
aws docdb modify-db-cluster-snapshot-attribute \  
  --db-cluster-snapshot-identifier sample-cluster-snapshot \  
  --attribute-name restore \  
  --values-to-add all 123456789011 123456789012 123456789013
```

输出：

```
{  
  "DBClusterSnapshotAttributesResult": {  
    "DBClusterSnapshotAttributes": [  
      {  
        "AttributeName": "restore",  
        "AttributeValues": [  
          "all",  
          "123456789011",  
          "123456789012",  
          "123456789013"  
        ]  
      }  
    ],  
    "DBClusterSnapshotIdentifier": "sample-cluster-snapshot"  
  }  
}
```

示例 2：从 Amazon DocumentDB 快照中移除属性

以下`modify-db-cluster-snapshot-attribute`示例从 Amazon DocumentDB 集群快照中删除了两个属性值。

```
aws docdb modify-db-cluster-snapshot-attribute \  
  --db-cluster-snapshot-identifier sample-cluster-snapshot \  
  --attribute-name restore \  
  --values-to-remove 123456789012 all
```

输出：

```
{  
  "DBClusterSnapshotAttributesResult": {  
    "DBClusterSnapshotAttributes": [  
      {  
        "AttributeName": "restore",  
        "AttributeValues": [  
          "123456789011",  
          "123456789013"  
        ]  
      }  
    ],  
    "DBClusterSnapshotIdentifier": "sample-cluster-snapshot"  
  }  
}
```

有关更多信息，请参阅亚马逊 DocumentDB [DB 开发者指南ClusterSnapshotAttribute](#)中的 `modify DB`。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyDbClusterSnapshotAttribute](#)中的。

## modify-db-cluster

以下代码示例演示了如何使用 `modify-db-cluster`。

### AWS CLI

#### 修改亚马逊 DocumentDB 集群

以下`modify-db-cluster`示例修改了 Amazon DocumentDB `sample-cluster` 集群，将自动备份的保留期限设为 7 天，并更改备份和维护的首选时段。所有更改都将在下一个维护时段生效。

```
aws docdb modify-db-cluster \  
  --db-cluster-identifier sample-cluster \  
  --no-apply-immediately \  
  --backup-retention-period 7 \  
  --preferred-maintenance-window
```

```
--preferred-backup-window 18:00-18:30 \  
--preferred-maintenance-window sun:20:00-sun:20:30
```

输出：

```
{  
  "DBCluster": {  
    "Endpoint": "sample-cluster.cluster-corcjozrlsfc.us-  
west-2.docdb.amazonaws.com",  
    "DBClusterMembers": [  
      {  
        "DBClusterParameterGroupStatus": "in-sync",  
        "DBInstanceIdentifier": "sample-cluster",  
        "IsClusterWriter": true,  
        "PromotionTier": 1  
      },  
      {  
        "DBClusterParameterGroupStatus": "in-sync",  
        "DBInstanceIdentifier": "sample-cluster2",  
        "IsClusterWriter": false,  
        "PromotionTier": 2  
      }  
    ],  
    "HostedZoneId": "ZNKXH85TT8WVW",  
    "StorageEncrypted": false,  
    "PreferredBackupWindow": "18:00-18:30",  
    "MultiAZ": true,  
    "EngineVersion": "3.6.0",  
    "MasterUsername": "master-user",  
    "ReaderEndpoint": "sample-cluster.cluster-ro-corcjozrlsfc.us-  
west-2.docdb.amazonaws.com",  
    "DBSubnetGroup": "default",  
    "LatestRestorableTime": "2019-03-18T22:08:13.408Z",  
    "EarliestRestorableTime": "2019-03-15T20:30:47.020Z",  
    "PreferredMaintenanceWindow": "sun:20:00-sun:20:30",  
    "AssociatedRoles": [],  
    "EnabledCloudwatchLogsExports": [  
      "audit"  
    ],  
    "Engine": "docdb",  
    "DBClusterParameterGroup": "default.docdb3.6",  
    "DBClusterArn": "arn:aws:rds:us-west-2:123456789012:cluster:sample-cluster",  
    "BackupRetentionPeriod": 7,  
  }  
}
```

```
    "DBClusterIdentifier": "sample-cluster",
    "AvailabilityZones": [
      "us-west-2a",
      "us-west-2c",
      "us-west-2b"
    ],
    "Status": "available",
    "DbClusterResourceId": "cluster-UP4EF2PVDDFVHHDJQTYDAIGHLE",
    "ClusterCreateTime": "2019-03-15T20:29:58.836Z",
    "VpcSecurityGroups": [
      {
        "VpcSecurityGroupId": "sg-77186e0d",
        "Status": "active"
      }
    ],
    "Port": 27017
  }
}
```

有关更多信息，请参阅亚马逊 DocumentDB [开发者指南中的修改亚马逊 DocumentDB 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyDbCluster](#)中的。

## modify-db-instance

以下代码示例演示了如何使用 modify-db-instance。

### AWS CLI

#### 修改亚马逊文档数据库实例

以下modify-db-instance示例修改了 Amazon DocumentDB sample-cluster2 实例，方法是将其实例类更改db.r4.4xlarge为，将其促销层更改为。5更改会立即生效，但只有在实例状态变为可用后才能看到。

```
aws docdb modify-db-instance \
  --db-instance-identifier sample-cluster2 \
  --apply-immediately \
  --db-instance-class db.r4.4xlarge \
  --promotion-tier 5
```

输出：

```
{
  "DBInstance": {
    "EngineVersion": "3.6.0",
    "StorageEncrypted": false,
    "DBInstanceClass": "db.r4.large",
    "PreferredMaintenanceWindow": "mon:08:39-mon:09:09",
    "AutoMinorVersionUpgrade": true,
    "VpcSecurityGroups": [
      {
        "VpcSecurityGroupId": "sg-77186e0d",
        "Status": "active"
      }
    ],
    "PreferredBackupWindow": "18:00-18:30",
    "EnabledCloudwatchLogsExports": [
      "audit"
    ],
    "AvailabilityZone": "us-west-2f",
    "DBInstanceIdentifier": "sample-cluster2",
    "InstanceCreateTime": "2019-03-15T20:36:06.338Z",
    "Engine": "docdb",
    "BackupRetentionPeriod": 7,
    "DBSubnetGroup": {
      "DBSubnetGroupName": "default",
      "DBSubnetGroupDescription": "default",
      "SubnetGroupStatus": "Complete",
      "Subnets": [
        {
          "SubnetIdentifier": "subnet-4e26d263",
          "SubnetAvailabilityZone": {
            "Name": "us-west-2a"
          },
          "SubnetStatus": "Active"
        },
        {
          "SubnetIdentifier": "subnet-afc329f4",
          "SubnetAvailabilityZone": {
            "Name": "us-west-2c"
          },
          "SubnetStatus": "Active"
        },
        {
          "SubnetIdentifier": "subnet-53ab3636",
```



```
        "SubnetAvailabilityZone": {
            "Name": "us-west-2d"
        },
        "SubnetStatus": "Active"
    },
    {
        "SubnetIdentifier": "subnet-991cb8d0",
        "SubnetAvailabilityZone": {
            "Name": "us-west-2b"
        },
        "SubnetStatus": "Active"
    }
],
"VpcId": "vpc-91280df6"
},
"PromotionTier": 2,
"Endpoint": {
    "Address": "sample-cluster2.corcjozrlsfc.us-west-2.docdb.amazonaws.com",
    "HostedZoneId": "ZNKXH85TT8WW",
    "Port": 27017
},
"DbiResourceId": "db-A2GIKUV6KPOHITGGKI2NHVISZA",
"DBClusterIdentifier": "sample-cluster",
"DBInstanceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster2",
"PendingModifiedValues": {
    "DBInstanceClass": "db.r4.4xlarge"
},
"PubliclyAccessible": false,
"DBInstanceStatus": "available"
}
}
```

有关更多信息，请参阅[亚马逊 DocumentDB 开发者指南中的修改亚马逊文档数据库实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyDbInstance](#)中的。

## modify-db-subnet-group

以下代码示例演示了如何使用 modify-db-subnet-group。

### AWS CLI

#### 修改 Amazon DocumentDB 子网组

以下modify-db-subnet-group示例sample-subnet-group通过添加指定的子网和新的描述来修改子网组。

```
aws docdb modify-db-subnet-group \  
  --db-subnet-group-name sample-subnet-group \  
  --subnet-ids subnet-b3806e8f subnet-53ab3636 subnet-991cb8d0 \  
  --db-subnet-group-description "New subnet description"
```

输出：

```
{  
  "DBSubnetGroup": {  
    "DBSubnetGroupName": "sample-subnet-group",  
    "SubnetGroupStatus": "Complete",  
    "DBSubnetGroupArn": "arn:aws:rds:us-west-2:123456789012:subgrp:sample-  
subnet-group",  
    "VpcId": "vpc-91280df6",  
    "DBSubnetGroupDescription": "New subnet description",  
    "Subnets": [  
      {  
        "SubnetIdentifier": "subnet-b3806e8f",  
        "SubnetStatus": "Active",  
        "SubnetAvailabilityZone": {  
          "Name": "us-west-2a"  
        }  
      },  
      {  
        "SubnetIdentifier": "subnet-53ab3636",  
        "SubnetStatus": "Active",  
        "SubnetAvailabilityZone": {  
          "Name": "us-west-2c"  
        }  
      },  
      {  
        "SubnetIdentifier": "subnet-991cb8d0",  
        "SubnetStatus": "Active",  
        "SubnetAvailabilityZone": {  
          "Name": "us-west-2b"  
        }  
      }  
    ]  
  }  
}
```

```
}
```

有关更多信息，请参阅[亚马逊 DocumentDB 开发者指南中的修改亚马逊 DocumentDB 子网组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyDbSubnetGroup](#) 中的。

## reboot-db-instance

以下代码示例演示了如何使用 `reboot-db-instance`。

### AWS CLI

#### 重启亚马逊 DocumentDB 实例

以下 `reboot-db-instance` 示例重新启动亚马逊文档 `sample-cluster2` 数据库实例。

```
aws docdb reboot-db-instance \  
  --db-instance-identifier sample-cluster2
```

此命令不生成任何输出。输出：

```
{  
  "DBInstance": {  
    "PreferredBackupWindow": "18:00-18:30",  
    "DBInstanceIdentifier": "sample-cluster2",  
    "VpcSecurityGroups": [  
      {  
        "Status": "active",  
        "VpcSecurityGroupId": "sg-77186e0d"  
      }  
    ],  
    "DBSubnetGroup": {  
      "VpcId": "vpc-91280df6",  
      "Subnets": [  
        {  
          "SubnetStatus": "Active",  
          "SubnetAvailabilityZone": {  
            "Name": "us-west-2a"  
          },  
          "SubnetIdentifier": "subnet-4e26d263"  
        },  
        {  
          "SubnetStatus": "Active",
```

```
        "SubnetAvailabilityZone": {
            "Name": "us-west-2c"
        },
        "SubnetIdentifier": "subnet-afc329f4"
    },
    {
        "SubnetStatus": "Active",
        "SubnetAvailabilityZone": {
            "Name": "us-west-2d"
        },
        "SubnetIdentifier": "subnet-53ab3636"
    },
    {
        "SubnetStatus": "Active",
        "SubnetAvailabilityZone": {
            "Name": "us-west-2b"
        },
        "SubnetIdentifier": "subnet-991cb8d0"
    }
],
"SubnetGroupStatus": "Complete",
"DBSubnetGroupName": "default",
"DBSubnetGroupDescription": "default"
},
"PendingModifiedValues": {},
"Endpoint": {
    "Address": "sample-cluster2.corcjozrlsfc.us-west-2.docdb.amazonaws.com",
    "HostedZoneId": "ZNKXH85TT8WVW",
    "Port": 27017
},
"EnabledCloudwatchLogsExports": [
    "audit"
],
"StorageEncrypted": false,
"DbiResourceId": "db-A2GIKUV6KPOHITGGKI2NHVISZA",
"AutoMinorVersionUpgrade": true,
"Engine": "docdb",
"InstanceCreateTime": "2019-03-15T20:36:06.338Z",
"EngineVersion": "3.6.0",
"PromotionTier": 5,
"BackupRetentionPeriod": 7,
"DBClusterIdentifier": "sample-cluster",
"PreferredMaintenanceWindow": "mon:08:39-mon:09:09",
"PubliclyAccessible": false,
```

```
    "DBInstanceClass": "db.r4.4xlarge",
    "AvailabilityZone": "us-west-2d",
    "DBInstanceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-cluster2",
    "DBInstanceStatus": "rebooting"
  }
}
```

有关更多信息，请参阅亚马逊 DocumentDB 开发者指南中的[重启亚马逊 DocumentDB 实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RebootDbInstance](#)中的。

## remove-tags-from-resource

以下代码示例演示了如何使用 `remove-tags-from-resource`。

### AWS CLI

从 Amazon DocumentDB 资源中移除标签

以下 `remove-tags-from-resource` 示例从 Amazon DocumentDB 集群 B 中删除密钥名为的标签。 `sample-cluster`

```
aws docdb remove-tags-from-resource \
  --resource-name arn:aws:rds:us-west-2:123456789012:cluster:sample-cluster \
  --tag-keys B
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 DocumentDB 开发者指南中的[从亚马逊 DocumentDB 资源中移除标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RemoveTagsFromResource](#)中的。

## reset-db-cluster-parameter-group

以下代码示例演示了如何使用 `reset-db-cluster-parameter-group`。

### AWS CLI

在 Amazon DocumentDB 参数组中将指定的参数值重置为其默认值

以下 `reset-db-cluster-parameter-group` 示例将 Amazon DocumentDB 参数 `custom3-6-param-grp` 组 `ttl_monitor` 中的参数重置为其默认值。

```
aws docdb reset-db-cluster-parameter-group \  
  --db-cluster-parameter-group-name custom3-6-param-grp \  
  --parameters ParameterName=ttl_monitor,ApplyMethod=immediate
```

输出：

```
{  
  "DBClusterParameterGroupName": "custom3-6-param-grp"  
}
```

有关更多信息，请参阅亚马逊 DocumentDB 开发者指南中的标题。

在 Amazon DocumentDB 参数组中将指定或所有参数值重置为其默认值

以下 `reset-db-cluster-parameter-group` 示例将 Amazon DocumentDB 参数 `custom3-6-param-grp` 组中的所有参数重置为其默认值。

```
aws docdb reset-db-cluster-parameter-group \  
  --db-cluster-parameter-group-name custom3-6-param-grp \  
  --reset-all-parameters
```

输出：

```
{  
  "DBClusterParameterGroupName": "custom3-6-param-grp"  
}
```

有关更多信息，请参阅亚马逊 DocumentDB 开发者指南中的 [重置亚马逊 DocumentDB 集群参数组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ResetDbClusterParameterGroup](#) 中的。

## restore-db-cluster-from-snapshot

以下代码示例演示了如何使用 `restore-db-cluster-from-snapshot`。

### AWS CLI

从自动或手动快照还原 Amazon DocumentDB 集群

以下 `restore-db-cluster-from-snapshot` 示例根据快照创建了一个名为 `sample-cluster-2019-03-16-00-01-restored` 的新 Amazon DocumentDB 集群。 `rds:sample-cluster-2019-03-16-00-01`

```
aws docdb restore-db-cluster-from-snapshot \  
  --db-cluster-identifier sample-cluster-2019-03-16-00-01-restored \  
  --engine docdb \  
  --snapshot-identifier rds:sample-cluster-2019-03-16-00-01
```

输出：

```
{  
  "DBCluster": {  
    "ClusterCreateTime": "2019-03-19T18:45:01.857Z",  
    "HostedZoneId": "ZNKXH85TT8WVW",  
    "Engine": "docdb",  
    "DBClusterMembers": [],  
    "MultiAZ": false,  
    "AvailabilityZones": [  
      "us-west-2a",  
      "us-west-2c",  
      "us-west-2b"  
    ],  
    "StorageEncrypted": false,  
    "ReaderEndpoint": "sample-cluster-2019-03-16-00-01-restored.cluster-ro-  
corcjzrlsfc.us-west-2.docdb.amazonaws.com",  
    "Endpoint": "sample-cluster-2019-03-16-00-01-restored.cluster-  
corcjzrlsfc.us-west-2.docdb.amazonaws.com",  
    "Port": 27017,  
    "PreferredBackupWindow": "00:00-00:30",  
    "DBSubnetGroup": "default",  
    "DBClusterIdentifier": "sample-cluster-2019-03-16-00-01-restored",  
    "PreferredMaintenanceWindow": "sat:04:30-sat:05:00",  
    "DBClusterArn": "arn:aws:rds:us-west-2:123456789012:cluster:sample-  
cluster-2019-03-16-00-01-restored",  
    "DBClusterParameterGroup": "default.docdb3.6",  
    "DbClusterResourceId": "cluster-X0046Q3RH4LWSYNH3NMZKXPISU",  
    "MasterUsername": "master-user",  
    "EngineVersion": "3.6.0",  
    "BackupRetentionPeriod": 3,  
    "AssociatedRoles": [],  
    "Status": "creating",  
    "VpcSecurityGroups": [  

```

```
{
  "Status": "active",
  "VpcSecurityGroupId": "sg-77186e0d"
}
]
```

有关更多信息，请参阅 Amazon DocumentDB 开发者[指南中的从集群快照恢复](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RestoreDbClusterFromSnapshot](#)中的。

## restore-db-cluster-to-point-in-time

以下代码示例演示了如何使用 `restore-db-cluster-to-point-in-time`。

### AWS CLI

`point-in-time` 从手动快照将 Amazon DocumentDB 集群还原到

以下 `restore-db-cluster-to-point-in-time` 示例使用使用最新的可恢复 `sample-cluster-snapshot` 时间创建新的 Amazon DocumentDB 集群。 `sample-cluster-pit`

```
aws docdb restore-db-cluster-to-point-in-time \
  --db-cluster-identifier sample-cluster-pit \
  --source-db-cluster-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-cluster \
  --use-latest-restorable-time
```

输出：

```
{
  "DBCluster": {
    "StorageEncrypted": false,
    "BackupRetentionPeriod": 3,
    "MasterUsername": "master-user",
    "HostedZoneId": "ZNKXH85TT8WVW",
    "PreferredBackupWindow": "00:00-00:30",
    "MultiAZ": false,
    "DBClusterIdentifier": "sample-cluster-pit",
    "DBSubnetGroup": "default",
    "ClusterCreateTime": "2019-04-03T15:55:21.320Z",
    "AssociatedRoles": [],
```



```

    "DBClusterParameterGroup": "default.docdb3.6",
    "DBClusterMembers": [],
    "Status": "creating",
    "AvailabilityZones": [
        "us-west-2a",
        "us-west-2d",
        "us-west-2b"
    ],
    "ReaderEndpoint": "sample-cluster-pit.cluster-ro-corcjozrlsfc.us-
west-2.docdb.amazonaws.com",
    "Port": 27017,
    "Engine": "docdb",
    "EngineVersion": "3.6.0",
    "VpcSecurityGroups": [
        {
            "VpcSecurityGroupId": "sg-77186e0d",
            "Status": "active"
        }
    ],
    "PreferredMaintenanceWindow": "sat:04:30-sat:05:00",
    "Endpoint": "sample-cluster-pit.cluster-corcjozrlsfc.us-
west-2.docdb.amazonaws.com",
    "DbClusterResourceId": "cluster-NLCABBX0SE2QPQ4GOLZIFWEPLM",
    "DBClusterArn": "arn:aws:rds:us-west-2:123456789012:cluster:sample-cluster-
pit"
}
}

```

有关更多信息，请参阅 Amazon DocumentDB 开发者[指南中的将快照恢复到某个时间点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RestoreDbClusterToPointInTime](#)中的。

## start-db-cluster

以下代码示例演示了如何使用 start-db-cluster。

### AWS CLI

启动已停止的 Amazon DocumentDB 集群

以下 start-db-cluster 示例启动指定的 Amazon DocumentDB 集群。

```
aws docdb start-db-cluster \
```

```
--db-cluster-identifier sample-cluster
```

输出：

```
{
  "DBCluster": {
    "ClusterCreateTime": "2019-03-19T18:45:01.857Z",
    "HostedZoneId": "ZKXHX85TT8WVW",
    "Engine": "docdb",
    "DBClusterMembers": [],
    "MultiAZ": false,
    "AvailabilityZones": [
      "us-east-1a",
      "us-east-1c",
      "us-east-1f"
    ],
    "StorageEncrypted": false,
    "ReaderEndpoint": "sample-cluster-2019-03-16-00-01-restored.cluster-ro-
corcjorzrlsfc.us-east-1.docdb.amazonaws.com",
    "Endpoint": "sample-cluster-2019-03-16-00-01-restored.cluster-
corcjorzrlsfc.us-east-1.docdb.amazonaws.com",
    "Port": 27017,
    "PreferredBackupWindow": "00:00-00:30",
    "DBSubnetGroup": "default",
    "DBClusterIdentifier": "sample-cluster-2019-03-16-00-01-restored",
    "PreferredMaintenanceWindow": "sat:04:30-sat:05:00",
    "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-
cluster-2019-03-16-00-01-restored",
    "DBClusterParameterGroup": "default.docdb3.6",
    "DbClusterResourceId": "cluster-X0046Q3RH4LWSYNH3NMZKXPISU",
    "MasterUsername": "master-user",
    "EngineVersion": "3.6.0",
    "BackupRetentionPeriod": 3,
    "AssociatedRoles": [],
    "Status": "creating",
    "VpcSecurityGroups": [
      {
        "Status": "active",
        "VpcSecurityGroupId": "sg-77186e0d"
      }
    ]
  }
}
```

有关更多信息，请参阅[亚马逊 DocumentDB 开发者指南中的停止和启动亚马逊 DocumentDB 集群](#)。

- 有关 API 的详细信息，请参阅[AWS CLI 命令参考 StartDbCluster](#)中的。

## stop-db-cluster

以下代码示例演示了如何使用 stop-db-cluster。

### AWS CLI

停止正在运行的 Amazon DocumentDB 集群

以下 stop-db-cluster 示例停止指定的亚马逊文档数据库集群。

```
aws docdb stop-db-cluster \
  --db-cluster-identifier sample-cluster
```

输出：

```
{
  "DBCluster": {
    "ClusterCreateTime": "2019-03-19T18:45:01.857Z",
    "HostedZoneId": "ZNKXH85TT8WVW",
    "Engine": "docdb",
    "DBClusterMembers": [],
    "MultiAZ": false,
    "AvailabilityZones": [
      "us-east-1a",
      "us-east-1c",
      "us-east-1f"
    ],
    "StorageEncrypted": false,
    "ReaderEndpoint": "sample-cluster-2019-03-16-00-01-restored.cluster-ro-
corcjorzrlsfc.us-east-1.docdb.amazonaws.com",
    "Endpoint": "sample-cluster-2019-03-16-00-01-restored.cluster-
corcjorzrlsfc.us-east-1.docdb.amazonaws.com",
    "Port": 27017,
    "PreferredBackupWindow": "00:00-00:30",
    "DBSubnetGroup": "default",
    "DBClusterIdentifier": "sample-cluster-2019-03-16-00-01-restored",
    "PreferredMaintenanceWindow": "sat:04:30-sat:05:00",
```

```
    "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-
cluster-2019-03-16-00-01-restored",
    "DBClusterParameterGroup": "default.docdb3.6",
    "DbClusterResourceId": "cluster-X0046Q3RH4LWSYNH3NMZKXPISU",
    "MasterUsername": "master-user",
    "EngineVersion": "3.6.0",
    "BackupRetentionPeriod": 3,
    "AssociatedRoles": [],
    "Status": "creating",
    "VpcSecurityGroups": [
      {
        "Status": "active",
        "VpcSecurityGroupId": "sg-77186e0d"
      }
    ]
  }
}
```

有关更多信息，请参阅亚马逊 Document [tDB 开发者指南中的停止和启动亚马逊 DocumentDB 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopDbCluster](#)中的。

## 使用的 DynamoDB 示例 AWS CLI

以下代码示例向您展示了如何在 DynamoDB 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### batch-get-item

以下代码示例演示了如何使用 `batch-get-item`。

#### AWS CLI

##### 检索表中的多个项

以下 `batch-get-items` 示例将使用一批三个 `GetItem` 请求从 `MusicCollection` 表中读取多个项，并请求该操作所用的读取容量单位数。该命令仅返回 `AlbumTitle` 属性。

```
aws dynamodb batch-get-item \  
  --request-items file://request-items.json \  
  --return-consumed-capacity TOTAL
```

`request-items.json` 的内容：

```
{  
  "MusicCollection": {  
    "Keys": [  
      {  
        "Artist": {"S": "No One You Know"},  
        "SongTitle": {"S": "Call Me Today"}  
      },  
      {  
        "Artist": {"S": "Acme Band"},  
        "SongTitle": {"S": "Happy Day"}  
      },  
      {  
        "Artist": {"S": "No One You Know"},  
        "SongTitle": {"S": "Scared of My Shadow"}  
      }  
    ],  
    "ProjectionExpression": "AlbumTitle"  
  }  
}
```

输出：

```
{  
  "Responses": {
```

```
    "MusicCollection": [
      {
        "AlbumTitle": {
          "S": "Somewhat Famous"
        }
      },
      {
        "AlbumTitle": {
          "S": "Blue Sky Blues"
        }
      },
      {
        "AlbumTitle": {
          "S": "Louder Than Ever"
        }
      }
    ],
    "UnprocessedKeys": {},
    "ConsumedCapacity": [
      {
        "TableName": "MusicCollection",
        "CapacityUnits": 1.5
      }
    ]
  }
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[基本操作](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetItem](#)中的。

## batch-write-item

以下代码示例演示了如何使用 batch-write-item。

### AWS CLI

向表中添加多个项

以下 batch-write-item 示例将使用一批三个 PutItem 请求向 MusicCollection 表中添加三个新项。它还会请求有关操作所用的写入容量单位数以及操作修改的任何项集合的信息。

```
aws dynamodb batch-write-item \
```

```
--request-items file://request-items.json \  
--return-consumed-capacity INDEXES \  
--return-item-collection-metrics SIZE
```

request-items.json 的内容：

```
{  
  "MusicCollection": [  
    {  
      "PutRequest": {  
        "Item": {  
          "Artist": {"S": "No One You Know"},  
          "SongTitle": {"S": "Call Me Today"},  
          "AlbumTitle": {"S": "Somewhat Famous"}  
        }  
      }  
    },  
    {  
      "PutRequest": {  
        "Item": {  
          "Artist": {"S": "Acme Band"},  
          "SongTitle": {"S": "Happy Day"},  
          "AlbumTitle": {"S": "Songs About Life"}  
        }  
      }  
    },  
    {  
      "PutRequest": {  
        "Item": {  
          "Artist": {"S": "No One You Know"},  
          "SongTitle": {"S": "Scared of My Shadow"},  
          "AlbumTitle": {"S": "Blue Sky Blues"}  
        }  
      }  
    }  
  ]  
}
```

输出：

```
{  
  "UnprocessedItems": {},  
  "ItemCollectionMetrics": {
```

```
    "MusicCollection": [
      {
        "ItemCollectionKey": {
          "Artist": {
            "S": "No One You Know"
          }
        },
        "SizeEstimateRangeGB": [
          0.0,
          1.0
        ]
      },
      {
        "ItemCollectionKey": {
          "Artist": {
            "S": "Acme Band"
          }
        },
        "SizeEstimateRangeGB": [
          0.0,
          1.0
        ]
      }
    ],
    "ConsumedCapacity": [
      {
        "TableName": "MusicCollection",
        "CapacityUnits": 6.0,
        "Table": {
          "CapacityUnits": 3.0
        },
        "LocalSecondaryIndexes": {
          "AlbumTitleIndex": {
            "CapacityUnits": 3.0
          }
        }
      }
    ]
  }
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[基本操作](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchWriteItem](#)中的。



## create-backup

以下代码示例演示了如何使用 create-backup。

### AWS CLI

为现有 DynamoDB 表创建备份

以下create-backup示例创建了MusicCollection表的备份。

```
aws dynamodb create-backup \  
  --table-name MusicCollection \  
  --backup-name MusicCollectionBackup
```

输出：

```
{  
  "BackupDetails": {  
    "BackupArn": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection/  
backup/01576616366715-b4e58d3a",  
    "BackupName": "MusicCollectionBackup",  
    "BackupSizeBytes": 0,  
    "BackupStatus": "CREATING",  
    "BackupType": "USER",  
    "BackupCreationDateTime": 1576616366.715  
  }  
}
```

有关更多信息，请参阅《亚马逊 [DynamoDB 开发者指南](#)》中的 [DynamoDB 按需备份和恢复](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateBackup](#)中的。

## create-global-table

以下代码示例演示了如何使用 create-global-table。

### AWS CLI

创建全局表

以下create-global-table示例根据指定的、独立的 AWS 区域中的两个相同表创建全局表。

```
aws dynamodb create-global-table \  
  --table-name MusicCollection
```

```
--global-table-name MusicCollection \  
--replication-group RegionName=us-east-2 RegionName=us-east-1 \  
--region us-east-2
```

输出：

```
{  
  "GlobalTableDescription": {  
    "ReplicationGroup": [  
      {  
        "RegionName": "us-east-2"  
      },  
      {  
        "RegionName": "us-east-1"  
      }  
    ],  
    "GlobalTableArn": "arn:aws:dynamodb::123456789012:global-table/  
MusicCollection",  
    "CreationDateTime": 1576625818.532,  
    "GlobalTableStatus": "CREATING",  
    "GlobalTableName": "MusicCollection"  
  }  
}
```

有关更多信息，请参阅亚马逊 [DynamoDB 开发者指南中的 DynamoDB 全局表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateGlobalTable](#)中的。

## create-table

以下代码示例演示了如何使用 create-table。

### AWS CLI

#### 示例 1：创建带标签的表

以下 create-table 示例将使用指定的属性和键架构来创建名为 MusicCollection 的表。此表使用预配置的吞吐量，并使用默认 AWS 拥有的 CMK 进行静态加密。该命令还将标签应用于该表，其键为 Owner，值为 blueTeam。

```
aws dynamodb create-table \  
--table-name MusicCollection \  
--tags Key=Owner,Value=blueTeam
```

```
--attribute-definitions AttributeName=Artist,AttributeType=S
AttributeName=SongTitle,AttributeType=S \
--key-schema AttributeName=Artist,KeyType=HASH
AttributeName=SongTitle,KeyType=RANGE \
--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \
--tags Key=Owner,Value=blueTeam
```

输出：

```
{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "Artist",
        "AttributeType": "S"
      },
      {
        "AttributeName": "SongTitle",
        "AttributeType": "S"
      }
    ],
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "WriteCapacityUnits": 5,
      "ReadCapacityUnits": 5
    },
    "TableSizeBytes": 0,
    "TableName": "MusicCollection",
    "TableStatus": "CREATING",
    "KeySchema": [
      {
        "KeyType": "HASH",
        "AttributeName": "Artist"
      },
      {
        "KeyType": "RANGE",
        "AttributeName": "SongTitle"
      }
    ],
    "ItemCount": 0,
    "CreationDateTime": "2020-05-26T16:04:41.627000-07:00",
    "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection",
    "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  }
}
```

```
}  
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[表的基本操作](#)。

#### 示例 2：在按需模式下创建表

以下示例将使用按需模式（而不是预调配吞吐量模式）创建名为 MusicCollection 的表。这对于工作负载不可预测的表很有用。

```
aws dynamodb create-table \  
  --table-name MusicCollection \  
  --attribute-definitions AttributeName=Artist,AttributeType=S  
  AttributeName=SongTitle,AttributeType=S \  
  --key-schema AttributeName=Artist,KeyType=HASH  
  AttributeName=SongTitle,KeyType=RANGE \  
  --billing-mode PAY_PER_REQUEST
```

输出：

```
{  
  "TableDescription": {  
    "AttributeDefinitions": [  
      {  
        "AttributeName": "Artist",  
        "AttributeType": "S"  
      },  
      {  
        "AttributeName": "SongTitle",  
        "AttributeType": "S"  
      }  
    ],  
    "TableName": "MusicCollection",  
    "KeySchema": [  
      {  
        "AttributeName": "Artist",  
        "KeyType": "HASH"  
      },  
      {  
        "AttributeName": "SongTitle",  
        "KeyType": "RANGE"  
      }  
    ],  
  },  
}
```

```

    "TableStatus": "CREATING",
    "CreationDateTime": "2020-05-27T11:44:10.807000-07:00",
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 0,
      "WriteCapacityUnits": 0
    },
    "TableSizeBytes": 0,
    "ItemCount": 0,
    "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection",
    "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "BillingModeSummary": {
      "BillingMode": "PAY_PER_REQUEST"
    }
  }
}

```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[表的基本操作](#)。

示例 3：创建表并使用客户托管的 CMK 对其进行加密

以下示例将创建一个名为 MusicCollection 的表并使用客户托管的 CMK 对其进行加密。

```

aws dynamodb create-table \
  --table-name MusicCollection \
  --attribute-definitions AttributeName=Artist,AttributeType=S
  AttributeName=SongTitle,AttributeType=S \
  --key-schema AttributeName=Artist,KeyType=HASH
  AttributeName=SongTitle,KeyType=RANGE \
  --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \
  --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=abcd1234-abcd-1234-
  a123-ab1234a1b234

```

输出：

```

{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "Artist",
        "AttributeType": "S"
      },
      {

```

```
        "AttributeName": "SongTitle",
        "AttributeType": "S"
    }
],
"TableName": "MusicCollection",
"KeySchema": [
    {
        "AttributeName": "Artist",
        "KeyType": "HASH"
    },
    {
        "AttributeName": "SongTitle",
        "KeyType": "RANGE"
    }
],
"TableStatus": "CREATING",
"CreationDateTime": "2020-05-27T11:12:16.431000-07:00",
"ProvisionedThroughput": {
    "NumberOfDecreasesToday": 0,
    "ReadCapacityUnits": 5,
    "WriteCapacityUnits": 5
},
"TableSizeBytes": 0,
"ItemCount": 0,
"TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection",
"TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
"SSEDescription": {
    "Status": "ENABLED",
    "SSEType": "KMS",
    "KMSMasterKeyArn": "arn:aws:kms:us-west-2:123456789012:key/abcd1234-
abcd-1234-a123-ab1234a1b234"
}
}
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[表的基本操作](#)。

#### 示例 4：创建具有本地二级索引的表

以下示例将使用指定的属性和键架构来创建名为 MusicCollection 且其本地二级索引名为 AlbumTitleIndex 的表。

```
aws dynamodb create-table \
```

```

--table-name MusicCollection \
--attribute-definitions AttributeName=Artist,AttributeType=S
AttributeName=SongTitle,AttributeType=S AttributeName=AlbumTitle,AttributeType=S \
--key-schema AttributeName=Artist,KeyType=HASH
AttributeName=SongTitle,KeyType=RANGE \
--provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
--local-secondary-indexes \
  "[
    {
      \"IndexName\": \"AlbumTitleIndex\",
      \"KeySchema\": [
        {\"AttributeName\": \"Artist\", \"KeyType\": \"HASH\"},
        {\"AttributeName\": \"AlbumTitle\", \"KeyType\": \"RANGE\"}
      ],
      \"Projection\": {
        \"ProjectionType\": \"INCLUDE\",
        \"NonKeyAttributes\": [\"Genre\", \"Year\"]
      }
    }
  ]"

```

输出：

```

{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "AlbumTitle",
        "AttributeType": "S"
      },
      {
        "AttributeName": "Artist",
        "AttributeType": "S"
      },
      {
        "AttributeName": "SongTitle",
        "AttributeType": "S"
      }
    ],
    "TableName": "MusicCollection",
    "KeySchema": [
      {
        "AttributeName": "Artist",

```

```
        "KeyType": "HASH"
      },
      {
        "AttributeName": "SongTitle",
        "KeyType": "RANGE"
      }
    ],
    "TableStatus": "CREATING",
    "CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 10,
      "WriteCapacityUnits": 5
    },
    "TableSizeBytes": 0,
    "ItemCount": 0,
    "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection",
    "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "LocalSecondaryIndexes": [
      {
        "IndexName": "AlbumTitleIndex",
        "KeySchema": [
          {
            "AttributeName": "Artist",
            "KeyType": "HASH"
          },
          {
            "AttributeName": "AlbumTitle",
            "KeyType": "RANGE"
          }
        ],
        "Projection": {
          "ProjectionType": "INCLUDE",
          "NonKeyAttributes": [
            "Genre",
            "Year"
          ]
        }
      },
      {
        "IndexSizeBytes": 0,
        "ItemCount": 0,
        "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitleIndex"
      }
    ]
  ]
}
```



```
    }
  }
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[表的基本操作](#)。

#### 示例 5：创建具有全局二级索引的表

以下示例将创建一个名为 GameScores 且其全局二级索引名为 GameTitleIndex 的表。基表的分区键为 UserId，排序键 GameTitle，可以有效地找到特定游戏的单个用户的最佳分数，而 GSI 则具有分区键 GameTitle 和排序键 TopScore，允许您快速找到特定游戏的总体最高分。

```
aws dynamodb create-table \
  --table-name GameScores \
  --attribute-definitions AttributeName=UserId,AttributeType=S
  AttributeName=GameTitle,AttributeType=S AttributeName=TopScore,AttributeType=N \
  --key-schema AttributeName=UserId,KeyType=HASH \
    AttributeName=GameTitle,KeyType=RANGE \
  --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
  --global-secondary-indexes \
    "[
      {
        \"IndexName\": \"GameTitleIndex\",
        \"KeySchema\": [
          {\"AttributeName\": \"GameTitle\", \"KeyType\": \"HASH\"},
          {\"AttributeName\": \"TopScore\", \"KeyType\": \"RANGE\"}
        ],
        \"Projection\": {
          \"ProjectionType\": \"INCLUDE\",
          \"NonKeyAttributes\": [\"UserId\"]
        },
        \"ProvisionedThroughput\": {
          \"ReadCapacityUnits\": 10,
          \"WriteCapacityUnits\": 5
        }
      }
    ]"
```

输出：

```
{
  "TableDescription": {
    "AttributeDefinitions": [
      {
```

```
        "AttributeName": "GameTitle",
        "AttributeType": "S"
    },
    {
        "AttributeName": "TopScore",
        "AttributeType": "N"
    },
    {
        "AttributeName": "UserId",
        "AttributeType": "S"
    }
],
"TableName": "GameScores",
"KeySchema": [
    {
        "AttributeName": "UserId",
        "KeyType": "HASH"
    },
    {
        "AttributeName": "GameTitle",
        "KeyType": "RANGE"
    }
],
"TableStatus": "CREATING",
"CreationDateTime": "2020-05-26T17:28:15.602000-07:00",
"ProvisionedThroughput": {
    "NumberOfDecreasesToday": 0,
    "ReadCapacityUnits": 10,
    "WriteCapacityUnits": 5
},
"TableSizeBytes": 0,
"ItemCount": 0,
"TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
"TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
"GlobalSecondaryIndexes": [
    {
        "IndexName": "GameTitleIndex",
        "KeySchema": [
            {
                "AttributeName": "GameTitle",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "TopScore",
```

```

        "KeyType": "RANGE"
      }
    ],
    "Projection": {
      "ProjectionType": "INCLUDE",
      "NonKeyAttributes": [
        "UserId"
      ]
    },
    "IndexStatus": "CREATING",
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 10,
      "WriteCapacityUnits": 5
    },
    "IndexSizeBytes": 0,
    "ItemCount": 0,
    "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/index/GameTitleIndex"
  }
]
}
}
}

```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[表的基本操作](#)。

示例 6：一次创建一个具有多个全局二级索引的表

以下示例将创建一个名为 GameScores 且具有两个全局二级索引的表。GSI 架构通过文件传递，而不是通过命令行传递。

```

aws dynamodb create-table \
  --table-name GameScores \
  --attribute-definitions AttributeName=UserId,AttributeType=S
  AttributeName=GameTitle,AttributeType=S AttributeName=TopScore,AttributeType=N
  AttributeName=Date,AttributeType=S \
  --key-schema AttributeName=UserId,KeyType=HASH
  AttributeName=GameTitle,KeyType=RANGE \
  --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
  --global-secondary-indexes file://gsi.json

```

gsi.json 的内容：

```
[
  {
    "IndexName": "GameTitleIndex",
    "KeySchema": [
      {
        "AttributeName": "GameTitle",
        "KeyType": "HASH"
      },
      {
        "AttributeName": "TopScore",
        "KeyType": "RANGE"
      }
    ],
    "Projection": {
      "ProjectionType": "ALL"
    },
    "ProvisionedThroughput": {
      "ReadCapacityUnits": 10,
      "WriteCapacityUnits": 5
    }
  },
  {
    "IndexName": "GameDateIndex",
    "KeySchema": [
      {
        "AttributeName": "GameTitle",
        "KeyType": "HASH"
      },
      {
        "AttributeName": "Date",
        "KeyType": "RANGE"
      }
    ],
    "Projection": {
      "ProjectionType": "ALL"
    },
    "ProvisionedThroughput": {
      "ReadCapacityUnits": 5,
      "WriteCapacityUnits": 5
    }
  }
]
```

输出：

```
{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "Date",
        "AttributeType": "S"
      },
      {
        "AttributeName": "GameTitle",
        "AttributeType": "S"
      },
      {
        "AttributeName": "TopScore",
        "AttributeType": "N"
      },
      {
        "AttributeName": "UserId",
        "AttributeType": "S"
      }
    ],
    "TableName": "GameScores",
    "KeySchema": [
      {
        "AttributeName": "UserId",
        "KeyType": "HASH"
      },
      {
        "AttributeName": "GameTitle",
        "KeyType": "RANGE"
      }
    ],
    "TableStatus": "CREATING",
    "CreationDateTime": "2020-08-04T16:40:55.524000-07:00",
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 10,
      "WriteCapacityUnits": 5
    },
    "TableSizeBytes": 0,
    "ItemCount": 0,
    "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
    "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  }
}
```

```
"GlobalSecondaryIndexes": [  
  {  
    "IndexName": "GameTitleIndex",  
    "KeySchema": [  
      {  
        "AttributeName": "GameTitle",  
        "KeyType": "HASH"  
      },  
      {  
        "AttributeName": "TopScore",  
        "KeyType": "RANGE"  
      }  
    ],  
    "Projection": {  
      "ProjectionType": "ALL"  
    },  
    "IndexStatus": "CREATING",  
    "ProvisionedThroughput": {  
      "NumberOfDecreasesToday": 0,  
      "ReadCapacityUnits": 10,  
      "WriteCapacityUnits": 5  
    },  
    "IndexSizeBytes": 0,  
    "ItemCount": 0,  
    "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/  
GameScores/index/GameTitleIndex"  
  },  
  {  
    "IndexName": "GameDateIndex",  
    "KeySchema": [  
      {  
        "AttributeName": "GameTitle",  
        "KeyType": "HASH"  
      },  
      {  
        "AttributeName": "Date",  
        "KeyType": "RANGE"  
      }  
    ],  
    "Projection": {  
      "ProjectionType": "ALL"  
    },  
    "IndexStatus": "CREATING",  
    "ProvisionedThroughput": {
```

```

        "NumberOfDecreasesToday": 0,
        "ReadCapacityUnits": 5,
        "WriteCapacityUnits": 5
    },
    "IndexSizeBytes": 0,
    "ItemCount": 0,
    "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/index/GameDateIndex"
    }
]
}
}
}

```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[表的基本操作](#)。

#### 示例 7：创建启用了 Streams 的表

以下示例将创建一个名为 GameScores 且启用了 DynamoDB Streams 的表。每个项的新旧映像都将写入流中。

```

aws dynamodb create-table \
  --table-name GameScores \
  --attribute-definitions AttributeName=UserId,AttributeType=S
AttributeType=S \
  --key-schema AttributeName=UserId,KeyType=HASH
AttributeType=S,KeyName=GameTitle,KeyType=RANGE \
  --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
  --stream-specification StreamEnabled=TRUE,StreamViewType=NEW_AND_OLD_IMAGES

```

输出：

```

{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "GameTitle",
        "AttributeType": "S"
      },
      {
        "AttributeName": "UserId",
        "AttributeType": "S"
      }
    ],

```

```

    "TableName": "GameScores",
    "KeySchema": [
      {
        "AttributeName": "UserId",
        "KeyType": "HASH"
      },
      {
        "AttributeName": "GameTitle",
        "KeyType": "RANGE"
      }
    ],
    "TableStatus": "CREATING",
    "CreationDateTime": "2020-05-27T10:49:34.056000-07:00",
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 10,
      "WriteCapacityUnits": 5
    },
    "TableSizeBytes": 0,
    "ItemCount": 0,
    "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
    "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "StreamSpecification": {
      "StreamEnabled": true,
      "StreamViewType": "NEW_AND_OLD_IMAGES"
    },
    "LatestStreamLabel": "2020-05-27T17:49:34.056",
    "LatestStreamArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/stream/2020-05-27T17:49:34.056"
  }
}

```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[表的基本操作](#)。

#### 示例 8：创建启用了 Keys-Only Stream 的表

以下示例将创建一个名为 GameScores 且启用了 DynamoDB Streams 的表。仅将所修改项的键属性写入流中。

```

aws dynamodb create-table \
  --table-name GameScores \
  --attribute-definitions AttributeName=UserId,AttributeType=S
AttributeName=GameTitle,AttributeType=S \

```



```
--key-schema AttributeName=UserId,KeyType=HASH
AttributeName=GameTitle,KeyType=RANGE \
--provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
--stream-specification StreamEnabled=TRUE,StreamViewType=KEYS_ONLY
```

输出：

```
{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "GameTitle",
        "AttributeType": "S"
      },
      {
        "AttributeName": "UserId",
        "AttributeType": "S"
      }
    ],
    "TableName": "GameScores",
    "KeySchema": [
      {
        "AttributeName": "UserId",
        "KeyType": "HASH"
      },
      {
        "AttributeName": "GameTitle",
        "KeyType": "RANGE"
      }
    ],
    "TableStatus": "CREATING",
    "CreationDateTime": "2023-05-25T18:45:34.140000+00:00",
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 10,
      "WriteCapacityUnits": 5
    },
    "TableSizeBytes": 0,
    "ItemCount": 0,
    "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
    "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "StreamSpecification": {
      "StreamEnabled": true,
```

```
        "StreamViewType": "KEYS_ONLY"
    },
    "LatestStreamLabel": "2023-05-25T18:45:34.140",
    "LatestStreamArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/stream/2023-05-25T18:45:34.140",
    "DeletionProtectionEnabled": false
}
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[更改 DynamoDB Streams 的数据捕获](#)。

### 示例 9：使用 Standard Infrequent Access 类创建表

以下示例将创建名为 GameScores 的表并分配 Standard-Infrequent Access ( DynamoDB 标准-IA ) 表类。此表类针对主要的存储成本进行了优化。

```
aws dynamodb create-table \
  --table-name GameScores \
  --attribute-definitions AttributeName=UserId,AttributeType=S
AttributeName=GameTitle,AttributeType=S \
  --key-schema AttributeName=UserId,KeyType=HASH
AttributeName=GameTitle,KeyType=RANGE \
  --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
  --table-class STANDARD_INFREQUENT_ACCESS
```

输出：

```
{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "GameTitle",
        "AttributeType": "S"
      },
      {
        "AttributeName": "UserId",
        "AttributeType": "S"
      }
    ],
    "TableName": "GameScores",
    "KeySchema": [
```

```

    {
      "AttributeName": "UserId",
      "KeyType": "HASH"
    },
    {
      "AttributeName": "GameTitle",
      "KeyType": "RANGE"
    }
  ],
  "TableStatus": "CREATING",
  "CreationDateTime": "2023-05-25T18:33:07.581000+00:00",
  "ProvisionedThroughput": {
    "NumberOfDecreasesToday": 0,
    "ReadCapacityUnits": 10,
    "WriteCapacityUnits": 5
  },
  "TableSizeBytes": 0,
  "ItemCount": 0,
  "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
  "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "TableClassSummary": {
    "TableClass": "STANDARD_INFREQUENT_ACCESS"
  },
  "DeletionProtectionEnabled": false
}
}

```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[表类](#)。

示例 10：创建启用了删除保护功能的表

以下示例将创建一个名为 GameScores 的表并启用删除保护。

```

aws dynamodb create-table \
  --table-name GameScores \
  --attribute-definitions AttributeName=UserId,AttributeType=S \
  AttributeName=GameTitle,AttributeType=S \
  --key-schema AttributeName=UserId,KeyType=HASH \
  AttributeName=GameTitle,KeyType=RANGE \
  --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
  --deletion-protection-enabled

```

输出：

```
{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "GameTitle",
        "AttributeType": "S"
      },
      {
        "AttributeName": "UserId",
        "AttributeType": "S"
      }
    ],
    "TableName": "GameScores",
    "KeySchema": [
      {
        "AttributeName": "UserId",
        "KeyType": "HASH"
      },
      {
        "AttributeName": "GameTitle",
        "KeyType": "RANGE"
      }
    ],
    "TableStatus": "CREATING",
    "CreationDateTime": "2023-05-25T23:02:17.093000+00:00",
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 10,
      "WriteCapacityUnits": 5
    },
    "TableSizeBytes": 0,
    "ItemCount": 0,
    "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
    "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "DeletionProtectionEnabled": true
  }
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[使用删除保护](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTable](#)中的。

## delete-backup

以下代码示例演示了如何使用 delete-backup。

### AWS CLI

删除现有 DynamoDB 备份

以下delete-backup示例删除了指定的现有备份。

```
aws dynamodb delete-backup \  
  --backup-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection/  
  backup/01576616366715-b4e58d3a
```

输出：

```
{  
  "BackupDescription": {  
    "BackupDetails": {  
      "BackupArn": "arn:aws:dynamodb:us-west-2:123456789012:table/  
MusicCollection/backup/01576616366715-b4e58d3a",  
      "BackupName": "MusicCollectionBackup",  
      "BackupSizeBytes": 0,  
      "BackupStatus": "DELETED",  
      "BackupType": "USER",  
      "BackupCreationDateTime": 1576616366.715  
    },  
    "SourceTableDetails": {  
      "TableName": "MusicCollection",  
      "TableId": "b0c04bcc-309b-4352-b2ae-9088af169fe2",  
      "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/  
MusicCollection",  
      "TableSizeBytes": 0,  
      "KeySchema": [  
        {  
          "AttributeName": "Artist",  
          "KeyType": "HASH"  
        },  
        {  
          "AttributeName": "SongTitle",  
          "KeyType": "RANGE"  
        }  
      ],  
    }  
  }  
}
```

```
    "TableCreationDateTime": 1576615228.571,  
    "ProvisionedThroughput": {  
      "ReadCapacityUnits": 5,  
      "WriteCapacityUnits": 5  
    },  
    "ItemCount": 0,  
    "BillingMode": "PROVISIONED"  
  },  
  "SourceTableFeatureDetails": {}  
}
```

有关更多信息，请参阅《亚马逊 [DynamoDB 开发者指南](#)》中的 [DynamoDB 按需备份和恢复](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBackup](#)中的。

## delete-item

以下代码示例演示了如何使用 delete-item。

### AWS CLI

#### 示例 1：删除项

以下 delete-item 示例将从 MusicCollection 表中删除项，并请求有关已删除的项以及请求使用的容量的详细信息。

```
aws dynamodb delete-item \  
  --table-name MusicCollection \  
  --key file://key.json \  
  --return-values ALL_OLD \  
  --return-consumed-capacity TOTAL \  
  --return-item-collection-metrics SIZE
```

key.json 的内容：

```
{  
  "Artist": {"S": "No One You Know"},  
  "SongTitle": {"S": "Scared of My Shadow"}  
}
```

输出：

```
{
  "Attributes": {
    "AlbumTitle": {
      "S": "Blue Sky Blues"
    },
    "Artist": {
      "S": "No One You Know"
    },
    "SongTitle": {
      "S": "Scared of My Shadow"
    }
  },
  "ConsumedCapacity": {
    "TableName": "MusicCollection",
    "CapacityUnits": 2.0
  },
  "ItemCollectionMetrics": {
    "ItemCollectionKey": {
      "Artist": {
        "S": "No One You Know"
      }
    },
    "SizeEstimateRangeGB": [
      0.0,
      1.0
    ]
  }
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[写入项](#)。

#### 示例 2：有条件地删除项

以下示例仅在某项的 ProductCategory 为 Sporting Goods 或 Gardening Supplies 且其价格介于 500 和 600 之间时，才会将其从 ProductCatalog 表中删除。它会返回有关已删除项的详细信息。

```
aws dynamodb delete-item \
  --table-name ProductCatalog \
  --key '{"Id":{"N":"456"}}' \
  --condition-expression "(ProductCategory IN (:cat1, :cat2)) and (#P between :lo
and :hi)" \
  --expression-attribute-names file://names.json \
```

```
--expression-attribute-values file://values.json \  
--return-values ALL_OLD
```

names.json 的内容：

```
{  
  "#P": "Price"  
}
```

values.json 的内容：

```
{  
  "cat1": {"S": "Sporting Goods"},  
  "cat2": {"S": "Gardening Supplies"},  
  "lo": {"N": "500"},  
  "hi": {"N": "600"}  
}
```

输出：

```
{  
  "Attributes": {  
    "Id": {  
      "N": "456"  
    },  
    "Price": {  
      "N": "550"  
    },  
    "ProductCategory": {  
      "S": "Sporting Goods"  
    }  
  }  
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[写入项](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteItem](#)中的。

## delete-table

以下代码示例演示了如何使用 delete-table。



## AWS CLI

### 删除表

以下 `delete-table` 示例将删除 `MusicCollection` 表。

```
aws dynamodb delete-table \  
  --table-name MusicCollection
```

输出：

```
{  
  "TableDescription": {  
    "TableStatus": "DELETING",  
    "TableSizeBytes": 0,  
    "ItemCount": 0,  
    "TableName": "MusicCollection",  
    "ProvisionedThroughput": {  
      "NumberOfDecreasesToday": 0,  
      "WriteCapacityUnits": 5,  
      "ReadCapacityUnits": 5  
    }  
  }  
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[删除表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTable](#)中的。

## describe-backup

以下代码示例演示了如何使用 `describe-backup`。

### AWS CLI

获取有关表现有备份的信息

以下 `describe-backup` 示例显示有关指定现有备份的信息。

```
aws dynamodb describe-backup \  
  --backup-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection/  
  backup/01576616366715-b4e58d3a
```

输出：

```
{
  "BackupDescription": {
    "BackupDetails": {
      "BackupArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/backup/01576616366715-b4e58d3a",
      "BackupName": "MusicCollectionBackup",
      "BackupSizeBytes": 0,
      "BackupStatus": "AVAILABLE",
      "BackupType": "USER",
      "BackupCreationDateTime": 1576616366.715
    },
    "SourceTableDetails": {
      "TableName": "MusicCollection",
      "TableId": "b0c04bcc-309b-4352-b2ae-9088af169fe2",
      "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
      "TableSizeBytes": 0,
      "KeySchema": [
        {
          "AttributeName": "Artist",
          "KeyType": "HASH"
        },
        {
          "AttributeName": "SongTitle",
          "KeyType": "RANGE"
        }
      ],
      "TableCreationDateTime": 1576615228.571,
      "ProvisionedThroughput": {
        "ReadCapacityUnits": 5,
        "WriteCapacityUnits": 5
      },
      "ItemCount": 0,
      "BillingMode": "PROVISIONED"
    },
    "SourceTableFeatureDetails": {}
  }
}
```

有关更多信息，请参阅《亚马逊 [DynamoDB 开发者指南](#)》中的 [DynamoDB 按需备份和恢复](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeBackup](#) 中的。

## describe-continuous-backups

以下代码示例演示了如何使用 describe-continuous-backups。

### AWS CLI

获取有关 DynamoDB 表连续备份的信息

以下describe-continuous-backups示例显示了有关MusicCollection表连续备份设置的详细信息。

```
aws dynamodb describe-continuous-backups \  
  --table-name MusicCollection
```

输出：

```
{  
  "ContinuousBackupsDescription": {  
    "ContinuousBackupsStatus": "ENABLED",  
    "PointInTimeRecoveryDescription": {  
      "PointInTimeRecoveryStatus": "DISABLED"  
    }  
  }  
}
```

有关更多信息，请参阅《亚马逊 DynamoDB 开发者指南》中的 [DynamoDB 时间点恢复](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeContinuousBackups](#)中的。

## describe-contributor-insights

以下代码示例演示了如何使用 describe-contributor-insights。

### AWS CLI

查看 DynamoDB 表的“贡献者见解”设置

以下describe-contributor-insights示例显示了MusicCollection表和AlbumTitle-index全局二级索引的“贡献者见解”设置。

```
aws dynamodb describe-contributor-insights \  
  --table-name MusicCollection \  
  --index-name AlbumTitle-index
```

```
--index-name AlbumTitle-index
```

输出：

```
{
  "TableName": "MusicCollection",
  "IndexName": "AlbumTitle-index",
  "ContributorInsightsRuleList": [
    "DynamoDBContributorInsights-PKC-MusicCollection-1576629651520",
    "DynamoDBContributorInsights-SKC-MusicCollection-1576629651520",
    "DynamoDBContributorInsights-PKT-MusicCollection-1576629651520",
    "DynamoDBContributorInsights-SKT-MusicCollection-1576629651520"
  ],
  "ContributorInsightsStatus": "ENABLED",
  "LastUpdateDateTime": 1576629654.78
}
```

有关更多信息，请参阅《亚马逊 [DynamoDB 开发者指南](#)》中的“使用 CloudWatch 贡献者洞察分析 DynamoDB 的数据访问权限”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeContributorInsights](#) 中的。

## describe-endpoints

以下代码示例演示了如何使用 describe-endpoints。

### AWS CLI

查看区域终端节点信息

以下 describe-endpoints 示例显示有关当前 AWS 区域终端节点的详细信息。

```
aws dynamodb describe-endpoints
```

输出：

```
{
  "Endpoints": [
    {
      "Address": "dynamodb.us-west-2.amazonaws.com",
      "CachePeriodInMinutes": 1440
    }
  ]
}
```

```
]
}
```

有关更多信息，请参阅一般参考中的[亚马逊 DynamoDB 终端节点和配额](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEndpoints](#)中的。

## describe-global-table-settings

以下代码示例演示了如何使用 describe-global-table-settings。

### AWS CLI

获取有关 DynamoDB 全局表设置的信息

以下describe-global-table-settings示例显示了MusicCollection全局表的设置。

```
aws dynamodb describe-global-table-settings \
  --global-table-name MusicCollection
```

输出：

```
{
  "GlobalTableName": "MusicCollection",
  "ReplicaSettings": [
    {
      "RegionName": "us-east-1",
      "ReplicaStatus": "ACTIVE",
      "ReplicaProvisionedReadCapacityUnits": 10,
      "ReplicaProvisionedReadCapacityAutoScalingSettings": {
        "AutoScalingDisabled": true
      },
      "ReplicaProvisionedWriteCapacityUnits": 5,
      "ReplicaProvisionedWriteCapacityAutoScalingSettings": {
        "AutoScalingDisabled": true
      }
    },
    {
      "RegionName": "us-east-2",
      "ReplicaStatus": "ACTIVE",
      "ReplicaProvisionedReadCapacityUnits": 10,
      "ReplicaProvisionedReadCapacityAutoScalingSettings": {
        "AutoScalingDisabled": true
      }
    }
  ]
}
```

```

    },
    "ReplicaProvisionedWriteCapacityUnits": 5,
    "ReplicaProvisionedWriteCapacityAutoScalingSettings": {
      "AutoScalingDisabled": true
    }
  }
]
}

```

有关更多信息，请参阅亚马逊 [DynamoDB 开发者指南中的 DynamoDB 全局表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeGlobalTableSettings](#) 中的。

## describe-global-table

以下代码示例演示了如何使用 describe-global-table。

### AWS CLI

显示有关 DynamoDB 全局表的信息

以下 describe-global-table 示例显示了有关 MusicCollection 全局表的详细信息。

```
aws dynamodb describe-global-table \
  --global-table-name MusicCollection
```

输出：

```
{
  "GlobalTableDescription": {
    "ReplicationGroup": [
      {
        "RegionName": "us-east-2"
      },
      {
        "RegionName": "us-east-1"
      }
    ],
    "GlobalTableArn": "arn:aws:dynamodb::123456789012:global-table/
MusicCollection",
    "CreationDateTime": 1576625818.532,
    "GlobalTableStatus": "ACTIVE",
    "GlobalTableName": "MusicCollection"
  }
}
```

```
}  
}
```

有关更多信息，请参阅亚马逊 [DynamoDB 开发者指南中的 DynamoDB 全局表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeGlobalTable](#)中的。

## describe-limits

以下代码示例演示了如何使用 describe-limits。

### AWS CLI

查看预配置容量限制

以下describe-limits示例显示了您账户在当前 AWS 区域的预配置容量限制。

```
aws dynamodb describe-limits
```

输出：

```
{  
  "AccountMaxReadCapacityUnits": 80000,  
  "AccountMaxWriteCapacityUnits": 80000,  
  "TableMaxReadCapacityUnits": 40000,  
  "TableMaxWriteCapacityUnits": 40000  
}
```

有关更多信息，请参阅亚马逊 [DynamoDB 开发者指南中的 DynamoDB 限制](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLimits](#)中的。

## describe-table-replica-auto-scaling

以下代码示例演示了如何使用 describe-table-replica-auto-scaling。

### AWS CLI

查看跨全局表副本的 auto 缩放设置

以下describe-table-replica-auto-scaling示例显示了全MusicCollection全局表副本之间的 auto Scaling 设置。

```
aws dynamodb describe-table-replica-auto-scaling \  
--table-name MusicCollection
```

输出：

```
{  
  "TableAutoScalingDescription": {  
    "TableName": "MusicCollection",  
    "TableStatus": "ACTIVE",  
    "Replicas": [  
      {  
        "RegionName": "us-east-1",  
        "GlobalSecondaryIndexes": [],  
        "ReplicaProvisionedReadCapacityAutoScalingSettings": {  
          "MinimumUnits": 5,  
          "MaximumUnits": 40000,  
          "AutoScalingRoleArn": "arn:aws:iam::123456789012:role/  
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/  
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable",  
          "ScalingPolicies": [  
            {  
              "PolicyName": "DynamoDBReadCapacityUtilization:table/  
MusicCollection",  
              "TargetTrackingScalingPolicyConfiguration": {  
                "TargetValue": 70.0  
              }  
            }  
          ]  
        },  
        "ReplicaProvisionedWriteCapacityAutoScalingSettings": {  
          "MinimumUnits": 5,  
          "MaximumUnits": 40000,  
          "AutoScalingRoleArn": "arn:aws:iam::123456789012:role/  
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/  
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable",  
          "ScalingPolicies": [  
            {  
              "PolicyName": "DynamoDBWriteCapacityUtilization:table/  
MusicCollection",  
              "TargetTrackingScalingPolicyConfiguration": {  
                "TargetValue": 70.0  
              }  
            }  
          ]  
        }  
      }  
    ]  
  }  
}
```



```
    ]
  },
  "ReplicaStatus": "ACTIVE"
},
{
  "RegionName": "us-east-2",
  "GlobalSecondaryIndexes": [],
  "ReplicaProvisionedReadCapacityAutoScalingSettings": {
    "MinimumUnits": 5,
    "MaximumUnits": 40000,
    "AutoScalingRoleArn": "arn:aws:iam::123456789012:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable",
    "ScalingPolicies": [
      {
        "PolicyName": "DynamoDBReadCapacityUtilization:table/
MusicCollection",
        "TargetTrackingScalingPolicyConfiguration": {
          "TargetValue": 70.0
        }
      }
    ]
  },
  "ReplicaProvisionedWriteCapacityAutoScalingSettings": {
    "MinimumUnits": 5,
    "MaximumUnits": 40000,
    "AutoScalingRoleArn": "arn:aws:iam::123456789012:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable",
    "ScalingPolicies": [
      {
        "PolicyName": "DynamoDBWriteCapacityUtilization:table/
MusicCollection",
        "TargetTrackingScalingPolicyConfiguration": {
          "TargetValue": 70.0
        }
      }
    ]
  },
  "ReplicaStatus": "ACTIVE"
}
]
}
```

```
}
```

有关更多信息，请参阅亚马逊 [DynamoDB 开发者指南中的 DynamoDB 全局表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeTableReplicaAutoScaling](#) 中的。

## describe-table

以下代码示例演示了如何使用 describe-table。

### AWS CLI

#### 描述表

以下 describe-table 示例将描述 MusicCollection 表。

```
aws dynamodb describe-table \  
  --table-name MusicCollection
```

输出：

```
{  
  "Table": {  
    "AttributeDefinitions": [  
      {  
        "AttributeName": "Artist",  
        "AttributeType": "S"  
      },  
      {  
        "AttributeName": "SongTitle",  
        "AttributeType": "S"  
      }  
    ],  
    "ProvisionedThroughput": {  
      "NumberOfDecreasesToday": 0,  
      "WriteCapacityUnits": 5,  
      "ReadCapacityUnits": 5  
    },  
    "TableSizeBytes": 0,  
    "TableName": "MusicCollection",  
    "TableStatus": "ACTIVE",  
    "KeySchema": [  
      {
```

```
        "KeyType": "HASH",
        "AttributeName": "Artist"
    },
    {
        "KeyType": "RANGE",
        "AttributeName": "SongTitle"
    }
],
"ItemCount": 0,
"CreationDateTime": 1421866952.062
}
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[描述表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTable](#)中的。

## describe-time-to-live

以下代码示例演示了如何使用 describe-time-to-live。

### AWS CLI

#### 查看桌子的生存时间设置

以下describe-time-to-live示例显示了MusicCollection表格的生存时间设置。

```
aws dynamodb describe-time-to-live \
  --table-name MusicCollection
```

输出：

```
{
  "TimeToLiveDescription": {
    "TimeToLiveStatus": "ENABLED",
    "AttributeName": "ttl"
  }
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的[上线时间](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTimeToLive](#)中的。

## get-item

以下代码示例演示了如何使用 `get-item`。

### AWS CLI

#### 示例 1：读取表中的项

以下 `get-item` 示例将从 `MusicCollection` 表中检索项。该表具有 `hash-and-range` 主键 (`Artist`和`SongTitle`)，因此必须同时指定这两个属性。该命令还请求有关操作所用的读取容量的信息。

```
aws dynamodb get-item \  
  --table-name MusicCollection \  
  --key file://key.json \  
  --return-consumed-capacity TOTAL
```

`key.json` 的内容：

```
{  
  "Artist": {"S": "Acme Band"},  
  "SongTitle": {"S": "Happy Day"}  
}
```

输出：

```
{  
  "Item": {  
    "AlbumTitle": {  
      "S": "Songs About Life"  
    },  
    "SongTitle": {  
      "S": "Happy Day"  
    },  
    "Artist": {  
      "S": "Acme Band"  
    }  
  },  
  "ConsumedCapacity": {  
    "TableName": "MusicCollection",  
    "CapacityUnits": 0.5  
  }  
}
```

```
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[读取项](#)。

#### 示例 2：使用一致性读取来读取项

以下示例将使用强一致性读取从 MusicCollection 表中检索项。

```
aws dynamodb get-item \  
  --table-name MusicCollection \  
  --key file://key.json \  
  --consistent-read \  
  --return-consumed-capacity TOTAL
```

key.json 的内容：

```
{  
  "Artist": {"S": "Acme Band"},  
  "SongTitle": {"S": "Happy Day"}  
}
```

输出：

```
{  
  "Item": {  
    "AlbumTitle": {  
      "S": "Songs About Life"  
    },  
    "SongTitle": {  
      "S": "Happy Day"  
    },  
    "Artist": {  
      "S": "Acme Band"  
    }  
  },  
  "ConsumedCapacity": {  
    "TableName": "MusicCollection",  
    "CapacityUnits": 1.0  
  }  
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[读取项](#)。

### 示例 3：检索项的特定属性

以下示例将使用投影表达式仅检索所需项的三个属性。

```
aws dynamodb get-item \  
  --table-name ProductCatalog \  
  --key '{"Id": {"N": "102"}}' \  
  --projection-expression "#T, #C, #P" \  
  --expression-attribute-names file://names.json
```

names.json 的内容：

```
{  
  "#T": "Title",  
  "#C": "ProductCategory",  
  "#P": "Price"  
}
```

输出：

```
{  
  "Item": {  
    "Price": {  
      "N": "20"  
    },  
    "Title": {  
      "S": "Book 102 Title"  
    },  
    "ProductCategory": {  
      "S": "Book"  
    }  
  }  
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[读取项](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetItem](#)中的。

## list-backups

以下代码示例演示了如何使用 list-backups。

## AWS CLI

示例 1：列出所有现有 DynamoDB 备份

以下 `list-backups` 示例列出了您的所有现有备份。

```
aws dynamodb list-backups
```

输出：

```
{
  "BackupSummaries": [
    {
      "TableName": "MusicCollection",
      "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
      "BackupArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/backup/01234567890123-a1bcd234",
      "BackupName": "MusicCollectionBackup1",
      "BackupCreationDateTime": "2020-02-12T14:41:51.617000-08:00",
      "BackupStatus": "AVAILABLE",
      "BackupType": "USER",
      "BackupSizeBytes": 170
    },
    {
      "TableName": "MusicCollection",
      "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
      "BackupArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/backup/01234567890123-b2abc345",
      "BackupName": "MusicCollectionBackup2",
      "BackupCreationDateTime": "2020-06-26T11:08:35.431000-07:00",
      "BackupStatus": "AVAILABLE",
      "BackupType": "USER",
      "BackupSizeBytes": 400
    }
  ]
}
```

有关更多信息，请参阅《[亚马逊 DynamoDB 开发者指南](#)》中的 [DynamoDB 按需备份和恢复](#)。

## 示例 2：列出特定时间范围内用户创建的备份

以下示例仅列出了用户创建的MusicCollection表的备份（不是由 DynamoDB 自动创建的备份），其创建日期介于 2020 年 1 月 1 日至 2020 年 3 月 1 日之间。

```
aws dynamodb list-backups \  
  --table-name MusicCollection \  
  --time-range-lower-bound 1577836800 \  
  --time-range-upper-bound 1583020800 \  
  --backup-type USER
```

输出：

```
{  
  "BackupSummaries": [  
    {  
      "TableName": "MusicCollection",  
      "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/  
MusicCollection",  
      "BackupArn": "arn:aws:dynamodb:us-west-2:123456789012:table/  
MusicCollection/backup/01234567890123-a1bcd234",  
      "BackupName": "MusicCollectionBackup1",  
      "BackupCreationDateTime": "2020-02-12T14:41:51.617000-08:00",  
      "BackupStatus": "AVAILABLE",  
      "BackupType": "USER",  
      "BackupSizeBytes": 170  
    }  
  ]  
}
```

有关更多信息，请参阅《[亚马逊 DynamoDB 开发者指南](#)》中的 [DynamoDB 按需备份和恢复](#)。

## 示例 3：限制页面大小

以下示例返回所有现有备份的列表，但在每次调用中仅检索一个项目，必要时执行多次调用以获取整个列表。在对大量资源运行列表命令时，限制页面大小非常有用，使用默认页面大小 1000 时，可能会导致“超时”错误。

```
aws dynamodb list-backups \  
  --page-size 1
```



输出：

```
{
  "BackupSummaries": [
    {
      "TableName": "MusicCollection",
      "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
      "BackupArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/backup/01234567890123-a1bcd234",
      "BackupName": "MusicCollectionBackup1",
      "BackupCreationDateTime": "2020-02-12T14:41:51.617000-08:00",
      "BackupStatus": "AVAILABLE",
      "BackupType": "USER",
      "BackupSizeBytes": 170
    },
    {
      "TableName": "MusicCollection",
      "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
      "BackupArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/backup/01234567890123-b2abc345",
      "BackupName": "MusicCollectionBackup2",
      "BackupCreationDateTime": "2020-06-26T11:08:35.431000-07:00",
      "BackupStatus": "AVAILABLE",
      "BackupType": "USER",
      "BackupSizeBytes": 400
    }
  ]
}
```

有关更多信息，请参阅《[亚马逊 DynamoDB 开发者指南](#)》中的 [DynamoDB 按需备份和恢复](#)。

#### 示例 4：限制退回的商品数量

以下示例将返回的商品数量限制为 1。响应包含用于检索下一页结果的 NextToken 值。

```
aws dynamodb list-backups \
  --max-items 1
```

输出：

```
{
  "BackupSummaries": [
    {
      "TableName": "MusicCollection",
      "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
      "BackupArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/backup/01234567890123-a1bcd234",
      "BackupName": "MusicCollectionBackup1",
      "BackupCreationDateTime": "2020-02-12T14:41:51.617000-08:00",
      "BackupStatus": "AVAILABLE",
      "BackupType": "USER",
      "BackupSizeBytes": 170
    }
  ],
  "NextToken":
  "abCDeFGhiJKlMnOPqrSTuvwxYZ1aBCdEFghijK7LM51nOpqRSTuv3WxY3ZabC5dEFGhI2Jk3LmnoPQ6RST9"
}
```

有关更多信息，请参阅《亚马逊 [DynamoDB 开发者指南](#)》中的 [DynamoDB 按需备份和恢复](#)。

#### 示例 5：检索下一页的结果

以下命令将使用先前对 `list-backups` 命令的调用中的 `NextToken` 值来检索另一页结果。由于本例中的响应不包含 `NextToken` 值，因此，我们知道已经到达结果末尾。

```
aws dynamodb list-backups \
  --starting-token
  abCDeFGhiJKlMnOPqrSTuvwxYZ1aBCdEFghijK7LM51nOpqRSTuv3WxY3ZabC5dEFGhI2Jk3LmnoPQ6RST9
```

#### 输出

```
{
  "BackupSummaries": [
    {
      "TableName": "MusicCollection",
      "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
      "BackupArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/backup/01234567890123-b2abc345",
```

```
        "BackupName": "MusicCollectionBackup2",
        "BackupCreationDateTime": "2020-06-26T11:08:35.431000-07:00",
        "BackupStatus": "AVAILABLE",
        "BackupType": "USER",
        "BackupSizeBytes": 400
    }
]
}
```

有关更多信息，请参阅《亚马逊 [DynamoDB 开发者指南](#)》中的 [DynamoDB 按需备份和恢复](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListBackups](#) 中的。

## list-contributor-insights

以下代码示例演示了如何使用 `list-contributor-insights`。

### AWS CLI

示例 1：查看投稿人见解摘要列表

以下 `list-contributor-insights` 示例显示了“贡献者见解”摘要列表。

```
aws dynamodb list-contributor-insights
```

输出：

```
{
  "ContributorInsightsSummaries": [
    {
      "TableName": "MusicCollection",
      "IndexName": "AlbumTitle-index",
      "ContributorInsightsStatus": "ENABLED"
    },
    {
      "TableName": "ProductCatalog",
      "ContributorInsightsStatus": "ENABLED"
    },
    {
      "TableName": "Forum",
      "ContributorInsightsStatus": "ENABLED"
    },
  ],
}
```

```
{
  "TableName": "Reply",
  "ContributorInsightsStatus": "ENABLED"
},
{
  "TableName": "Thread",
  "ContributorInsightsStatus": "ENABLED"
}
]
```

有关更多信息，请参阅《亚马逊 [DynamoDB 开发者指南](#)》中的“使用 CloudWatch 贡献者洞察分析 DynamoDB 的数据访问权限”。

### 示例 2：限制退回的商品数量

以下示例将返回的商品数量限制为 4。响应包含用于检索下一页结果的 NextToken 值。

```
aws dynamodb list-contributor-insights \
  --max-results 4
```

输出：

```
{
  "ContributorInsightsSummaries": [
    {
      "TableName": "MusicCollection",
      "IndexName": "AlbumTitle-index",
      "ContributorInsightsStatus": "ENABLED"
    },
    {
      "TableName": "ProductCatalog",
      "ContributorInsightsStatus": "ENABLED"
    },
    {
      "TableName": "Forum",
      "ContributorInsightsStatus": "ENABLED"
    }
  ],
  "NextToken":
  "abCDeFGhiJKlMnOPqrSTuvwXYZ1aBCdEFghijK7LM51n0pqRSTuv3WxY3ZabC5dEFGhI2Jk3LmnoPQ6RST9"
}
```

有关更多信息，请参阅《亚马逊 [DynamoDB 开发者指南](#)》中的“[使用 CloudWatch 贡献者洞察分析 DynamoDB 的数据访问权限](#)”。

### 示例 3：检索下一页的结果

以下命令将使用先前对 `list-contributor-insights` 命令的调用中的 `NextToken` 值来检索另一页结果。由于本例中的响应不包含 `NextToken` 值，因此，我们知道已经到达结果末尾。

```
aws dynamodb list-contributor-insights \  
  --max-results 4 \  
  --next-token  
  abCDeFGhiJKlmnOPqrSTuvwXYZ1aBCdEFghijK7LM51n0pqRSTuv3WxY3ZabC5dEFGhI2Jk3LmnoPQ6RST9
```

输出：

```
{  
  "ContributorInsightsSummaries": [  
    {  
      "TableName": "Reply",  
      "ContributorInsightsStatus": "ENABLED"  
    },  
    {  
      "TableName": "Thread",  
      "ContributorInsightsStatus": "ENABLED"  
    }  
  ]  
}
```

有关更多信息，请参阅《亚马逊 [DynamoDB 开发者指南](#)》中的“[使用 CloudWatch 贡献者洞察分析 DynamoDB 的数据访问权限](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListContributorInsights](#) 中的。

## list-global-tables

以下代码示例演示了如何使用 `list-global-tables`。

AWS CLI

列出现有 DynamoDB 全局表

以下 `list-global-tables` 示例列出了所有现有的全局表。

```
aws dynamodb list-global-tables
```

输出：

```
{
  "GlobalTables": [
    {
      "GlobalTableName": "MusicCollection",
      "ReplicationGroup": [
        {
          "RegionName": "us-east-2"
        },
        {
          "RegionName": "us-east-1"
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅亚马逊 [DynamoDB 开发者指南中的 DynamoDB 全局表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListGlobalTables](#)中的。

## list-tables

以下代码示例演示了如何使用 list-tables。

AWS CLI

示例 1：列出表

以下list-tables示例列出了与当前 AWS 账户和区域关联的所有表。

```
aws dynamodb list-tables
```

输出：

```
{
  "TableNames": [
    "Forum",
  ]
}
```

```
        "ProductCatalog",
        "Reply",
        "Thread"
    ]
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[列出表名称](#)。

### 示例 2：限制页面大小

以下示例将返回所有现有表的列表，但在每次调用中仅检索一个项，必要时执行多次调用以获取整个列表。在对大量资源运行列表命令时，限制页面大小非常有用，使用默认页面大小 1000 时，可能会导致“超时”错误。

```
aws dynamodb list-tables \
  --page-size 1
```

输出：

```
{
  "TableNames": [
    "Forum",
    "ProductCatalog",
    "Reply",
    "Thread"
  ]
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[列出表名称](#)。

### 示例 3：限制返回项的数量

以下示例将返回项的数量限制为 2。响应包含用于检索下一页结果的 NextToken 值。

```
aws dynamodb list-tables \
  --max-items 2
```

输出：

```
{
```

```

    "TableNames": [
      "Forum",
      "ProductCatalog"
    ],
    "NextToken":
    "abCDeFGhiJKlmnOPqrSTuvwXYZ1aBCdEFghijK7LM51n0pqRSTuv3WxY3ZabC5dEFGhI2Jk3LmnoPQ6RST9"
  }

```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[列出表名称](#)。

#### 示例 4：检索下一页结果

以下命令将使用先前对 `list-tables` 命令的调用中的 `NextToken` 值来检索另一页结果。由于本例中的响应不包含 `NextToken` 值，因此，我们知道已经到达结果末尾。

```

aws dynamodb list-tables \
  --starting-token
  abCDeFGhiJKlmnOPqrSTuvwXYZ1aBCdEFghijK7LM51n0pqRSTuv3WxY3ZabC5dEFGhI2Jk3LmnoPQ6RST9

```

输出：

```

{
  "TableNames": [
    "Reply",
    "Thread"
  ]
}

```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[列出表名称](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTables](#)中的。

## list-tags-of-resource

以下代码示例演示了如何使用 `list-tags-of-resource`。

### AWS CLI

#### 示例 1：列出 DynamoDB 资源的标签

以下 `list-tags-of-resource` 示例显示了 `MusicCollection` 表的标签。



```
aws dynamodb list-tags-of-resource \  
  --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "Owner",  
      "Value": "blueTeam"  
    },  
    {  
      "Key": "Environment",  
      "Value": "Production"  
    }  
  ]  
}
```

有关更多信息，请参阅亚马逊 [Dynamo DB 开发者指南](#) 中的 [为 DynamoDB 添加标签](#)。

示例 2：限制返回的标签数量

以下示例将返回的标签数量限制为 1。响应包含用于检索下一页结果的 NextToken 值。

```
aws dynamodb list-tags-of-resource \  
  --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection \  
  --max-items 1
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "Owner",  
      "Value": "blueTeam"  
    }  
  ],  
  "NextToken":  
  "abCDeFGhiJKlmnOPqrSTuvwXYZ1aBCdEFghijK7LM51n0pqRSTuv3WxY3ZabC5dEFghI2Jk3LmnoPQ6RST9"  
}
```

有关更多信息，请参阅亚马逊 Dynamo [DB 开发者指南中的为 DynamoDB 添加标签](#)。

### 示例 3：检索下一页的结果

以下命令将使用先前对 `list-tags-of-resource` 命令的调用中的 `NextToken` 值来检索另一页结果。由于本例中的响应不包含 `NextToken` 值，因此，我们知道已经到达结果末尾。

```
aws dynamodb list-tags-of-resource \  
  --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection \  
  --starting-token  
abCDeFGhiJKlMnOPqrSTuvwxYZ1aBCdEFghijK7LM51nOpqRSTuv3WxY3ZabC5dEFGhI2Jk3LmnoPQ6RST9
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "Environment",  
      "Value": "Production"  
    }  
  ]  
}
```

有关更多信息，请参阅亚马逊 Dynamo [DB 开发者指南中的为 DynamoDB 添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsOfResource](#)中的。

## put-item

以下代码示例演示了如何使用 `put-item`。

### AWS CLI

#### 示例 1：向表中添加项

以下`put-item`示例向`MusicCollection`表中添加了一个新项目。

```
aws dynamodb put-item \  
  --table-name MusicCollection \  
  --item file://item.json \  
  --return-consumed-capacity TOTAL \  
  --
```

```
--return-item-collection-metrics SIZE
```

item.json 的内容：

```
{
  "Artist": {"S": "No One You Know"},
  "SongTitle": {"S": "Call Me Today"},
  "AlbumTitle": {"S": "Greatest Hits"}
}
```

输出：

```
{
  "ConsumedCapacity": {
    "TableName": "MusicCollection",
    "CapacityUnits": 1.0
  },
  "ItemCollectionMetrics": {
    "ItemCollectionKey": {
      "Artist": {
        "S": "No One You Know"
      }
    },
    "SizeEstimateRangeGB": [
      0.0,
      1.0
    ]
  }
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[写入项](#)。

示例 2：有条件地覆盖表中的项

仅当 MusicCollection 表中的现有项具有值为 Greatest Hits 的 AlbumTitle 属性时，以下 put-item 示例才会覆盖该项。该命令将返回该项先前的值。

```
aws dynamodb put-item \
  --table-name MusicCollection \
  --item file://item.json \
  --condition-expression "#A = :A" \
  --expression-attribute-names file://names.json \
```

```
--expression-attribute-values file://values.json \  
--return-values ALL_OLD
```

item.json 的内容 :

```
{  
  "Artist": {"S": "No One You Know"},  
  "SongTitle": {"S": "Call Me Today"},  
  "AlbumTitle": {"S": "Somewhat Famous"}  
}
```

names.json 的内容 :

```
{  
  "#A": "AlbumTitle"  
}
```

values.json 的内容 :

```
{  
  ":A": {"S": "Greatest Hits"}  
}
```

输出 :

```
{  
  "Attributes": {  
    "AlbumTitle": {  
      "S": "Greatest Hits"  
    },  
    "Artist": {  
      "S": "No One You Know"  
    },  
    "SongTitle": {  
      "S": "Call Me Today"  
    }  
  }  
}
```

如果键已存在，您应看到以下输出：

```
A client error (ConditionalCheckFailedException) occurred when calling the PutItem operation: The conditional request failed.
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[写入项](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutItem](#)中的。

## query

以下代码示例演示了如何使用 query。

### AWS CLI

#### 示例 1：查询表

以下 query 示例将查询 MusicCollection 表中的项。该表具有 hash-and-range 主键（Artist和SongTitle），但此查询仅指定哈希键值。它返回名为“No One You Know”的艺术家的歌名。

```
aws dynamodb query \  
  --table-name MusicCollection \  
  --projection-expression "SongTitle" \  
  --key-condition-expression "Artist = :v1" \  
  --expression-attribute-values file://expression-attributes.json \  
  --return-consumed-capacity TOTAL
```

expression-attributes.json 的内容：

```
{  
  ":v1": {"S": "No One You Know"}  
}
```

输出：

```
{  
  "Items": [  
    {  
      "SongTitle": {  
        "S": "Call Me Today"  
      },  
      "SongTitle": {
```

```

        "S": "Scared of My Shadow"
      }
    }
  ],
  "Count": 2,
  "ScannedCount": 2,
  "ConsumedCapacity": {
    "TableName": "MusicCollection",
    "CapacityUnits": 0.5
  }
}

```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[使用 DynamoDB 中的查询](#)。

### 示例 2：使用强一致性读取查询表并按降序遍历索引

以下示例将执行与第一个示例相同的查询，但返回结果的顺序相反，并且使用强一致性读取。

```

aws dynamodb query \
  --table-name MusicCollection \
  --projection-expression "SongTitle" \
  --key-condition-expression "Artist = :v1" \
  --expression-attribute-values file://expression-attributes.json \
  --consistent-read \
  --no-scan-index-forward \
  --return-consumed-capacity TOTAL

```

expression-attributes.json 的内容：

```

{
  ":v1": {"S": "No One You Know"}
}

```

输出：

```

{
  "Items": [
    {
      "SongTitle": {
        "S": "Scared of My Shadow"
      }
    },

```

```

    {
      "SongTitle": {
        "S": "Call Me Today"
      }
    }
  ],
  "Count": 2,
  "ScannedCount": 2,
  "ConsumedCapacity": {
    "TableName": "MusicCollection",
    "CapacityUnits": 1.0
  }
}

```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[使用 DynamoDB 中的查询](#)。

### 示例 3：筛选出特定结果

以下示例将查询 MusicCollection，但不包括 AlbumTitle 属性中含特定值的结果。请注意，这不会影响 ScannedCount 或 ConsumedCapacity，因为筛选器在读取项之后应用。

```

aws dynamodb query \
  --table-name MusicCollection \
  --key-condition-expression "#n1 = :v1" \
  --filter-expression "NOT (#n2 IN (:v2, :v3))" \
  --expression-attribute-names file://names.json \
  --expression-attribute-values file://values.json \
  --return-consumed-capacity TOTAL

```

values.json 的内容：

```

{
  ":v1": {"S": "No One You Know"},
  ":v2": {"S": "Blue Sky Blues"},
  ":v3": {"S": "Greatest Hits"}
}

```

names.json 的内容：

```

{
  "#n1": "Artist",
  "#n2": "AlbumTitle"
}

```

```
}

```

输出：

```
{
  "Items": [
    {
      "AlbumTitle": {
        "S": "Somewhat Famous"
      },
      "Artist": {
        "S": "No One You Know"
      },
      "SongTitle": {
        "S": "Call Me Today"
      }
    }
  ],
  "Count": 1,
  "ScannedCount": 2,
  "ConsumedCapacity": {
    "TableName": "MusicCollection",
    "CapacityUnits": 0.5
  }
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[使用 DynamoDB 中的查询](#)。

示例 4：仅检索项计数

以下示例将检索与查询匹配的项计数，但不检索任何项本身。

```
aws dynamodb query \
  --table-name MusicCollection \
  --select COUNT \
  --key-condition-expression "Artist = :v1" \
  --expression-attribute-values file://expression-attributes.json
```

expression-attributes.json 的内容：

```
{
  ":v1": {"S": "No One You Know"}
```



```
}
```

输出：

```
{
  "Count": 2,
  "ScannedCount": 2,
  "ConsumedCapacity": null
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[使用 DynamoDB 中的查询](#)。

### 示例 5：查询索引

以下示例将查询本地二级索引 AlbumTitleIndex。该查询返回基表中已投影到本地二级索引的所有属性。请注意，查询本地二级索引或全局二级索引时，您还必须使用 table-name 参数提供基表的名称。

```
aws dynamodb query \
  --table-name MusicCollection \
  --index-name AlbumTitleIndex \
  --key-condition-expression "Artist = :v1" \
  --expression-attribute-values file://expression-attributes.json \
  --select ALL_PROJECTED_ATTRIBUTES \
  --return-consumed-capacity INDEXES
```

expression-attributes.json 的内容：

```
{
  ":v1": {"S": "No One You Know"}
}
```

输出：

```
{
  "Items": [
    {
      "AlbumTitle": {
        "S": "Blue Sky Blues"
      },
      "Artist": {
        "S": "No One You Know"
      }
    }
  ]
}
```

```
    },
    "SongTitle": {
      "S": "Scared of My Shadow"
    }
  },
  {
    "AlbumTitle": {
      "S": "Somewhat Famous"
    },
    "Artist": {
      "S": "No One You Know"
    },
    "SongTitle": {
      "S": "Call Me Today"
    }
  }
],
"Count": 2,
"ScannedCount": 2,
"ConsumedCapacity": {
  "TableName": "MusicCollection",
  "CapacityUnits": 0.5,
  "Table": {
    "CapacityUnits": 0.0
  },
  "LocalSecondaryIndexes": {
    "AlbumTitleIndex": {
      "CapacityUnits": 0.5
    }
  }
}
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[使用 DynamoDB 中的查询](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的[Query](#)。

## restore-table-from-backup

以下代码示例演示了如何使用 `restore-table-from-backup`。

### AWS CLI

从现有备份中恢复 DynamoDB 表

以下 `restore-table-from-backup` 示例从现有备份中恢复指定的表。

```
aws dynamodb restore-table-from-backup \  
  --target-table-name MusicCollection \  
  --backup-arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection/  
  backup/01576616366715-b4e58d3a
```

输出：

```
{  
  "TableDescription": {  
    "AttributeDefinitions": [  
      {  
        "AttributeName": "Artist",  
        "AttributeType": "S"  
      },  
      {  
        "AttributeName": "SongTitle",  
        "AttributeType": "S"  
      }  
    ],  
    "TableName": "MusicCollection2",  
    "KeySchema": [  
      {  
        "AttributeName": "Artist",  
        "KeyType": "HASH"  
      },  
      {  
        "AttributeName": "SongTitle",  
        "KeyType": "RANGE"  
      }  
    ],  
    "TableStatus": "CREATING",  
    "CreationDateTime": 1576618274.326,  
    "ProvisionedThroughput": {  
      "NumberOfDecreasesToday": 0,  
      "ReadCapacityUnits": 5,  
      "WriteCapacityUnits": 5  
    },  
    "TableSizeBytes": 0,  
    "ItemCount": 0,  
    "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/  
    MusicCollection2",
```

```
    "TableId": "114865c9-5ef3-496c-b4d1-c4cbdd2d44fb",
    "BillingModeSummary": {
      "BillingMode": "PROVISIONED"
    },
    "RestoreSummary": {
      "SourceBackupArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/backup/01576616366715-b4e58d3a",
      "SourceTableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
      "RestoreDateTime": 1576616366.715,
      "RestoreInProgress": true
    }
  }
}
```

有关更多信息，请参阅《亚马逊 [DynamoDB 开发者指南](#)》中的 [DynamoDB 按需备份和恢复](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RestoreTableFromBackup](#)中的。

## restore-table-to-point-in-time

以下代码示例演示了如何使用 `restore-table-to-point-in-time`。

### AWS CLI

将 DynamoDB 表还原到某个时间点

以下 `restore-table-to-point-in-time` 示例将 `MusicCollection` 表恢复到指定的时间点。

```
aws dynamodb restore-table-to-point-in-time \
  --source-table-name MusicCollection \
  --target-table-name MusicCollectionRestore \
  --restore-date-time 1576622404.0
```

输出：

```
{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "Artist",
        "AttributeType": "S"
      }
    ]
  }
}
```

```

    },
    {
      "AttributeName": "SongTitle",
      "AttributeType": "S"
    }
  ],
  "TableName": "MusicCollectionRestore",
  "KeySchema": [
    {
      "AttributeName": "Artist",
      "KeyType": "HASH"
    },
    {
      "AttributeName": "SongTitle",
      "KeyType": "RANGE"
    }
  ],
  "TableStatus": "CREATING",
  "CreationDateTime": 1576623311.86,
  "ProvisionedThroughput": {
    "NumberOfDecreasesToday": 0,
    "ReadCapacityUnits": 5,
    "WriteCapacityUnits": 5
  },
  "TableSizeBytes": 0,
  "ItemCount": 0,
  "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollectionRestore",
  "TableId": "befd9e0e-1843-4dc6-a147-d6d00e85cb1f",
  "BillingModeSummary": {
    "BillingMode": "PROVISIONED"
  },
  "RestoreSummary": {
    "SourceTableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
    "RestoreDateTime": 1576622404.0,
    "RestoreInProgress": true
  }
}
}
}

```

有关更多信息，请参阅《亚马逊 DynamoDB 开发者指南》中的 [DynamoDB 时间点恢复](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RestoreTableToPointInTime](#) 中的。

## scan

以下代码示例演示了如何使用 scan。

### AWS CLI

#### 扫描表

以下 scan 示例将扫描整个 MusicCollection 表，然后将结果范围缩小到艺术家“No One You Know”的歌曲。对于每个项，仅返回专辑名称和歌曲名称。

```
aws dynamodb scan \  
  --table-name MusicCollection \  
  --filter-expression "Artist = :a" \  
  --projection-expression "#ST, #AT" \  
  --expression-attribute-names file://expression-attribute-names.json \  
  --expression-attribute-values file://expression-attribute-values.json
```

expression-attribute-names.json 的内容：

```
{  
  "#ST": "SongTitle",  
  "#AT": "AlbumTitle"  
}
```

expression-attribute-values.json 的内容：

```
{  
  ":a": {"S": "No One You Know"}  
}
```

输出：

```
{  
  "Count": 2,  
  "Items": [  
    {  
      "SongTitle": {  
        "S": "Call Me Today"  
      },  
      "AlbumTitle": {
```

```
        "S": "Somewhat Famous"
      }
    },
    {
      "SongTitle": {
        "S": "Scared of My Shadow"
      },
      "AlbumTitle": {
        "S": "Blue Sky Blues"
      }
    }
  ],
  "ScannedCount": 3,
  "ConsumedCapacity": null
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[使用 DynamoDB 中的扫描](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的[Scan](#)。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

向 DynamoDB 资源添加标签

以下tag-resource示例向表中添加了标签键/值对。MusicCollection

```
aws dynamodb tag-resource \
  --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection \
  --tags Key=Owner,Value=blueTeam
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 Dynamo [DB 开发者指南中的为 DynamoDB 添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## transact-get-items

以下代码示例演示了如何使用 transact-get-items。

## AWS CLI

从一个或多个表中以原子方式检索多个项目

以下transact-get-items示例以原子方式检索多个项目。

```
aws dynamodb transact-get-items \  
  --transact-items file://transact-items.json \  
  --return-consumed-capacity TOTAL
```

transact-items.json 的内容：

```
[  
  {  
    "Get": {  
      "Key": {  
        "Artist": {"S": "Acme Band"},  
        "SongTitle": {"S": "Happy Day"}  
      },  
      "TableName": "MusicCollection"  
    }  
  },  
  {  
    "Get": {  
      "Key": {  
        "Artist": {"S": "No One You Know"},  
        "SongTitle": {"S": "Call Me Today"}  
      },  
      "TableName": "MusicCollection"  
    }  
  }  
]
```

输出：

```
{  
  "ConsumedCapacity": [  
    {  
      "TableName": "MusicCollection",  
      "CapacityUnits": 4.0,  
      "ReadCapacityUnits": 4.0  
    }  
  ]  
}
```



```
],
  "Responses": [
    {
      "Item": {
        "AlbumTitle": {
          "S": "Songs About Life"
        },
        "Artist": {
          "S": "Acme Band"
        },
        "SongTitle": {
          "S": "Happy Day"
        }
      }
    },
    {
      "Item": {
        "AlbumTitle": {
          "S": "Somewhat Famous"
        },
        "Artist": {
          "S": "No One You Know"
        },
        "SongTitle": {
          "S": "Call Me Today"
        }
      }
    }
  ]
}
```

有关更多信息，请参阅亚马逊 [DynamoDB 开发者指南中的使用 DynamoDB 事务管理复杂的工作流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TransactGetItems](#)中的。

## transact-write-items

以下代码示例演示了如何使用 transact-write-items。

### AWS CLI

示例 1：以原子方式将项目写入一个或多个表

以下`transact-write-items`示例更新一个项目并删除另一个项目。如果任一操作失败，或者任一项目包含`Rating`属性，则操作将失败。

```
aws dynamodb transact-write-items \  
  --transact-items file://transact-items.json \  
  --return-consumed-capacity TOTAL \  
  --return-item-collection-metrics SIZE
```

`transact-items.json`文件内容：

```
[  
  {  
    "Update": {  
      "Key": {  
        "Artist": {"S": "Acme Band"},  
        "SongTitle": {"S": "Happy Day"}  
      },  
      "UpdateExpression": "SET AlbumTitle = :newval",  
      "ExpressionAttributeValues": {  
        ":newval": {"S": "Updated Album Title"}  
      },  
      "TableName": "MusicCollection",  
      "ConditionExpression": "attribute_not_exists(Rating)"  
    },  
    {  
      "Delete": {  
        "Key": {  
          "Artist": {"S": "No One You Know"},  
          "SongTitle": {"S": "Call Me Today"}  
        },  
        "TableName": "MusicCollection",  
        "ConditionExpression": "attribute_not_exists(Rating)"  
      }  
    }  
  }  
]
```

输出：

```
{  
  "ConsumedCapacity": [  
    {
```

```

        "TableName": "MusicCollection",
        "CapacityUnits": 10.0,
        "WriteCapacityUnits": 10.0
    }
],
"ItemCollectionMetrics": {
    "MusicCollection": [
        {
            "ItemCollectionKey": {
                "Artist": {
                    "S": "No One You Know"
                }
            },
            "SizeEstimateRangeGB": [
                0.0,
                1.0
            ]
        },
        {
            "ItemCollectionKey": {
                "Artist": {
                    "S": "Acme Band"
                }
            },
            "SizeEstimateRangeGB": [
                0.0,
                1.0
            ]
        }
    ]
}
}

```

有关更多信息，请参阅亚马逊 [DynamoDB 开发者指南中的使用 DynamoDB 事务管理复杂的工作流程](#)。

#### 示例 2：使用客户端请求令牌以原子方式写入项目

以下命令使用客户端请求令牌来调用 `transact-write-items idempotent`，这意味着多个调用与单个调用具有相同的效果。

```

aws dynamodb transact-write-items \
  --transact-items file://transact-items.json \

```

```
--client-request-token abc123
```

transact-items.json文件内容：

```
[
  {
    "Update": {
      "Key": {
        "Artist": {"S": "Acme Band"},
        "SongTitle": {"S": "Happy Day"}
      },
      "UpdateExpression": "SET AlbumTitle = :newval",
      "ExpressionAttributeValues": {
        ":newval": {"S": "Updated Album Title"}
      },
      "TableName": "MusicCollection",
      "ConditionExpression": "attribute_not_exists(Rating)"
    },
  },
  {
    "Delete": {
      "Key": {
        "Artist": {"S": "No One You Know"},
        "SongTitle": {"S": "Call Me Today"}
      },
      "TableName": "MusicCollection",
      "ConditionExpression": "attribute_not_exists(Rating)"
    }
  }
]
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 [DynamoDB 开发者指南中的使用 DynamoDB 事务管理复杂的工作流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TransactWriteItems](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

## AWS CLI

从 DynamoDB 资源中移除标签

以下 `untag-resource` 示例 `Owner` 从 `MusicCollection` 表中删除带有密钥的标签。

```
aws dynamodb untag-resource \  
  --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection \  
  --tag-keys Owner
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 Dynamo [DB 开发者指南中的为 DynamoDB 添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-continuous-backups

以下代码示例演示了如何使用 `update-continuous-backups`。

### AWS CLI

更新 DynamoDB 表的连续备份设置

以下 `update-continuous-backups` 示例启用了 `MusicCollection` 表的 `point-in-time` 恢复。

```
aws dynamodb update-continuous-backups \  
  --table-name MusicCollection \  
  --point-in-time-recovery-specification PointInTimeRecoveryEnabled=true
```

输出：

```
{  
  "ContinuousBackupsDescription": {  
    "ContinuousBackupsStatus": "ENABLED",  
    "PointInTimeRecoveryDescription": {  
      "PointInTimeRecoveryStatus": "ENABLED",  
      "EarliestRestorableDateTime": 1576622404.0,  
      "LatestRestorableDateTime": 1576622404.0  
    }  
  }  
}
```

有关更多信息，请参阅《亚马逊 DynamoDB 开发者指南》中的 [DynamoDB 时间点恢复](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateContinuousBackups](#)中的。

## update-contributor-insights

以下代码示例演示了如何使用 update-contributor-insights。

### AWS CLI

在表格上启用“贡献者见解”

以下update-contributor-insights示例在MusicCollection表格和AlbumTitle-index全局二级索引上启用“贡献者见解”。

```
aws dynamodb update-contributor-insights \  
  --table-name MusicCollection \  
  --index-name AlbumTitle-index \  
  --contributor-insights-action ENABLE
```

输出：

```
{  
  "TableName": "MusicCollection",  
  "IndexName": "AlbumTitle-index",  
  "ContributorInsightsStatus": "ENABLING"  
}
```

有关更多信息，请参阅《亚马逊 [DynamoDB 开发者指南](#)》中的“[使用 CloudWatch 贡献者洞察分析 DynamoDB 的数据访问权限](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateContributorInsights](#)中的。

## update-global-table-settings

以下代码示例演示了如何使用 update-global-table-settings。

### AWS CLI

更新 DynamoDB 全局表上的预配置写入容量设置

以下update-global-table-settings示例将MusicCollection全局表的预配置写入容量设置为 15。

```
aws dynamodb update-global-table-settings \  
  --global-table-name MusicCollection \  
  --global-table-provisioned-write-capacity-units 15
```

输出：

```
{  
  "GlobalTableName": "MusicCollection",  
  "ReplicaSettings": [  
    {  
      "RegionName": "eu-west-1",  
      "ReplicaStatus": "UPDATING",  
      "ReplicaProvisionedReadCapacityUnits": 10,  
      "ReplicaProvisionedReadCapacityAutoScalingSettings": {  
        "AutoScalingDisabled": true  
      },  
      "ReplicaProvisionedWriteCapacityUnits": 10,  
      "ReplicaProvisionedWriteCapacityAutoScalingSettings": {  
        "AutoScalingDisabled": true  
      }  
    },  
    {  
      "RegionName": "us-east-1",  
      "ReplicaStatus": "UPDATING",  
      "ReplicaProvisionedReadCapacityUnits": 10,  
      "ReplicaProvisionedReadCapacityAutoScalingSettings": {  
        "AutoScalingDisabled": true  
      },  
      "ReplicaProvisionedWriteCapacityUnits": 10,  
      "ReplicaProvisionedWriteCapacityAutoScalingSettings": {  
        "AutoScalingDisabled": true  
      }  
    },  
    {  
      "RegionName": "us-east-2",  
      "ReplicaStatus": "UPDATING",  
      "ReplicaProvisionedReadCapacityUnits": 10,  
      "ReplicaProvisionedReadCapacityAutoScalingSettings": {  
        "AutoScalingDisabled": true  
      },  
      "ReplicaProvisionedWriteCapacityUnits": 10,  
      "ReplicaProvisionedWriteCapacityAutoScalingSettings": {  
        "AutoScalingDisabled": true  
      }  
    }  
  ]  
}
```

```

    }
  }
]
}

```

有关更多信息，请参阅亚马逊 [DynamoDB 开发者指南中的 DynamoDB 全局表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateGlobalTableSettings](#) 中的。

## update-global-table

以下代码示例演示了如何使用 `update-global-table`。

### AWS CLI

#### 更新 DynamoDB 全局表

以下 `update-global-table` 示例将指定区域中的副本添加到 `MusicCollection` 全局表。

```

aws dynamodb update-global-table \
  --global-table-name MusicCollection \
  --replica-updates Create={RegionName=eu-west-1}

```

输出：

```

{
  "GlobalTableDescription": {
    "ReplicationGroup": [
      {
        "RegionName": "eu-west-1"
      },
      {
        "RegionName": "us-east-2"
      },
      {
        "RegionName": "us-east-1"
      }
    ],
    "GlobalTableArn": "arn:aws:dynamodb::123456789012:global-table/MusicCollection",
    "CreationDateTime": 1576625818.532,
    "GlobalTableStatus": "ACTIVE",
  }
}

```



```
    "GlobalTableName": "MusicCollection"
  }
}
```

有关更多信息，请参阅亚马逊 [DynamoDB 开发者指南中的 DynamoDB 全局表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateGlobalTable](#) 中的。

## update-item

以下代码示例演示了如何使用 update-item。

### AWS CLI

#### 示例 1：更新表中的项

下面的 update-item 示例更新 MusicCollection 表的项目。它会添加一个新属性 (Year) 并修改 AlbumTitle 属性。响应中会返回更新后显示的项中的所有属性。

```
aws dynamodb update-item \
  --table-name MusicCollection \
  --key file://key.json \
  --update-expression "SET #Y = :y, #AT = :t" \
  --expression-attribute-names file://expression-attribute-names.json \
  --expression-attribute-values file://expression-attribute-values.json \
  --return-values ALL_NEW \
  --return-consumed-capacity TOTAL \
  --return-item-collection-metrics SIZE
```

key.json 的内容：

```
{
  "Artist": {"S": "Acme Band"},
  "SongTitle": {"S": "Happy Day"}
}
```

expression-attribute-names.json 的内容：

```
{
  "#Y": "Year", "#AT": "AlbumTitle"
}
```

expression-attribute-values.json 的内容：

```
{
  "y":{"N": "2015"},
  "t":{"S": "Louder Than Ever"}
}
```

输出：

```
{
  "Attributes": {
    "AlbumTitle": {
      "S": "Louder Than Ever"
    },
    "Awards": {
      "N": "10"
    },
    "Artist": {
      "S": "Acme Band"
    },
    "Year": {
      "N": "2015"
    },
    "SongTitle": {
      "S": "Happy Day"
    }
  },
  "ConsumedCapacity": {
    "TableName": "MusicCollection",
    "CapacityUnits": 3.0
  },
  "ItemCollectionMetrics": {
    "ItemCollectionKey": {
      "Artist": {
        "S": "Acme Band"
      }
    },
    "SizeEstimateRangeGB": [
      0.0,
      1.0
    ]
  }
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[写入项](#)。

## 示例 2：有条件地更新项

以下示例将更新 MusicCollection 表中的项，但前提是现有项还没有 Year 属性。

```
aws dynamodb update-item \  
  --table-name MusicCollection \  
  --key file://key.json \  
  --update-expression "SET #Y = :y, #AT = :t" \  
  --expression-attribute-names file://expression-attribute-names.json \  
  --expression-attribute-values file://expression-attribute-values.json \  
  --condition-expression "attribute_not_exists(#Y)"
```

key.json 的内容：

```
{  
  "Artist": {"S": "Acme Band"},  
  "SongTitle": {"S": "Happy Day"}  
}
```

expression-attribute-names.json 的内容：

```
{  
  "#Y": "Year",  
  "#AT": "AlbumTitle"  
}
```

expression-attribute-values.json 的内容：

```
{  
  ":y": {"N": "2015"},  
  ":t": {"S": "Louder Than Ever"}  
}
```

如果该项已有 Year 属性，DynamoDB 会返回以下输出。

```
An error occurred (ConditionalCheckFailedException) when calling the UpdateItem  
operation: The conditional request failed
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[写入项](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateItem](#)中的。

## update-table-replica-auto-scaling

以下代码示例演示了如何使用 update-table-replica-auto-scaling。

### AWS CLI

更新跨全局表副本的 auto 缩放设置

以下update-table-replica-auto-scaling示例更新了指定全局表副本间的写入容量 auto Scaling 设置。

```
aws dynamodb update-table-replica-auto-scaling \  
  --table-name MusicCollection \  
  --provisioned-write-capacity-auto-scaling-update file://auto-scaling-policy.json
```

auto-scaling-policy.json 的内容：

```
{  
  "MinimumUnits": 10,  
  "MaximumUnits": 100,  
  "AutoScalingDisabled": false,  
  "ScalingPolicyUpdate": {  
    "PolicyName": "DynamoDBWriteCapacityUtilization:table/MusicCollection",  
    "TargetTrackingScalingPolicyConfiguration": {  
      "TargetValue": 80  
    }  
  }  
}
```

输出：

```
{  
  "TableAutoScalingDescription": {  
    "TableName": "MusicCollection",  
    "TableStatus": "ACTIVE",  
    "Replicas": [  
      {  
        "RegionName": "eu-central-1",  
        "GlobalSecondaryIndexes": [],  
        "ReplicaProvisionedReadCapacityAutoScalingSettings": {
```

```
        "MinimumUnits": 5,
        "MaximumUnits": 40000,
        "AutoScalingRoleArn": "arn:aws:iam::123456789012:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable",
        "ScalingPolicies": [
            {
                "PolicyName": "DynamoDBReadCapacityUtilization:table/
MusicCollection",
                "TargetTrackingScalingPolicyConfiguration": {
                    "TargetValue": 70.0
                }
            }
        ],
    },
    "ReplicaProvisionedWriteCapacityAutoScalingSettings": {
        "MinimumUnits": 10,
        "MaximumUnits": 100,
        "AutoScalingRoleArn": "arn:aws:iam::123456789012:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable",
        "ScalingPolicies": [
            {
                "PolicyName": "DynamoDBWriteCapacityUtilization:table/
MusicCollection",
                "TargetTrackingScalingPolicyConfiguration": {
                    "TargetValue": 80.0
                }
            }
        ]
    },
    "ReplicaStatus": "ACTIVE"
},
{
    "RegionName": "us-east-1",
    "GlobalSecondaryIndexes": [],
    "ReplicaProvisionedReadCapacityAutoScalingSettings": {
        "MinimumUnits": 5,
        "MaximumUnits": 40000,
        "AutoScalingRoleArn": "arn:aws:iam::123456789012:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable",
        "ScalingPolicies": [
            {
```

```

        "PolicyName": "DynamoDBReadCapacityUtilization:table/
MusicCollection",
        "TargetTrackingScalingPolicyConfiguration": {
            "TargetValue": 70.0
        }
    }
],
},
"ReplicaProvisionedWriteCapacityAutoScalingSettings": {
    "MinimumUnits": 10,
    "MaximumUnits": 100,
    "AutoScalingRoleArn": "arn:aws:iam::123456789012:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable",
    "ScalingPolicies": [
        {
            "PolicyName": "DynamoDBWriteCapacityUtilization:table/
MusicCollection",
            "TargetTrackingScalingPolicyConfiguration": {
                "TargetValue": 80.0
            }
        }
    ]
},
"ReplicaStatus": "ACTIVE"
},
{
    "RegionName": "us-east-2",
    "GlobalSecondaryIndexes": [],
    "ReplicaProvisionedReadCapacityAutoScalingSettings": {
        "MinimumUnits": 5,
        "MaximumUnits": 40000,
        "AutoScalingRoleArn": "arn:aws:iam::123456789012:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable",
        "ScalingPolicies": [
            {
                "PolicyName": "DynamoDBReadCapacityUtilization:table/
MusicCollection",
                "TargetTrackingScalingPolicyConfiguration": {
                    "TargetValue": 70.0
                }
            }
        ]
    ]
}

```

```

    },
    "ReplicaProvisionedWriteCapacityAutoScalingSettings": {
      "MinimumUnits": 10,
      "MaximumUnits": 100,
      "AutoScalingRoleArn": "arn:aws:iam::123456789012:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable",
      "ScalingPolicies": [
        {
          "PolicyName": "DynamoDBWriteCapacityUtilization:table/
MusicCollection",
          "TargetTrackingScalingPolicyConfiguration": {
            "TargetValue": 80.0
          }
        }
      ]
    },
    "ReplicaStatus": "ACTIVE"
  }
]
}
}
}

```

有关更多信息，请参阅亚马逊 [DynamoDB 开发者指南中的 DynamoDB 全局表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateTableReplicaAutoScaling](#) 中的。

## update-table

以下代码示例演示了如何使用 update-table。

### AWS CLI

#### 示例 1：修改表的计费模式

以下 update-table 示例增加了 MusicCollection 表上的预置读取和写入容量。

```

aws dynamodb update-table \
  --table-name MusicCollection \
  --billing-mode PROVISIONED \
  --provisioned-throughput ReadCapacityUnits=15,WriteCapacityUnits=10

```

输出：

```
{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "AlbumTitle",
        "AttributeType": "S"
      },
      {
        "AttributeName": "Artist",
        "AttributeType": "S"
      },
      {
        "AttributeName": "SongTitle",
        "AttributeType": "S"
      }
    ],
    "TableName": "MusicCollection",
    "KeySchema": [
      {
        "AttributeName": "Artist",
        "KeyType": "HASH"
      },
      {
        "AttributeName": "SongTitle",
        "KeyType": "RANGE"
      }
    ],
    "TableStatus": "UPDATING",
    "CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
    "ProvisionedThroughput": {
      "LastIncreaseDateTime": "2020-07-28T13:18:18.921000-07:00",
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 15,
      "WriteCapacityUnits": 10
    },
    "TableSizeBytes": 182,
    "ItemCount": 2,
    "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection",
    "TableId": "abcd0123-01ab-23cd-0123-abcdef123456",
    "BillingModeSummary": {
      "BillingMode": "PROVISIONED",
      "LastUpdateToPayPerRequestDateTime": "2020-07-28T13:14:48.366000-07:00"
    }
  }
}
```



```
}  
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[更新表](#)。

### 示例 2：创建全局二级索引

下面的示例在 MusicCollection 表上添加了全局二级索引。

```
aws dynamodb update-table \  
  --table-name MusicCollection \  
  --attribute-definitions AttributeName=AlbumTitle,AttributeType=S \  
  --global-secondary-index-updates file://gsi-updates.json
```

gsi-updates.json 的内容：

```
[  
  {  
    "Create": {  
      "IndexName": "AlbumTitle-index",  
      "KeySchema": [  
        {  
          "AttributeName": "AlbumTitle",  
          "KeyType": "HASH"  
        }  
      ],  
      "ProvisionedThroughput": {  
        "ReadCapacityUnits": 10,  
        "WriteCapacityUnits": 10  
      },  
      "Projection": {  
        "ProjectionType": "ALL"  
      }  
    }  
  }  
]
```

输出：

```
{  
  "TableDescription": {  
    "AttributeDefinitions": [  
      {
```

```
        "AttributeName": "AlbumTitle",
        "AttributeType": "S"
    },
    {
        "AttributeName": "Artist",
        "AttributeType": "S"
    },
    {
        "AttributeName": "SongTitle",
        "AttributeType": "S"
    }
],
"TableName": "MusicCollection",
"KeySchema": [
    {
        "AttributeName": "Artist",
        "KeyType": "HASH"
    },
    {
        "AttributeName": "SongTitle",
        "KeyType": "RANGE"
    }
],
"TableStatus": "UPDATING",
"CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
"ProvisionedThroughput": {
    "LastIncreaseDateTime": "2020-07-28T12:59:17.537000-07:00",
    "NumberOfDecreasesToday": 0,
    "ReadCapacityUnits": 15,
    "WriteCapacityUnits": 10
},
"TableSizeBytes": 182,
"ItemCount": 2,
"TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection",
"TableId": "abcd0123-01ab-23cd-0123-abcdef123456",
"BillingModeSummary": {
    "BillingMode": "PROVISIONED",
    "LastUpdateToPayPerRequestDateTime": "2020-07-28T13:14:48.366000-07:00"
},
"GlobalSecondaryIndexes": [
    {
        "IndexName": "AlbumTitle-index",
        "KeySchema": [
            {
```

```

        "AttributeName": "AlbumTitle",
        "KeyType": "HASH"
      }
    ],
    "Projection": {
      "ProjectionType": "ALL"
    },
    "IndexStatus": "CREATING",
    "Backfilling": false,
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 10,
      "WriteCapacityUnits": 10
    },
    "IndexSizeBytes": 0,
    "ItemCount": 0,
    "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitle-index"
  }
]
}
}

```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[更新表](#)。

### 示例 3：在表上启用 DynamoDB Streams

以下命令在 MusicCollection 表上启用 DynamoDB Streams。

```

aws dynamodb update-table \
  --table-name MusicCollection \
  --stream-specification StreamEnabled=true,StreamViewType=NEW_IMAGE

```

输出：

```

{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "AlbumTitle",
        "AttributeType": "S"
      },
      {

```

```
        "AttributeName": "Artist",
        "AttributeType": "S"
    },
    {
        "AttributeName": "SongTitle",
        "AttributeType": "S"
    }
],
"TableName": "MusicCollection",
"KeySchema": [
    {
        "AttributeName": "Artist",
        "KeyType": "HASH"
    },
    {
        "AttributeName": "SongTitle",
        "KeyType": "RANGE"
    }
],
"TableStatus": "UPDATING",
"CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
"ProvisionedThroughput": {
    "LastIncreaseDateTime": "2020-07-28T12:59:17.537000-07:00",
    "NumberOfDecreasesToday": 0,
    "ReadCapacityUnits": 15,
    "WriteCapacityUnits": 10
},
"TableSizeBytes": 182,
"ItemCount": 2,
"TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection",
"TableId": "abcd0123-01ab-23cd-0123-abcdef123456",
"BillingModeSummary": {
    "BillingMode": "PROVISIONED",
    "LastUpdateToPayPerRequestDateTime": "2020-07-28T13:14:48.366000-07:00"
},
"LocalSecondaryIndexes": [
    {
        "IndexName": "AlbumTitleIndex",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            }
        ]
    }
]
```

```
        "AttributeName": "AlbumTitle",
        "KeyType": "RANGE"
    }
],
"Projection": {
    "ProjectionType": "INCLUDE",
    "NonKeyAttributes": [
        "Year",
        "Genre"
    ]
},
"IndexSizeBytes": 139,
"ItemCount": 2,
"IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitleIndex"
}
],
"GlobalSecondaryIndexes": [
    {
        "IndexName": "AlbumTitle-index",
        "KeySchema": [
            {
                "AttributeName": "AlbumTitle",
                "KeyType": "HASH"
            }
        ],
        "Projection": {
            "ProjectionType": "ALL"
        },
        "IndexStatus": "ACTIVE",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 10
        },
        "IndexSizeBytes": 0,
        "ItemCount": 0,
        "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitle-index"
    }
],
"StreamSpecification": {
    "StreamEnabled": true,
    "StreamViewType": "NEW_IMAGE"
}
```

```
    },
    "LatestStreamLabel": "2020-07-28T21:53:39.112",
    "LatestStreamArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/stream/2020-07-28T21:53:39.112"
  }
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[更新表](#)。

#### 示例 4：启用服务器端加密

以下示例在 MusicCollection 表上启用服务器端加密。

```
aws dynamodb update-table \
  --table-name MusicCollection \
  --sse-specification Enabled=true,SSEType=KMS
```

输出：

```
{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "AlbumTitle",
        "AttributeType": "S"
      },
      {
        "AttributeName": "Artist",
        "AttributeType": "S"
      },
      {
        "AttributeName": "SongTitle",
        "AttributeType": "S"
      }
    ],
    "TableName": "MusicCollection",
    "KeySchema": [
      {
        "AttributeName": "Artist",
        "KeyType": "HASH"
      },
      {
        "AttributeName": "SongTitle",
```

```
        "KeyType": "RANGE"
      }
    ],
    "TableStatus": "ACTIVE",
    "CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
    "ProvisionedThroughput": {
      "LastIncreaseDateTime": "2020-07-28T12:59:17.537000-07:00",
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 15,
      "WriteCapacityUnits": 10
    },
    "TableSizeBytes": 182,
    "ItemCount": 2,
    "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection",
    "TableId": "abcd0123-01ab-23cd-0123-abcdef123456",
    "BillingModeSummary": {
      "BillingMode": "PROVISIONED",
      "LastUpdateToPayPerRequestDateTime": "2020-07-28T13:14:48.366000-07:00"
    },
    "LocalSecondaryIndexes": [
      {
        "IndexName": "AlbumTitleIndex",
        "KeySchema": [
          {
            "AttributeName": "Artist",
            "KeyType": "HASH"
          },
          {
            "AttributeName": "AlbumTitle",
            "KeyType": "RANGE"
          }
        ],
        "Projection": {
          "ProjectionType": "INCLUDE",
          "NonKeyAttributes": [
            "Year",
            "Genre"
          ]
        },
        "IndexSizeBytes": 139,
        "ItemCount": 2,
        "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitleIndex"
      }
    ]
  }
}
```

```
    ],
    "GlobalSecondaryIndexes": [
      {
        "IndexName": "AlbumTitle-index",
        "KeySchema": [
          {
            "AttributeName": "AlbumTitle",
            "KeyType": "HASH"
          }
        ],
        "Projection": {
          "ProjectionType": "ALL"
        },
        "IndexStatus": "ACTIVE",
        "ProvisionedThroughput": {
          "NumberOfDecreasesToday": 0,
          "ReadCapacityUnits": 10,
          "WriteCapacityUnits": 10
        },
        "IndexSizeBytes": 0,
        "ItemCount": 0,
        "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitle-index"
      }
    ],
    "StreamSpecification": {
      "StreamEnabled": true,
      "StreamViewType": "NEW_IMAGE"
    },
    "LatestStreamLabel": "2020-07-28T21:53:39.112",
    "LatestStreamArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/stream/2020-07-28T21:53:39.112",
    "SSEDescription": {
      "Status": "UPDATING"
    }
  }
}
```

有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[更新表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateTable](#)中的。



## update-time-to-live

以下代码示例演示了如何使用 update-time-to-live。

### AWS CLI

更新桌子上的 Time to Live 设置

以下update-time-to-live示例在指定表上启用 Time to Live。

```
aws dynamodb update-time-to-live \  
  --table-name MusicCollection \  
  --time-to-live-specification Enabled=true,AttributeName=ttl
```

输出：

```
{  
  "TimeToLiveSpecification": {  
    "Enabled": true,  
    "AttributeName": "ttl"  
  }  
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的[上线时间](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateTimeToLive](#)中的。

## 使用 DynamoDB Streams 示例 AWS CLI

以下代码示例向您展示了如何在 DynamoDB Streams 中 AWS Command Line Interface 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

## 操作

### describe-stream

以下代码示例演示了如何使用 describe-stream。

#### AWS CLI

获取有关 DynamoDB 直播的信息

以下describe-stream命令显示有关特定 DynamoDB 流的信息。

```
aws dynamodbstreams describe-stream \  
  --stream-arn arn:aws:dynamodb:us-west-1:123456789012:table/Music/  
  stream/2019-10-22T18:02:01.576
```

输出：

```
{  
  "StreamDescription": {  
    "StreamArn": "arn:aws:dynamodb:us-west-1:123456789012:table/Music/  
stream/2019-10-22T18:02:01.576",  
    "StreamLabel": "2019-10-22T18:02:01.576",  
    "StreamStatus": "ENABLED",  
    "StreamViewType": "NEW_AND_OLD_IMAGES",  
    "CreationRequestDateTime": 1571767321.571,  
    "TableName": "Music",  
    "KeySchema": [  
      {  
        "AttributeName": "Artist",  
        "KeyType": "HASH"  
      },  
      {  
        "AttributeName": "SongTitle",  
        "KeyType": "RANGE"  
      }  
    ],  
    "Shards": [  
      {  
        "ShardId": "shardId-00000001571767321804-697ce3d2",  
        "SequenceNumberRange": {  
          "StartingSequenceNumber": "4000000000000642977831",  
          "EndingSequenceNumber": "4000000000000642977831"  
        }  
      }  
    ]  
  }  
}
```

```

    },
    {
      "ShardId": "shardId-00000001571780995058-40810d86",
      "SequenceNumberRange": {
        "StartingSequenceNumber": "757400000000005655171150"
      },
      "ParentShardId": "shardId-00000001571767321804-697ce3d2"
    }
  ]
}
}

```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的[使用 DynamoDB Streams 捕获表格活动](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStream](#)中的。

## get-records

以下代码示例演示了如何使用 get-records。

### AWS CLI

从 Dynamodb 直播中获取记录

以下get-records命令使用指定的 Amazon DynamoDB 分片迭代器检索记录。

```

aws dynamodbstreams get-records \
  --shard-iterator "arn:aws:dynamodb:us-west-1:123456789012:table/Music/
stream/2019-10-22T18:02:01.576|1|
AAAAAAAAAAGgM3YZ89vLZZxjmoQeo33r9M4x3+zmmTLsiL86MfrF4+B4EbsByi52InVmi0Nmy6xVW4IRcIIbs1z07MNI
+CjNPlqQjnyRSAnf0wWmKhL1/KNParWSfz2odf780o00bIDIWRRMkt7+Hyzh9SD
+hFxFAWR5C7QI10XPc8mRBfNIazfrVCjJK8/jsjCzsqNyXKzJbhh+GXCoxYN
+Kpmg4nyj1EAsYhbGL35muvHFoHjcyuynbsczbWaxNfThDwRAYvoTmc8XhHKtAWUbJiaVd8ZPtQwDsThCrmDRPIdmTRG
+w/1EGS05ha1qNP+V14+tuhz2TRnhnJo/pny9GI/yGpce97mWvSPr5KPwy+Dtcm5BHayBs
+PVYHITaTliInFlT
+LCwvaz1QH3MY3b8A05Z800wjpkm60iQqtMeDwN4NX6FrcxR34JoFKGsgR8XkHVJzz2xr1xqSJ12ycpNTyHndusw==

```

输出：

```

{
  "Records": [
    {

```

```
"eventID": "c3b5d798eef6215d42f8137b19a88e50",
"eventName": "INSERT",
"eventVersion": "1.1",
"eventSource": "aws:dynamodb",
"awsRegion": "us-west-1",
"dynamodb": {
  "ApproximateCreationDateTime": 1571849028.0,
  "Keys": {
    "Artist": {
      "S": "No One You Know"
    },
    "SongTitle": {
      "S": "Call Me Today"
    }
  },
  "NewImage": {
    "AlbumTitle": {
      "S": "Somewhat Famous"
    },
    "Artist": {
      "S": "No One You Know"
    },
    "Awards": {
      "N": "1"
    },
    "SongTitle": {
      "S": "Call Me Today"
    }
  },
  "SequenceNumber": "700000000013256296913",
  "SizeBytes": 119,
  "StreamViewType": "NEW_AND_OLD_IMAGES"
},
{
  "eventID": "878960a6967867e2da16b27380a27328",
  "eventName": "INSERT",
  "eventVersion": "1.1",
  "eventSource": "aws:dynamodb",
  "awsRegion": "us-west-1",
  "dynamodb": {
    "ApproximateCreationDateTime": 1571849029.0,
    "Keys": {
      "Artist": {
```

```

        "S": "Acme Band"
      },
      "SongTitle": {
        "S": "Happy Day"
      }
    },
    "NewImage": {
      "AlbumTitle": {
        "S": "Songs About Life"
      },
      "Artist": {
        "S": "Acme Band"
      },
      "Awards": {
        "N": "10"
      },
      "SongTitle": {
        "S": "Happy Day"
      }
    },
    "SequenceNumber": "800000000013256297217",
    "SizeBytes": 100,
    "StreamViewType": "NEW_AND_OLD_IMAGES"
  }
},
{
  "eventID": "520fabde080e159fc3710b15ee1d4daa",
  "eventName": "MODIFY",
  "eventVersion": "1.1",
  "eventSource": "aws:dynamodb",
  "awsRegion": "us-west-1",
  "dynamodb": {
    "ApproximateCreationDateTime": 1571849734.0,
    "Keys": {
      "Artist": {
        "S": "Acme Band"
      },
      "SongTitle": {
        "S": "Happy Day"
      }
    }
  },
  "NewImage": {
    "AlbumTitle": {
      "S": "Updated Album Title"
    }
  }
}

```

```

    },
    "Artist": {
      "S": "Acme Band"
    },
    "Awards": {
      "N": "10"
    },
    "SongTitle": {
      "S": "Happy Day"
    }
  },
  "OldImage": {
    "AlbumTitle": {
      "S": "Songs About Life"
    },
    "Artist": {
      "S": "Acme Band"
    },
    "Awards": {
      "N": "10"
    },
    "SongTitle": {
      "S": "Happy Day"
    }
  },
  "SequenceNumber": "900000000013256687845",
  "SizeBytes": 170,
  "StreamViewType": "NEW_AND_OLD_IMAGES"
}
],
"NextShardIterator": "arn:aws:dynamodb:us-west-1:123456789012:table/
Music/stream/2019-10-23T16:41:08.740|1|AAAAAAAAAAAEhEI04jkFLW
+LK0wivjT8d/IHEh3iExV2xK00aTxEzVy1C1C7Kbb5+Z0W6bT9VQ2n1/
mrs7+PRia0ZCHJu7JHJVW7zlsq0i/ges3fw8GYEymyL+piEk35cx67rQqwKKyq
+Q6w9JyjreIOj4F2lWLv26lBwRTrIYC4IB7C3BZZK4715QwYdDxNdVHiSBRZX8UqoS6W0t0F87xZLNB9F/
NhYBLXi/wcGvAcBc0TNI0H+N0Nqwt0B/
FGcKnrF8YZ0xRoNN6RgGuVWHF3px0hxEJeFZoSoJTlKeG9YcYxzi5Ci/
mhdTm7tBXnbw5c6xmsGsBqTirNjldyJLcWl8C10UOLX63Ufo/5QliztcjEbKsQe28x8LM8o7VH1Is0fF/
ITt8awSA4igyJS0P87GN8Qri8kj8iaE35805jBHF2wvwT6Iy2xGrR2r2HzYps9dwG0arVdEITaJfWzNoL4HajMhmREZ
+V04i1YIeHMXJfcwetNRuIbdQXfJht2NQZa4PvV6iknY6d19MrdbSTMKoqAuvp6g3Q2jH4t7GKCLWgodcPAn8g5+43Da
}

```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的[使用 DynamoDB Streams 捕获表格活动](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRecords](#)中的。

## get-shard-iterator

以下代码示例演示了如何使用 get-shard-iterator。

### AWS CLI

#### 获取分片迭代器

以下get-shard-iterator命令检索指定分片的分片迭代器。

```
aws dynamodbstreams get-shard-iterator \  
  --stream-arn arn:aws:dynamodb:us-west-1:12356789012:table/Music/  
stream/2019-10-22T18:02:01.576 \  
  --shard-id shardId-00000001571780995058-40810d86 \  
  --shard-iterator-type LATEST
```

输出：

```
{  
  "ShardIterator": "arn:aws:dynamodb:us-west-1:123456789012:table/Music/  
stream/2019-10-22T18:02:01.576|1|  
AAAAAAAAAAGgM3YZ89vLZZxjmoQeo33r9M4x3+zmmTLsiL86MfrF4+B4EbsByi52InVmi0Nmy6xVW4IRcIIbs1z07MNI  
+CjNPlqQjnyRSAnf0wWmKhL1/KNParWSfz2odf780o00bIDIWRRMkt7+Hyzh9SD  
+hFxFAWR5C7QI10XPc8mRBfNIazfrVCjJK8/jsjCzsQnyXKzJbhH+GXCoxYN  
+Kpmg4nyj1EAsYhbGL35muvHFoHjcyuynbsczbWaXNfThDwRAYvoTmc8XhHKtAWUbJiaVd8ZPtQwDsThCrmDRPIdmTRG  
+w/1EGS05ha1qNP+V14+tuhz2TRnhnJo/pny9GI/yGpce97mWvSPr5KPwy+DtcM5BHayBs  
+PVYHITaTliInFlT  
+LCwvaz1QH3MY3b8A05Z800wjpkTm60iQqtMeDwN4NX6FrcxR34JoFKGsgR8XkHVJzz2xr1xqSJ12ycpNTyHnndusw=  
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的[使用 DynamoDB Streams 捕获表格活动](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetShardIterator](#)中的。

## list-streams

以下代码示例演示了如何使用 `list-streams`。

### AWS CLI

列出 DynamoDB 直播

以下 `list-streams` 命令列出了默认区域内的所有现有 Amazon DynamoDB 直播。AWS

```
aws dynamodbstreams list-streams
```

输出：

```
{
  "Streams": [
    {
      "StreamArn": "arn:aws:dynamodb:us-west-1:123456789012:table/Music/stream/2019-10-22T18:02:01.576",
      "TableName": "Music",
      "StreamLabel": "2019-10-22T18:02:01.576"
    }
  ]
}
```

有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的[使用 DynamoDB Streams 捕获表格活动](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListStreams](#)中的。

## 使用 Amazon EC2 的示例 AWS CLI

以下代码示例向您展示了如何在 Amazon EC2 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。



## 主题

- [操作](#)

## 操作

### **accept-address-transfer**

以下代码示例演示了如何使用 `accept-address-transfer`。

#### AWS CLI

接受向您的账户转移的弹性 IP 地址

以下 `accept-address-transfer` 示例接受将指定的弹性 IP 地址转移到您的账户。

```
aws ec2 accept-address-transfer \  
  --address 100.21.184.216
```

输出：

```
{  
  "AddressTransfer": {  
    "PublicIp": "100.21.184.216",  
    "AllocationId": "eipalloc-09ad461b0d03f6aaf",  
    "TransferAccountId": "123456789012",  
    "TransferOfferExpirationTimestamp": "2023-02-22T20:51:10.000Z",  
    "TransferOfferAcceptedTimestamp": "2023-02-22T22:52:54.000Z",  
    "AddressTransferStatus": "accepted"  
  }  
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的 [传输弹性 IP 地址](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AcceptAddressTransfer](#) 中的。

### **accept-reserved-instances-exchange-quote**

以下代码示例演示了如何使用 `accept-reserved-instances-exchange-quote`。

#### AWS CLI

执行可转换预留实例交换

此示例执行指定的可转换预留实例的交换。

命令:

```
aws ec2 accept-reserved-instances-exchange-quote --reserved-instance-ids 7b8750c3-397e-4da4-bbcb-a45ebexample --target-configurations OfferingId=b747b472-423c-48f3-8cee-679bcexample
```

输出:

```
{
  "ExchangeId": "riex-e68ed3c1-8bc8-4c17-af77-811afexample"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AcceptReservedInstancesExchangeQuote](#)中的。

## accept-transit-gateway-peering-attachment

以下代码示例演示了如何使用 `accept-transit-gateway-peering-attachment`。

### AWS CLI

接受公交网关对等连接

以下`accept-transit-gateway-peering-attachment`示例接受指定的传输网关对等连接。该`--region`参数指定接受者中转网关所在的区域。

```
aws ec2 accept-transit-gateway-peering-attachment \
  --transit-gateway-attachment-id tgw-attach-4455667788aabbccd \
  --region us-east-2
```

输出:

```
{
  "TransitGatewayPeeringAttachment": {
    "TransitGatewayAttachmentId": "tgw-attach-4455667788aabbccd",
    "RequesterTgwInfo": {
      "TransitGatewayId": "tgw-123abc05e04123abc",
      "OwnerId": "123456789012",
      "Region": "us-west-2"
    }
  }
}
```

```
    },
    "AcceptorTgwInfo": {
      "TransitGatewayId": "tgw-11223344aabbcc112",
      "OwnerId": "123456789012",
      "Region": "us-east-2"
    },
    "State": "pending",
    "CreationTime": "2019-12-09T11:38:31.000Z"
  }
}
```

有关更多信息，请参阅《[公交网关指南](#)》中的 [Transit Gateway 对等连接附件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AcceptTransitGatewayPeeringAttachment](#) 中的。

## accept-transit-gateway-vpc-attachment

以下代码示例演示了如何使用 `accept-transit-gateway-vpc-attachment`。

### AWS CLI

接受将 VPC 连接到中转网关的请求。

以下 `accept-transit-gateway-vpc-attachment` 示例接受请求 `forte` 指定的附件。

```
aws ec2 accept-transit-gateway-vpc-attachment \
  --transit-gateway-attachment-id tgw-attach-0a34fe6b4fEXAMPLE
```

输出：

```
{
  "TransitGatewayVpcAttachment": {
    "TransitGatewayAttachmentId": "tgw-attach-0a34fe6b4fEXAMPLE",
    "TransitGatewayId": "tgw-0262a0e521EXAMPLE",
    "VpcId": "vpc-07e8ffd50fEXAMPLE",
    "VpcOwnerId": "123456789012",
    "State": "pending",
    "SubnetIds": [
      "subnet-0752213d59EXAMPLE"
    ],
    "CreationTime": "2019-07-10T17:33:46.000Z",
    "Options": {
```

```
        "DnsSupport": "enable",
        "Ipv6Support": "disable"
    }
}
```

有关更多信息，请参阅《[传输网关指南](#)》中的 [VPC 的 Transit Gateway 附件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AcceptTransitGatewayVpcAttachment](#) 中的。

## accept-vpc-endpoint-connections

以下代码示例演示了如何使用 `accept-vpc-endpoint-connections`。

### AWS CLI

接受接口端点连接请求

此示例接受指定终端节点服务的指定端点连接请求。

命令:

```
aws ec2 accept-vpc-endpoint-connections --service-id vpce-svc-03d5ebb7d9579a2b3 --
vpc-endpoint-ids vpce-0c1308d7312217abc
```

输出:

```
{
  "Unsuccessful": []
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AcceptVpcEndpointConnections](#) 中的。

## accept-vpc-peering-connection

以下代码示例演示了如何使用 `accept-vpc-peering-connection`。

### AWS CLI

接受 VPC 对等连接

此示例接受指定的 VPC 对等连接请求。

**命令:**

```
aws ec2 accept-vpc-peering-connection --vpc-peering-connection-id pcx-1a2b3c4d
```

**输出 :**

```
{
  "VpcPeeringConnection": {
    "Status": {
      "Message": "Provisioning",
      "Code": "provisioning"
    },
    "Tags": [],
    "AccepterVpcInfo": {
      "OwnerId": "444455556666",
      "VpcId": "vpc-44455566",
      "CidrBlock": "10.0.1.0/28"
    },
    "VpcPeeringConnectionId": "pcx-1a2b3c4d",
    "RequesterVpcInfo": {
      "OwnerId": "444455556666",
      "VpcId": "vpc-111abc45",
      "CidrBlock": "10.0.0.0/28"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AcceptVpcPeeringConnection](#)中的。

**advertise-byoip-cidr**

以下代码示例演示了如何使用 advertise-byoip-cidr。

**AWS CLI****宣传地址范围**

以下advertise-byoip-cidr示例通告指定的公有 IPv4 地址范围。

```
aws ec2 advertise-byoip-cidr \
  --cidr 203.0.113.25/24
```

输出：

```
{
  "ByoipCidr": {
    "Cidr": "203.0.113.25/24",
    "StatusMessage": "ipv4pool-ec2-1234567890abcdef0",
    "State": "provisioned"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdvertiseByoipCidr](#)中的。

## allocate-address

以下代码示例演示了如何使用 allocate-address。

### AWS CLI

示例 1：从 Amazon 的地址池中分配弹性 IP 地址

以下 allocate-address 示例分配弹性 IP 地址。Amazon EC2 从 Amazon 的地址池中选择地址。

```
aws ec2 allocate-address
```

输出：

```
{
  "PublicIp": "70.224.234.241",
  "AllocationId": "eipalloc-01435ba59eEXAMPLE",
  "PublicIpv4Pool": "amazon",
  "NetworkBorderGroup": "us-west-2",
  "Domain": "vpc"
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[弹性 IP 地址](#)。

示例 2：分配弹性 IP 地址并将其与网络边界组关联

以下 allocate-address 示例分配弹性 IP 地址并将其与指定的网络边界组关联。

```
aws ec2 allocate-address \  
  --network-border-group us-west-2-lax-1
```

输出：

```
{  
  "PublicIp": "70.224.234.241",  
  "AllocationId": "eipalloc-e03dd489ceEXAMPLE",  
  "PublicIpv4Pool": "amazon",  
  "NetworkBorderGroup": "us-west-2-lax-1",  
  "Domain": "vpc"  
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[弹性 IP 地址](#)。

示例 3：从自己拥有的地址池中分配弹性 IP 地址

以下 `allocate-address` 示例从您引入 Amazon Web Services 账户的地址池中，分配弹性 IP 地址。Amazon EC2 从地址池中选择地址。

```
aws ec2 allocate-address \  
  --public-ipv4-pool ipv4pool-ec2-1234567890abcdef0
```

输出：

```
{  
  "AllocationId": "eipalloc-02463d08ceEXAMPLE",  
  "NetworkBorderGroup": "us-west-2",  
  "CustomerOwnedIp": "18.218.95.81",  
  "CustomerOwnedIpv4Pool": "ipv4pool-ec2-1234567890abcdef0",  
  "Domain": "vpc"  
  "NetworkBorderGroup": "us-west-2",  
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[弹性 IP 地址](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AllocateAddress](#)中的。

## allocate-hosts

以下代码示例演示了如何使用 `allocate-hosts`。

## AWS CLI

### 示例 1：分配专用主机

以下`allocate-hosts`示例在`eu-west-1a`可用区中分配了一台专用主机，您可以在该主机上启动`m5.large`实例。默认情况下，专用主机仅接受目标实例启动，不支持主机恢复。

```
aws ec2 allocate-hosts \  
  --instance-type m5.large \  
  --availability-zone eu-west-1a \  
  --quantity 1
```

输出：

```
{  
  "HostIds": [  
    "h-07879acf49EXAMPLE"  
  ]  
}
```

### 示例 2：分配启用自动放置和主机恢复的专用主机

以下`allocate-hosts`示例在`eu-west-1a`可用区中分配一台启用了自动放置和主机恢复的专用主机。

```
aws ec2 allocate-hosts \  
  --instance-type m5.large \  
  --availability-zone eu-west-1a \  
  --auto-placement on \  
  --host-recovery on \  
  --quantity 1
```

输出：

```
{  
  "HostIds": [  
    "h-07879acf49EXAMPLE"  
  ]  
}
```

### 示例 3：分配带有标签的专用主机



以下allocate-hosts示例分配了一台专用主机，并应用了密钥名为purpose、值为production的标签。

```
aws ec2 allocate-hosts \  
  --instance-type m5.large \  
  --availability-zone eu-west-1a \  
  --quantity 1 \  
  --tag-specifications 'ResourceType=dedicated-  
host,Tags={Key=purpose,Value=production}'
```

输出：

```
{  
  "HostIds": [  
    "h-07879acf49EXAMPLE"  
  ]  
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[分配专用主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AllocateHosts](#)中的。

## allocate-ipam-pool-cidr

以下代码示例演示了如何使用 allocate-ipam-pool-cidr。

AWS CLI

从 IPAM 池中分配 CIDR

以下allocate-ipam-pool-cidr示例从 IPAM 池中分配一个 CIDR。

(Linux)：

```
aws ec2 allocate-ipam-pool-cidr \  
  --ipam-pool-id ipam-pool-0533048da7d823723 \  
  --netmask-length 24
```

( 视窗 )：

```
aws ec2 allocate-ipam-pool-cidr ^  
  --ipam-pool-id ipam-pool-0533048da7d823723 ^
```

```
--netmask-length 24
```

输出：

```
{
  "IpamPoolAllocation": {
    "Cidr": "10.0.0.0/24",
    "IpamPoolAllocationId": "ipam-pool-alloc-018ecc28043b54ba38e2cd99943cebfbfd",
    "ResourceType": "custom",
    "ResourceOwner": "123456789012"
  }
}
```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的手动向池分配 CIDR 以保留 [IP 地址空间](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AllocateIpamPoolCidr](#) 中的。

## apply-security-groups-to-client-vpn-target-network

以下代码示例演示了如何使用 `apply-security-groups-to-client-vpn-target-network`。

AWS CLI

将安全组应用于 Client VPN 终端节点的目标网络

以下 `apply-security-groups-to-client-vpn-target-network` 示例将安全组 `sg-01f6e627a89f4db32` 应用于指定目标网络和 Client VPN 端点之间的关联。

```
aws ec2 apply-security-groups-to-client-vpn-target-network \
  --security-group-ids sg-01f6e627a89f4db32 \
  --vpc-id vpc-0e2110c2f324332e0 \
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde
```

输出：

```
{
  "SecurityGroupIds": [
    "sg-01f6e627a89f4db32"
  ]
}
```

有关更多信息，请参阅《AWS Client VPN 管理员指南》中的“[目标网络](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ApplySecurityGroupsToClientVpnTargetNetwork](#)中的。

## assign-ipv6-addresses

以下代码示例演示了如何使用 assign-ipv6-addresses。

### AWS CLI

为网络接口分配特定的 IPv6 地址

此示例将指定的 IPv6 地址分配给指定的网络接口。

命令:

```
aws ec2 assign-ipv6-addresses --network-interface-id eni-38664473 --ipv6-addresses
2001:db8:1234:1a00:3304:8879:34cf:4071 2001:db8:1234:1a00:9691:9503:25ad:1761
```

输出:

```
{
  "AssignedIpv6Addresses": [
    "2001:db8:1234:1a00:3304:8879:34cf:4071",
    "2001:db8:1234:1a00:9691:9503:25ad:1761"
  ],
  "NetworkInterfaceId": "eni-38664473"
}
```

将亚马逊选择的 IPv6 地址分配给网络接口

此示例为指定的网络接口分配两个 IPv6 地址。亚马逊会自动从子网的 IPv6 CIDR 区块范围内的可用 IPv6 地址中分配这些 IPv6 地址。

命令:

```
aws ec2 assign-ipv6-addresses --network-interface-id eni-38664473 --ipv6-address-
count 2
```

输出:

```
{
  "AssignedIpv6Addresses": [
    "2001:db8:1234:1a00:3304:8879:34cf:4071",
    "2001:db8:1234:1a00:9691:9503:25ad:1761"
  ],
  "NetworkInterfaceId": "eni-38664473"
}
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [AssignIpv6Addresses](#)。

## assign-private-ip-addresses

以下代码示例演示了如何使用 `assign-private-ip-addresses`。

### AWS CLI

为特定的辅助私有 IP 地址分配网络接口

此示例将指定的辅助私有 IP 地址分配给指定的网络接口。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 assign-private-ip-addresses --network-interface-id eni-e5aa89a3 --private-ip-addresses 10.0.0.82
```

将 Amazon EC2 选择的辅助私有 IP 地址分配给网络接口

此示例将两个辅助私有 IP 地址分配给指定的网络接口。Amazon EC2 会自动从与网络接口关联的子网的 CIDR 块范围内的可用 IP 地址中分配这些 IP 地址。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 assign-private-ip-addresses --network-interface-id eni-e5aa89a3 --secondary-private-ip-address-count 2
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssignPrivateIpAddresses](#) 中的。

## assign-private-nat-gateway-address

以下代码示例演示了如何使用 `assign-private-nat-gateway-address`。

## AWS CLI

为您的私有 NAT 网关分配私有 IP 地址

以下 `assign-private-nat-gateway-address` 示例为指定的私有 NAT 网关分配两个私有 IP 地址。

```
aws ec2 assign-private-nat-gateway-address \
  --nat-gateway-id nat-1234567890abcdef0 \
  --private-ip-address-count 2
```

输出：

```
{
  "NatGatewayId": "nat-1234567890abcdef0",
  "NatGatewayAddresses": [
    {
      "NetworkInterfaceId": "eni-0065a61b324d1897a",
      "IsPrimary": false,
      "Status": "assigning"
    },
    {
      "NetworkInterfaceId": "eni-0065a61b324d1897a",
      "IsPrimary": false,
      "Status": "assigning"
    }
  ]
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 [NAT 网关](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssignPrivateNatGatewayAddress](#) 中的。

## associate-address

以下代码示例演示了如何使用 `associate-address`。

## AWS CLI

在 EC2-Classical 中关联弹性 IP 地址

本示例将弹性 IP 地址与 EC2-Classical 中的实例相关联。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 associate-address --instance-id i-07ffe74c7330ebf53 --public-ip 198.51.100.0
```

关联 EC2-VPC 中的弹性 IP 地址

本示例将弹性 IP 地址与 VPC 中的实例相关联。

命令:

```
aws ec2 associate-address --instance-id i-0b263919b6498b123 --allocation-id eipalloc-64d5890a
```

输出 :

```
{
  "AssociationId": "eipassoc-2bebb745"
}
```

本示例将弹性 IP 地址与网络接口相关联。

命令:

```
aws ec2 associate-address --allocation-id eipalloc-64d5890a --network-interface-id eni-1a2b3c4d
```

本示例将弹性 IP 与和网络接口关联的私有 IP 地址相关联。

命令:

```
aws ec2 associate-address --allocation-id eipalloc-64d5890a --network-interface-id eni-1a2b3c4d --private-ip-address 10.0.0.85
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateAddress](#)中的。

## **associate-client-vpn-target-network**

以下代码示例演示了如何使用 `associate-client-vpn-target-network`。

AWS CLI

将目标网络与 Client VPN 端点相关联

以下 `associate-client-vpn-target-network` 示例将子网与指定的 Client VPN 端点相关联。

```
aws ec2 associate-client-vpn-target-network \  
  --subnet-id subnet-0123456789abcabca \  
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde
```

输出：

```
{  
  "AssociationId": "cvpn-assoc-12312312312312312",  
  "Status": {  
    "Code": "associating"  
  }  
}
```

有关更多信息，请参阅《AWS Client VPN 管理员指南》中的“[目标网络](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateClientVpnTargetNetwork](#) 中的。

## associate-dhcp-options

以下代码示例演示了如何使用 `associate-dhcp-options`。

### AWS CLI

将 DHCP 选项集与您的 VPC 关联

此示例将指定的 DHCP 选项集与指定的 VPC 关联起来。如果命令成功，则不返回任何输出。

命令：

```
aws ec2 associate-dhcp-options --dhcp-options-id dopt-d9070ebb --vpc-id vpc-a01106c2
```

将默认 DHCP 选项集与您的 VPC 相关联

此示例将默认 DHCP 选项集与指定的 VPC 关联起来。如果命令成功，则不返回任何输出。

命令：

```
aws ec2 associate-dhcp-options --dhcp-options-id default --vpc-id vpc-a01106c2
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateDhcpOptions](#) 中的。

## associate-iam-instance-profile

以下代码示例演示了如何使用 `associate-iam-instance-profile`。

### AWS CLI

将 IAM 实例配置文件与实例关联

此示例将名为 `admin-role` 的 IAM 实例配置文件与实例 `i-123456789abcde123` 相关联。

命令:

```
aws ec2 associate-iam-instance-profile --instance-id i-123456789abcde123 --iam-
instance-profile Name=admin-role
```

输出:

```
{
  "IamInstanceProfileAssociation": {
    "InstanceId": "i-123456789abcde123",
    "State": "associating",
    "AssociationId": "iip-assoc-0e7736511a163c209",
    "IamInstanceProfile": {
      "Id": "AIPAJBLK7RKJKWDXVHIEC",
      "Arn": "arn:aws:iam::123456789012:instance-profile/admin-role"
    }
  }
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateIamInstanceProfile](#) 中的。

## associate-instance-event-window

以下代码示例演示了如何使用 `associate-instance-event-window`。

### AWS CLI

示例 1：将一个或多个实例与事件窗口关联

以下 `associate-instance-event-window` 示例将一个或多个实例与事件窗口相关联。

```
aws ec2 associate-instance-event-window \
```



```
--region us-east-1 \
--instance-event-window-id iew-0abcdef1234567890 \
--association-target "InstanceIds=i-1234567890abcdef0,i-0598c7d356eba48d7"
```

输出：

```
{
  "InstanceEventWindow": {
    "InstanceEventWindowId": "iew-0abcdef1234567890",
    "Name": "myEventWindowName",
    "CronExpression": "* 21-23 * * 2,3",
    "AssociationTarget": {
      "InstanceIds": [
        "i-1234567890abcdef0",
        "i-0598c7d356eba48d7"
      ],
      "Tags": [],
      "DedicatedHostIds": []
    },
    "State": "creating"
  }
}
```

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中计划事件部分的[注意事项](#)。

示例 2：将实例标签与事件窗口关联

以下associate-instance-event-window示例将实例标签与事件窗口相关联。输入instance-event-window-id参数以指定事件窗口。要关联实例标签，请指定association-target参数，为参数值指定一个或多个标签。

```
aws ec2 associate-instance-event-window \
--region us-east-1 \
--instance-event-window-id iew-0abcdef1234567890 \
--association-target "InstanceTags=[{Key=k2,Value=v2},{Key=k1,Value=v1}]"
```

输出：

```
{
  "InstanceEventWindow": {
    "InstanceEventWindowId": "iew-0abcdef1234567890",
```

```

    "Name": "myEventWindowName",
    "CronExpression": "* 21-23 * * 2,3",
    "AssociationTarget": {
      "InstanceIds": [],
      "Tags": [
        {
          "Key": "k2",
          "Value": "v2"
        },
        {
          "Key": "k1",
          "Value": "v1"
        }
      ],
      "DedicatedHostIds": []
    },
    "State": "creating"
  }
}

```

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中计划事件部分的[注意事项](#)。

### 示例 3：将专用主机与事件窗口关联

以下 `associate-instance-event-window` 示例将专用主机与事件窗口相关联。输入 `instance-event-window-id` 参数以指定事件窗口。要关联专用主机，请指定 `--association-target` 参数，为参数值指定一个或多个专用主机 ID。

```

aws ec2 associate-instance-event-window \
  --region us-east-1 \
  --instance-event-window-id iew-0abcdef1234567890 \
  --association-target "DedicatedHostIds=h-029fa35a02b99801d"

```

输出：

```

{
  "InstanceEventWindow": {
    "InstanceEventWindowId": "iew-0abcdef1234567890",
    "Name": "myEventWindowName",
    "CronExpression": "* 21-23 * * 2,3",
    "AssociationTarget": {
      "InstanceIds": [],

```

```
        "Tags": [],
        "DedicatedHostIds": [
            "h-029fa35a02b99801d"
        ]
    },
    "State": "creating"
}
}
```

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中计划事件部分的[注意事项](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateInstanceEventWindow](#) 中的。

## associate-ipam-resource-discovery

以下代码示例演示了如何使用 `associate-ipam-resource-discovery`。

### AWS CLI

将资源发现与 IPAM 关联

在此示例中，您是 IPAM 委托的管理员，并且已创建资源发现并由另一个 AWS 账户与您共享，以便您可以使用 IPAM 管理和监控其他账户拥有的资源 CIDR。

### 备注

要完成此请求，您需要使用可以获得的资源发现 ID [describe-ipam-resource-discoveries](#) 和可以通过 `desc` [ribe-ipams](#) 获得的 IP AM ID。您要关联的资源发现必须先使用 AWS RAM 与您的账户共享。--region 您输入的内容必须与要关联的 IPAM 的主区域相匹配。

以下 `associate-ipam-resource-discovery` 示例将资源发现与 IPAM 关联起来。

```
aws ec2 associate-ipam-resource-discovery \
  --ipam-id ipam-005f921c17ebd5107 \
  --ipam-resource-discovery-id ipam-res-disco-03e0406de76a044ee \
  --tag-specifications 'ResourceType=ipam-resource-discovery,Tags=[{Key=cost-
center,Value=cc123}]' \
  --region us-east-1
```

输出：

```
{
```

```

{
  "IpamResourceDiscoveryAssociation": {
    "OwnerId": "320805250157",
    "IpamResourceDiscoveryAssociationId": "ipam-res-disco-
assoc-04382a6346357cf82",
    "IpamResourceDiscoveryAssociationArn": "arn:aws:ec2::320805250157:ipam-
resource-discovery-association/ipam-res-disco-assoc-04382a6346357cf82",
    "IpamResourceDiscoveryId": "ipam-res-disco-0365d2977fc1672fe",
    "IpamId": "ipam-005f921c17ebd5107",
    "IpamArn": "arn:aws:ec2::320805250157:ipam/ipam-005f921c17ebd5107",
    "IpamRegion": "us-east-1",
    "IsDefault": false,
    "ResourceDiscoveryStatus": "active",
    "State": "associate-in-progress",
    "Tags": []
  }
}

```

关联资源发现后，您可以监控和/或管理其他账户创建的资源的 IP 地址。有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的将 IPAM [M 与组织以外的账户集成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateIpamResourceDiscovery](#) 中的。

## associate-nat-gateway-address

以下代码示例演示了如何使用 `associate-nat-gateway-address`。

### AWS CLI

将弹性 IP 地址与公有 NAT 网关关联

以下 `associate-nat-gateway-address` 示例将指定的弹性 IP 地址与指定的公有 NAT 网关相关联。AWS 自动分配辅助私有 IPv4 地址。

```

aws ec2 associate-nat-gateway-address \
  --nat-gateway-id nat-1234567890abcdef0 \
  --allocation-ids eipalloc-0be6ecac95EXAMPLE

```

输出：

```

{

```

```
"NatGatewayId": "nat-1234567890abcdef0",
"NatGatewayAddresses": [
  {
    "AllocationId": "eipalloc-0be6ecac95EXAMPLE",
    "NetworkInterfaceId": "eni-09cc4b2558794f7f9",
    "IsPrimary": false,
    "Status": "associating"
  }
]
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 [NAT 网关](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateNatGatewayAddress](#)中的。

## associate-route-table

以下代码示例演示了如何使用 `associate-route-table`。

### AWS CLI

将路由表与子网关联

此示例将指定的路由表与指定的子网关联。

命令：

```
aws ec2 associate-route-table --route-table-id rtb-22574640 --subnet-id
subnet-9d4a7b6c
```

输出：

```
{
  "AssociationId": "rtbassoc-781d0d1a"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateRouteTable](#)中的。

## associate-subnet-cidr-block

以下代码示例演示了如何使用 `associate-subnet-cidr-block`。

## AWS CLI

### 将 IPv6 CIDR 块与子网关联

此示例将 IPv6 CIDR 块与指定的子网关联。

命令:

```
aws ec2 associate-subnet-cidr-block --subnet-id subnet-5f46ec3b --ipv6-cidr-block 2001:db8:1234:1a00::/64
```

输出:

```
{
  "SubnetId": "subnet-5f46ec3b",
  "Ipv6CidrBlockAssociation": {
    "Ipv6CidrBlock": "2001:db8:1234:1a00::/64",
    "AssociationId": "subnet-cidr-assoc-3aa54053",
    "Ipv6CidrBlockState": {
      "State": "associating"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateSubnetCidrBlock](#)中的。

## associate-transit-gateway-multicast-domain

以下代码示例演示了如何使用 `associate-transit-gateway-multicast-domain`。

### AWS CLI

#### 将传输网关与多播域关联

以下`associate-transit-gateway-multicast-domain`示例将指定的子网和连接与指定的多播域相关联。

```
aws ec2 associate-transit-gateway-multicast-domain \
  --transit-gateway-multicast-domain-id tgw-mcast-domain-0c4905cef79d6e597 \
  --transit-gateway-attachment-id tgw-attach-028c1dd0f8f5cbe8e \
  --subnet-ids subnet-000de86e3b49c932a \
  --transit-gateway-multicast-domain-id tgw-mcast-domain-0c4905cef7EXAMPLE
```

输出：

```
{
  "Associations": {
    "TransitGatewayMulticastDomainId": "tgw-mcast-domain-0c4905cef79d6e597",
    "TransitGatewayAttachmentId": "tgw-attach-028c1dd0f8f5cbe8e",
    "ResourceId": "vpc-01128d2c240c09bd5",
    "ResourceType": "vpc",
    "Subnets": [
      {
        "SubnetId": "subnet-000de86e3b49c932a",
        "State": "associating"
      }
    ]
  }
}
```

有关更多信息，请参阅《传输网关指南》中的[管理多播域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateTransitGatewayMulticastDomain](#)中的。

## associate-transit-gateway-route-table

以下代码示例演示了如何使用 `associate-transit-gateway-route-table`。

### AWS CLI

将公网网关路由表与公网网关附件关联

以下示例将指定的传输网关路由表与指定的 VPC 连接相关联。

```
aws ec2 associate-transit-gateway-route-table \
  --transit-gateway-route-table-id tgw-rtb-002573ed1eEXAMPLE \
  --transit-gateway-attachment-id tgw-attach-0b5968d3b6EXAMPLE
```

输出：

```
{
  "Association": {
    "TransitGatewayRouteTableId": "tgw-rtb-002573ed1eEXAMPLE",
    "TransitGatewayAttachmentId": "tgw-attach-0b5968d3b6EXAMPLE",
```

```

    "ResourceId": "vpc-0065acced4EXAMPLE",
    "ResourceType": "vpc",
    "State": "associating"
  }
}

```

有关更多信息，请参阅 [Transit Gateway 指南中的关联 Transit Gateway 路由表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateTransitGatewayRouteTable](#) 中的。

## associate-vpc-cidr-block

以下代码示例演示了如何使用 `associate-vpc-cidr-block`。

### AWS CLI

示例 1：将亚马逊提供的 IPv6 CIDR 区块与 VPC 关联

以下 `associate-vpc-cidr-block` 示例将 IPv6 网段与指定的 VPC 关联起来。：

```

aws ec2 associate-vpc-cidr-block \
  --amazon-provided-ipv6-cidr-block \
  --ipv6-cidr-block-network-border-group us-west-2-lax-1 \
  --vpc-id vpc-8EXAMPLE

```

输出：

```

{
  "Ipv6CidrBlockAssociation": {
    "AssociationId": "vpc-cidr-assoc-0838ce7d9dEXAMPLE",
    "Ipv6CidrBlockState": {
      "State": "associating"
    },
    "NetworkBorderGroup": "us-west-2-lax-1"
  },
  "VpcId": "vpc-8EXAMPLE"
}

```

示例 2：将额外的 IPv4 网段与 VPC 关联起来

以下 `associate-vpc-cidr-block` 示例将 IPv4 CIDR 块 `10.2.0.0/16` 与指定的 VPC 相关联。



```
aws ec2 associate-vpc-cidr-block \  
  --vpc-id vpc-1EXAMPLE \  
  --cidr-block 10.2.0.0/16
```

输出：

```
{  
  "CidrBlockAssociation": {  
    "AssociationId": "vpc-cidr-assoc-2EXAMPLE",  
    "CidrBlock": "10.2.0.0/16",  
    "CidrBlockState": {  
      "State": "associating"  
    }  
  },  
  "VpcId": "vpc-1EXAMPLE"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateVpcCidrBlock](#)中的。

## attach-classic-link-vpc

以下代码示例演示了如何使用 attach-classic-link-vpc。

### AWS CLI

将 EC2-Classic 实例链接（附加）到 VPC

此示例通过 VPC 安全组 sg-12312312 将实例 i-1234567890abcdef0 关联到 VPC vpc-88888888。

命令：

```
aws ec2 attach-classic-link-vpc --instance-id i-1234567890abcdef0 --vpc-id  
vpc-88888888 --groups sg-12312312
```

输出：

```
{  
  "Return": true  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachClassicLinkVpc](#)中的。

## attach-internet-gateway

以下代码示例演示了如何使用 attach-internet-gateway。

### AWS CLI

将互联网网关连接到您的 VPC

以下attach-internet-gateway示例将指定的互联网网关连接到特定 VPC。

```
aws ec2 attach-internet-gateway \  
  --internet-gateway-id igw-0d0fb496b3EXAMPLE \  
  --vpc-id vpc-0a60eb65b4EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon VPC 用户指南》中的[互联网网关](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachInternetGateway](#)中的。

## attach-network-interface

以下代码示例演示了如何使用 attach-network-interface。

### AWS CLI

示例 1：将网络接口连接到实例

以下attach-network-interface示例将指定的网络接口连接到指定的实例。

```
aws ec2 attach-network-interface \  
  --network-interface-id eni-0dc56a8d4640ad10a \  
  --instance-id i-1234567890abcdef0 \  
  --device-index 1
```

输出：

```
{  
  "AttachmentId": "eni-attach-01a8fc87363f07cf9"  
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[弹性网络接口](#)。

## 示例 2：将网络接口连接到带有多张网卡的实例

以下attach-network-interface示例将指定的网络接口连接到指定的实例和网卡。

```
aws ec2 attach-network-interface \  
  --network-interface-id eni-07483b1897541ad83 \  
  --instance-id i-01234567890abcdef \  
  --network-card-index 1 \  
  --device-index 1
```

输出：

```
{  
  "AttachmentId": "eni-attach-0fbd7ee87a88cd06c"  
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[弹性网络接口](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachNetworkInterface](#)中的。

## attach-verified-access-trust-provider

以下代码示例演示了如何使用 attach-verified-access-trust-provider。

### AWS CLI

将信任提供者附加到实例

以下attach-verified-access-trust-provider示例将指定的已验证访问信任提供者附加到指定的 Verified Access 实例。

```
aws ec2 attach-verified-access-trust-provider \  
  --verified-access-instance-id vai-0ce000c0b7643abea \  
  --verified-access-trust-provider-id vatp-0bb32de759a3e19e7
```

输出：

```
{  
  "VerifiedAccessTrustProvider": {  
    "VerifiedAccessTrustProviderId": "vatp-0bb32de759a3e19e7",  
    "Description": "",  
    "TrustProviderType": "user",
```

```

    "UserTrustProviderType": "iam-identity-center",
    "PolicyReferenceName": "idc",
    "CreationTime": "2023-08-25T19:00:38",
    "LastUpdatedTime": "2023-08-25T19:00:38"
  },
  "VerifiedAccessInstance": {
    "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",
    "Description": "",
    "VerifiedAccessTrustProviders": [
      {
        "VerifiedAccessTrustProviderId": "vatp-0bb32de759a3e19e7",
        "TrustProviderType": "user",
        "UserTrustProviderType": "iam-identity-center"
      }
    ],
    "CreationTime": "2023-08-25T18:27:56",
    "LastUpdatedTime": "2023-08-25T18:27:56"
  }
}

```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachVerifiedAccessTrustProvider](#)中的。

## attach-volume

以下代码示例演示了如何使用 attach-volume。

### AWS CLI

将卷连接到实例

此示例命令将卷 (vol-1234567890abcdef0) 附加到实例 (i-01474ef662b89480) 上/dev/sdf。

命令:

```
aws ec2 attach-volume --volume-id vol-1234567890abcdef0 --instance-id
i-01474ef662b89480 --device /dev/sdf
```

输出:

```
{
```

```
"AttachTime": "YYYY-MM-DDTHH:MM:SS.000Z",
"InstanceId": "i-01474ef662b89480",
"VolumeId": "vol-1234567890abcdef0",
"State": "attaching",
"Device": "/dev/sdf"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachVolume](#)中的。

## attach-vpn-gateway

以下代码示例演示了如何使用 attach-vpn-gateway。

### AWS CLI

将虚拟私有网关连接到您的 VPC

以下attach-vpn-gateway示例将指定的虚拟专用网关连接到指定的 VPC。

```
aws ec2 attach-vpn-gateway \
  --vpn-gateway-id vgw-9a4cacf3 \
  --vpc-id vpc-a01106c2
```

输出：

```
{
  "VpcAttachment": {
    "State": "attaching",
    "VpcId": "vpc-a01106c2"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachVpnGateway](#)中的。

## authorize-client-vpn-ingress

以下代码示例演示了如何使用 authorize-client-vpn-ingress。

### AWS CLI

为 Client VPN 端点添加授权规则

以下 `authorize-client-vpn-ingress` 示例添加了一个允许所有客户端访问互联网的入口授权规则 (`0.0.0.0/0`)。

```
aws ec2 authorize-client-vpn-ingress \  
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde \  
  --target-network-cidr 0.0.0.0/0 \  
  --authorize-all-groups
```

输出：

```
{  
  "Status": {  
    "Code": "authorizing"  
  }  
}
```

有关更多信息，请参阅《AWS Client VPN 管理员指南》中的[授权规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AuthorizeClientVpnIngress](#) 中的。

## **authorize-security-group-egress**

以下代码示例演示了如何使用 `authorize-security-group-egress`。

### AWS CLI

添加允许出站流量到达特定地址范围的规则

此示例命令添加了一条规则，用于授予对 TCP 端口 80 上指定地址范围的访问权限。

命令 ( Linux )：

```
aws ec2 authorize-security-group-egress --group-id sg-1a2b3c4d --ip-permissions  
  IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges='[{"CidrIp=10.0.0.0/16}]'
```

命令 ( Windows )：

```
aws ec2 authorize-security-group-egress --group-id sg-1a2b3c4d --ip-permissions  
  IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges=[{"CidrIp=10.0.0.0/16}]
```

添加允许出站流量进入特定安全组的规则

此示例命令添加了一条规则，用于授予对 TCP 端口 80 上指定安全组的访问权限。

命令 ( Linux ) :

```
aws ec2 authorize-security-group-egress --group-id sg-1a2b3c4d --ip-permissions
IpProtocol=tcp,FromPort=80,ToPort=80,UserIdGroupPairs='[{"GroupId=sg-4b51a32f}]'
```

命令 ( Windows ) :

```
aws ec2 authorize-security-group-egress --group-id sg-1a2b3c4d --ip-permissions
IpProtocol=tcp,FromPort=80,ToPort=80,UserIdGroupPairs=[{"GroupId=sg-4b51a32f}]
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AuthorizeSecurityGroupEgress](#) 中的。

## authorize-security-group-ingress

以下代码示例演示了如何使用 `authorize-security-group-ingress`。

### AWS CLI

示例 1：添加允许入站 SSH 流量的规则

以下 `authorize-security-group-ingress` 示例将添加一个规则，该规则允许入站流量通过 TCP 端口 22 (SSH)。

```
aws ec2 authorize-security-group-ingress \
  --group-id sg-1234567890abcdef0 \
  --protocol tcp \
  --port 22 \
  --cidr 203.0.113.0/24
```

输出：

```
{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-01afa97ef3e1bedfc",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
```

```

        "IsEgress": false,
        "IpProtocol": "tcp",
        "FromPort": 22,
        "ToPort": 22,
        "CidrIpv4": "203.0.113.0/24"
    }
]
}

```

## 示例 2：添加允许来自其他安全组的入站 HTTP 流量的规则

以下 `authorize-security-group-ingress` 示例将添加一个规则，该规则允许源安全组通过 TCP 端口 80 进行入站访问 `sg-1a2b3c4d`。源组必须在同一个 VPC 中，或者在对等 VPC 中（需要 VPC 对等连接）。允许的传入流量基于与源安全组相关联实例的私有 IP 地址（而不是公有 IP 或弹性 IP 地址）。

```

aws ec2 authorize-security-group-ingress \
  --group-id sg-1234567890abcdef0 \
  --protocol tcp \
  --port 80 \
  --source-group sg-1a2b3c4d

```

输出：

```

{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-01f4be99110f638a7",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "tcp",
      "FromPort": 80,
      "ToPort": 80,
      "ReferencedGroupInfo": {
        "GroupId": "sg-1a2b3c4d",
        "UserId": "123456789012"
      }
    }
  ]
}

```



### 示例 3：在同一个调用中添加多个规则

以下 `authorize-security-group-ingress` 示例使用 `ip-permissions` 参数添加两个入站规则，一个允许在 TCP 端口 3389 ( RDP ) 上进行入站访问，另一个启用 ping/ICMP。

```
aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0--ip-permissions
=tcp , =3389 , = FromPort 3389 , = "[{=172.31.0.0/16} IpProtocol]" =icmp , =-1 , =-1 , =
"[{ToPort=172.31.0.0/16}]" IpRanges CidrIp IpProtocol FromPort ToPort IpRanges CidrIp
```

输出：

```
{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-00e06e5d3690f29f3",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "tcp",
      "FromPort": 3389,
      "ToPort": 3389,
      "CidrIpv4": "172.31.0.0/16"
    },
    {
      "SecurityGroupRuleId": "sgr-0a133dd4493944b87",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "tcp",
      "FromPort": -1,
      "ToPort": -1,
      "CidrIpv4": "172.31.0.0/16"
    }
  ]
}
```

### 示例 4：为 ICMP 流量添加规则

以下 `authorize-security-group-ingress` 示例使用 `ip-permissions` 参数添加一个入站规则，该规则允许来自任何地方的 ICMP 消息 `Destination Unreachable: Fragmentation Needed and Don't Fragment was Set` ( 类型 3 , 代码 4 )。

```
aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0--ip-permissions
=icmp , =3 , =4 , = "[{=0.0.0.0/0}]] IpProtocol" FromPort ToPort IpRanges CidrIp
```

输出：

```
{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupId": "sgr-0de3811019069b787",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "icmp",
      "FromPort": 3,
      "ToPort": 4,
      "CidrIpv4": "0.0.0.0/0"
    }
  ]
}
```

示例 5：为 IPv6 流量添加规则

以下 `authorize-security-group-ingress` 示例使用 `ip-permissions` 参数添加一个入站规则，该规则允许从 IPv6 范围 `2001:db8:1234:1a00::/64` 进行 SSH 访问（端口 22）。

```
aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0--ip-permissions
=tcp , =22 , =22 , ipv6Ranges= "[{6 IpProtocol =2001: db 8:1234:1 a00:: /64}]]" FromPort ToPort
CidrIpv
```

输出：

```
{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupId": "sgr-0455bc68b60805563",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "tcp",
      "FromPort": 22,
      "ToPort": 22,

```

```

        "CidrIpv6": "2001:db8:1234:1a00::/64"
    }
]
}

```

### 示例 6：为 ICMPv6 流量添加规则

以下 `authorize-security-group-ingress` 示例使用 `ip-permissions` 参数添加一个入站规则，该规则允许来自任何地方的 ICMPv6 流量。

```
aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0--ip-permissions
=icmpv6 , Ipv6Ranges= "[{6:: /0}]" IpProtocol CidrIpv
```

输出：

```

{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-04b612d9363ab6327",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "icmpv6",
      "FromPort": -1,
      "ToPort": -1,
      "CidrIpv6": "::/0"
    }
  ]
}

```

### 示例 7：添加带有描述的规则

以下 `authorize-security-group-ingress` 示例使用 `ip-permissions` 参数添加一个入站规则，该规则允许来自指定 IPv4 地址范围的 RDP 流量。该规则包含描述，可帮助以后识别。

```
aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0--ip-permissions
=tcp , =3389 , =3389 , = "[{=203.0.113.0/24 , FromPortdescription IpProtocol ='RDP 从纽约办公室访问'}]" ToPort IpRanges CidrIp
```

输出：

```
{
```

```

"Return": true,
"SecurityGroupRules": [
  {
    "SecurityGroupRuleId": "sgr-0397bbcc01e974db3",
    "GroupId": "sg-1234567890abcdef0",
    "GroupOwnerId": "123456789012",
    "IsEgress": false,
    "IpProtocol": "tcp",
    "FromPort": 3389,
    "ToPort": 3389,
    "CidrIpv4": "203.0.113.0/24",
    "Description": "RDP access from NY office"
  }
]
}

```

### 示例 8：添加使用前缀列表的入站规则

以下 `authorize-security-group-ingress` 示例使用 `ip-permissions` 参数添加一个入站规则，该规则允许指定前缀列表中 CIDR 范围的所有流量。

```
aws ec2 authorize-security-group-ingress --group-id sg-04a351bfe432d4e71--ip-permissions
=all , =["pl-002dc3ec097de1514"] IpProtocol PrefixListIds PrefixListId
```

输出：

```

{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-09c74b32f677c6c7c",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "-1",
      "FromPort": -1,
      "ToPort": -1,
      "PrefixListId": "pl-0721453c7ac4ec009"
    }
  ]
}

```

有关更多信息，请参阅 Amazon VPC 用户指南中的[安全组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AuthorizeSecurityGroupIngress](#)中的。

## bundle-instance

以下代码示例演示了如何使用 bundle-instance。

### AWS CLI

#### 捆绑实例

此示例*i-1234567890abcdef0*将实例绑定到名为*bundletasks*的存储桶。在为访问密钥 ID 指定值之前，请查看并遵循管理 AWS 访问密钥的最佳实践中的指导。

命令:

```
aws ec2 bundle-instance --instance-id i-1234567890abcdef0 --bucket bundletasks --prefix winami --owner-akid AK12AJEXAMPLE --owner-sak example123example
```

输出:

```
{
  "BundleTask": {
    "UpdateTime": "2015-09-15T13:30:35.000Z",
    "InstanceId": "i-1234567890abcdef0",
    "Storage": {
      "S3": {
        "Prefix": "winami",
        "Bucket": "bundletasks"
      }
    },
    "State": "pending",
    "StartTime": "2015-09-15T13:30:35.000Z",
    "BundleId": "bun-294e041f"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BundleInstance](#)中的。

## cancel-bundle-task

以下代码示例演示了如何使用 cancel-bundle-task。

## AWS CLI

### 取消捆绑包任务

此示例取消捆绑任务**bun-2a4e041c**。

命令:

```
aws ec2 cancel-bundle-task --bundle-id bun-2a4e041c
```

输出:

```
{
  "BundleTask": {
    "UpdateTime": "2015-09-15T13:27:40.000Z",
    "InstanceId": "i-1234567890abcdef0",
    "Storage": {
      "S3": {
        "Prefix": "winami",
        "Bucket": "bundletasks"
      }
    },
    "State": "cancelling",
    "StartTime": "2015-09-15T13:24:35.000Z",
    "BundleId": "bun-2a4e041c"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelBundleTask](#)中的。

## cancel-capacity-reservation-fleets

以下代码示例演示了如何使用 `cancel-capacity-reservation-fleets`。

### AWS CLI

#### 取消容量预留舰队

以下`cancel-capacity-reservation-fleets`示例取消了指定的容量预留队列及其预留的容量。当您取消队列时，其状态将更改为`cancelled`，并且无法再创建新的容量预留。此外，队列中所有单独的容量预留都将被取消，以前以预留容量运行的实例继续以共享容量正常运行。

```
aws ec2 cancel-capacity-reservation-fleets \  
  --capacity-reservation-fleet-ids crf-abcdef01234567890
```

输出：

```
{  
  "SuccessfulFleetCancellations": [  
    {  
      "CurrentFleetState": "cancelling",  
      "PreviousFleetState": "active",  
      "CapacityReservationFleetId": "crf-abcdef01234567890"  
    }  
  ],  
  "FailedFleetCancellations": []  
}
```

有关容量预留队列的更多信息，请参阅 Amazon EC2 用户 [指南中的容量预留队列](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelCapacityReservationFleets](#)中的。

## cancel-capacity-reservation

以下代码示例演示了如何使用 cancel-capacity-reservation。

### AWS CLI

#### 取消容量预留

以下cancel-capacity-reservation示例取消了指定的容量预留。

```
aws ec2 cancel-capacity-reservation \  
  --capacity-reservation-id cr-1234abcd56EXAMPLE
```

输出：

```
{  
  "Return": true  
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[取消容量预留](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelCapacityReservation](#)中的。

## cancel-conversion-task

以下代码示例演示了如何使用 `cancel-conversion-task`。

### AWS CLI

取消实例或卷的有效转换

此示例取消了与任务 ID `import-i-fh 95npoc` 关联的上传。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 cancel-conversion-task --conversion-task-id import-i-fh95npoc
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelConversionTask](#)中的。

## cancel-export-task

以下代码示例演示了如何使用 `cancel-export-task`。

### AWS CLI

取消正在执行的导出任务

此示例取消了任务 ID 为 `export-i-fgelt 0i7` 的活动导出任务。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 cancel-export-task --export-task-id export-i-fgelt0i7
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelExportTask](#)中的。

## cancel-image-launch-permission

以下代码示例演示了如何使用 `cancel-image-launch-permission`。

### AWS CLI

取消与您的亚马逊 Web Services 账户共享 AMI

以下`cancel-image-launch-permission`示例将您的账户从指定 AMI 的启动权限中移除。



```
aws ec2 cancel-image-launch-permission \  
  --image-id ami-0123456789example \  
  --region us-east-1
```

输出：

```
{  
  "Return": true  
}
```

有关更多信息，请参阅亚马逊 EC2 用户指南中的[取消与您的亚马逊 Web Services 账户共享 AMI](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelImageLaunchPermission](#)中的。

## cancel-import-task

以下代码示例演示了如何使用 `cancel-import-task`。

### AWS CLI

#### 取消导入任务

以下`cancel-import-task`示例取消了指定的导入映像任务。

```
aws ec2 cancel-import-task \  
  --import-task-id import-ami-1234567890abcdef0
```

输出：

```
{  
  "ImportTaskId": "import-ami-1234567890abcdef0",  
  "PreviousState": "active",  
  "State": "deleting"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelImportTask](#)中的。

## cancel-reserved-instances-listing

以下代码示例演示了如何使用 `cancel-reserved-instances-listing`。

## AWS CLI

### 取消预留实例上架

以下cancel-reserved-instances-listing示例取消了指定的预留实例列表。

```
aws ec2 cancel-reserved-instances-listing \  
  --reserved-instances-listing-id 5ec28771-05ff-4b9b-aa31-9e57dexample
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelReservedInstancesListing](#)中的。

## cancel-spot-fleet-requests

以下代码示例演示了如何使用 cancel-spot-fleet-requests。

### AWS CLI

#### 示例 1：取消竞价型队列请求并终止关联的实例

以下cancel-spot-fleet-requests示例取消竞价型队列请求并终止相关的按需实例和竞价型实例。

```
aws ec2 cancel-spot-fleet-requests \  
  --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \  
  --terminate-instances
```

输出：

```
{  
  "SuccessfulFleetRequests": [  
    {  
      "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",  
      "CurrentSpotFleetRequestState": "cancelled_terminating",  
      "PreviousSpotFleetRequestState": "active"  
    }  
  ],  
  "UnsuccessfulFleetRequests": []  
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[取消竞价型队列请求](#)。

## 示例 2：在不终止关联实例的情况下取消竞价型队列请求

以下 `cancel-spot-fleet-requests` 示例在不终止关联的按需实例和竞价型实例的情况下取消竞价型队列请求。

```
aws ec2 cancel-spot-fleet-requests \  
  --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \  
  --no-terminate-instances
```

输出：

```
{  
  "SuccessfulFleetRequests": [  
    {  
      "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",  
      "CurrentSpotFleetRequestState": "cancelled_running",  
      "PreviousSpotFleetRequestState": "active"  
    }  
  ],  
  "UnsuccessfulFleetRequests": []  
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[取消竞价型队列请求](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CancelSpotFleetRequests](#) 中的。

## cancel-spot-instance-requests

以下代码示例演示了如何使用 `cancel-spot-instance-requests`。

### AWS CLI

#### 取消竞价型实例请求

此示例命令取消竞价型实例请求。

命令：

```
aws ec2 cancel-spot-instance-requests --spot-instance-request-ids sir-08b93456
```

输出：

```
{
  "CancelledSpotInstanceRequests": [
    {
      "State": "cancelled",
      "SpotInstanceRequestId": "sir-08b93456"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelSpotInstanceRequests](#)中的。

## confirm-product-instance

以下代码示例演示了如何使用 `confirm-product-instance`。

### AWS CLI

#### 确认产品实例

此示例确定指定的产品代码是否与指定的实例相关联。

命令:

```
aws ec2 confirm-product-instance --product-code 774F4FF8 --instance-id
i-1234567890abcdef0
```

输出:

```
{
  "OwnerId": "123456789012"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ConfirmProductInstance](#)中的。

## copy-fpga-image

以下代码示例演示了如何使用 `copy-fpga-image`。

### AWS CLI

#### 复制 Amazon FPGA 镜像

此示例将指定的 AFI 从该us-east-1区域复制到当前区域 (eu-west-1)。

命令:

```
aws ec2 copy-fpga-image --name copy-afi --source-fpga-image-id afi-0d123e123bfc85abc
--source-region us-east-1 --region eu-west-1
```

输出:

```
{
  "FpgaImageId": "afi-06b12350a123fbabc"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CopyFpgaImage](#)中的。

## copy-image

以下代码示例演示了如何使用 copy-image。

### AWS CLI

示例 1：将 AMI 复制到另一个区域

以下copy-image示例命令将指定的 AMI 从该us-west-2区域复制到该us-east-1区域并添加简短描述。

```
aws ec2 copy-image \
  --region us-east-1 \
  --name ami-name \
  --source-region us-west-2 \
  --source-image-id ami-066877671789bd71b \
  --description "This is my copied image."
```

输出:

```
{
  "ImageId": "ami-0123456789abcdefg"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[复制 AMI](#)。

## 示例 2：将 AMI 复制到另一个区域并加密备份快照

以下 `copy-image` 命令将指定的 AMI 从该 `us-west-2` 区域复制到当前区域，并使用指定的 KMS 密钥对备份快照进行加密。

```
aws ec2 copy-image \  
  --source-region us-west-2 \  
  --name ami-name \  
  --source-image-id ami-066877671789bd71b \  
  --encrypted \  
  --kms-key-id alias/my-kms-key
```

输出：

```
{  
  "ImageId": "ami-0123456789abcdefg"  
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[复制 AMI](#)。

## 示例 3：在复制 AMI 时包含用户定义的 AMI 标签

在复制 AMI 时，以下 `copy-image` 命令使用 `--copy-image-tags` 参数复制用户定义的 AMI 标签。

```
aws ec2 copy-image \  
  --region us-east-1 \  
  --name ami-name \  
  --source-region us-west-2 \  
  --source-image-id ami-066877671789bd71b \  
  --description "This is my copied image."  
  --copy-image-tags
```

输出：

```
{  
  "ImageId": "ami-0123456789abcdefg"  
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[复制 AMI](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CopyImage](#) 中的。

## copy-snapshot

以下代码示例演示了如何使用 copy-snapshot。

### AWS CLI

#### 示例 1：将快照复制到另一个区域

以下copy-snapshot示例命令将指定的快照从区域复制到该us-west-2区域，us-east-1并添加了简短的描述。

```
aws ec2 copy-snapshot \  
  --region us-east-1 \  
  --source-region us-west-2 \  
  --source-snapshot-id snap-066877671789bd71b \  
  --description "This is my copied snapshot."
```

输出：

```
{  
  "SnapshotId": "snap-066877671789bd71b"  
}
```

有关更多信息，请参阅 [Amazon EC2 用户指南中的复制 Amazon EBS 快照](#)。

#### 示例 2：复制未加密的快照并加密新快照

以下copy-snapshot命令将指定的未加密快照从该us-west-2区域复制到当前区域，并使用指定的 KMS 密钥对新快照进行加密。

```
aws ec2 copy-snapshot \  
  --source-region us-west-2 \  
  --source-snapshot-id snap-066877671789bd71b \  
  --encrypted \  
  --kms-key-id alias/my-kms-key
```

输出：

```
{  
  "SnapshotId": "snap-066877671789bd71b"  
}
```

有关更多信息，请参阅 [Amazon EC2 用户指南中的复制 Amazon EBS 快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CopySnapshot](#) 中的。

## create-capacity-reservation-fleet

以下代码示例演示了如何使用 create-capacity-reservation-fleet。

### AWS CLI

#### 创建容量预留队列

以下 create-capacity-reservation-fleet 示例为请求中指定的实例类型创建容量预留队列，最多不超过指定的总目标容量。容量预留机群为其预留容量的实例数取决于总目标容量和您在请求中指定的实例类型权重。指定要使用的实例类型以及每个指定实例类型的优先级。

```
aws ec2 create-capacity-reservation-fleet \  
--total-target-capacity 24 \  
--allocation-strategy prioritized \  
--instance-match-criteria open \  
--tenancy default \  
--end-date 2022-12-31T23:59:59.000Z \  
--instance-type-specifications file://instanceTypeSpecification.json
```

instanceTypeSpecification.json 的内容：

```
[  
  {  
    "InstanceType": "m5.xlarge",  
    "InstancePlatform": "Linux/UNIX",  
    "Weight": 3.0,  
    "AvailabilityZone": "us-east-1a",  
    "EbsOptimized": true,  
    "Priority" : 1  
  }  
]
```

输出：

```
{  
  "Status": "submitted",  
  "TotalFulfilledCapacity": 0.0,
```



```
"CapacityReservationFleetId": "crf-abcdef01234567890",  
"TotalTargetCapacity": 24  
}
```

有关容量预留队列的更多信息，请参阅 Amazon EC2 用户 [指南中的容量预留队列](#)。

有关实例类型重量和总目标容量的更多信息，请参阅 Amazon EC2 用户指南中的 [实例类型重量和目  
标总容量](#)。

有关为指定实例类型指定优先级的更多信息，请参阅 Amazon EC2 用户指南中的 [分配策略](#)和 [实例类  
型优先级](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCapacityReservationFleet](#)中的。

## create-capacity-reservation

以下代码示例演示了如何使用 create-capacity-reservation。

### AWS CLI

#### 示例 1：创建容量预留

以下create-capacity-reservation示例在eu-west-1a可用区创建容量预留，您可以在其中启动三个运行 Linux/Unix 操作系统的t2.medium实例。默认情况下，容量预留是在开放实例匹配条件下创建的，不支持临时存储，在您手动取消之前，容量预留将保持活动状态。

```
aws ec2 create-capacity-reservation \  
  --availability-zone eu-west-1a \  
  --instance-type t2.medium \  
  --instance-platform Linux/UNIX \  
  --instance-count 3
```

输出：

```
{  
  "CapacityReservation": {  
    "CapacityReservationId": "cr-1234abcd56EXAMPLE ",  
    "EndDateType": "unlimited",  
    "AvailabilityZone": "eu-west-1a",  
    "InstanceMatchCriteria": "open",  
    "EphemeralStorage": false,  
  }  
}
```

```

    "CreateDate": "2019-08-16T09:27:35.000Z",
    "AvailableInstanceCount": 3,
    "InstancePlatform": "Linux/UNIX",
    "TotalInstanceCount": 3,
    "State": "active",
    "Tenancy": "default",
    "EbsOptimized": false,
    "InstanceType": "t2.medium"
  }
}

```

## 示例 2：创建在指定日期/时间自动结束的容量预留

以下 `create-capacity-reservation` 示例在 `eu-west-1a` 可用区创建容量预留，您可以在其中启动三个运行 Linux/Unix 操作系统的 `m5.large` 实例。此容量预留将于 2019 年 8 月 31 日 23:59:59 自动结束。

```

aws ec2 create-capacity-reservation \
  --availability-zone eu-west-1a \
  --instance-type m5.large \
  --instance-platform Linux/UNIX \
  --instance-count 3 \
  --end-date-type limited \
  --end-date 2019-08-31T23:59:59Z

```

输出：

```

{
  "CapacityReservation": {
    "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
    "EndDateType": "limited",
    "AvailabilityZone": "eu-west-1a",
    "EndDate": "2019-08-31T23:59:59.000Z",
    "InstanceMatchCriteria": "open",
    "EphemeralStorage": false,
    "CreateDate": "2019-08-16T10:15:53.000Z",
    "AvailableInstanceCount": 3,
    "InstancePlatform": "Linux/UNIX",
    "TotalInstanceCount": 3,
    "State": "active",
    "Tenancy": "default",
    "EbsOptimized": false,

```

```
    "InstanceType": "m5.large"
  }
}
```

### 示例 3：创建仅接受目标实例启动的容量预留

以下 `create-capacity-reservation` 示例创建了仅接受目标实例启动的容量预留。

```
aws ec2 create-capacity-reservation \
  --availability-zone eu-west-1a \
  --instance-type m5.large \
  --instance-platform Linux/UNIX \
  --instance-count 3 \
  --instance-match-criteria targeted
```

输出：

```
{
  "CapacityReservation": {
    "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
    "EndDateType": "unlimited",
    "AvailabilityZone": "eu-west-1a",
    "InstanceMatchCriteria": "targeted",
    "EphemeralStorage": false,
    "CreateDate": "2019-08-16T10:21:57.000Z",
    "AvailableInstanceCount": 3,
    "InstancePlatform": "Linux/UNIX",
    "TotalInstanceCount": 3,
    "State": "active",
    "Tenancy": "default",
    "EbsOptimized": false,
    "InstanceType": "m5.large"
  }
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[创建容量预留](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateCapacityReservation](#) 中的。

## create-carrier-gateway

以下代码示例演示了如何使用 `create-carrier-gateway`。

## AWS CLI

### 创建运营商网关

以下`create-carrier-gateway`示例为指定的 VPC 创建运营商网关。

```
aws ec2 create-carrier-gateway \  
  --vpc-id vpc-0c529aEXAMPLE1111
```

输出：

```
{  
  "CarrierGateway": {  
    "CarrierGatewayId": "cagw-0465cdEXAMPLE1111",  
    "VpcId": "vpc-0c529aEXAMPLE1111",  
    "State": "pending",  
    "OwnerId": "123456789012"  
  }  
}
```

有关更多信息，请参阅《Wavelength AWS 用户指南》中的[运营商网关](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateCarrierGateway](#)中的。

## `create-client-vpn-endpoint`

以下代码示例演示了如何使用 `create-client-vpn-endpoint`。

## AWS CLI

### 创建 Client VPN 端点

以下`create-client-vpn-endpoint`示例创建了一个使用相互身份验证的 Client VPN 端点，并为客户端 CIDR 块指定一个值。

```
aws ec2 create-client-vpn-endpoint \  
  --client-cidr-block "172.31.0.0/16" \  
  --server-certificate-arn arn:aws:acm:ap-south-1:123456789012:certificate/  
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \  
  --authentication-options Type=certificate-  
authentication,MutualAuthentication={ClientRootCertificateChainArn=arn:aws:acm:ap-  
south-1:123456789012:certificate/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE} \  
  \
```

```
--connection-log-options Enabled=false
```

输出：

```
{
  "ClientVpnEndpointId": "cvpn-endpoint-123456789123abcde",
  "Status": {
    "Code": "pending-associate"
  },
  "DnsName": "cvpn-endpoint-123456789123abcde.prod.clientvpn.ap-
south-1.amazonaws.com"
}
```

有关更多信息，请参阅《[客户端 VPN 管理员指南](#)》中的[AWS Client VPN 端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateClientVpnEndpoint](#) 中的。

## create-client-vpn-route

以下代码示例演示了如何使用 `create-client-vpn-route`。

### AWS CLI

为 Client VPN 终端节点创建路由

以下 `create-client-vpn-route` 示例为 Client VPN 终端节点的指定子网添加了一条通往互联网的路由 (`0.0.0.0/0`)。

```
aws ec2 create-client-vpn-route \
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde \
  --destination-cidr-block 0.0.0.0/0 \
  --target-vpc-subnet-id subnet-0123456789abcabca
```

输出：

```
{
  "Status": {
    "Code": "creating"
  }
}
```

有关更多信息，请参阅《[AWS Client VPN 管理员指南](#)》中的[路由](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateClientVpnRoute](#)中的。

## create-coip-cidr

以下代码示例演示了如何使用 create-coip-cidr。

### AWS CLI

创建一系列客户拥有的 IP (CoIP) 地址

以下create-coip-cidr示例在指定的 CoIP 池中创建指定范围的 CoIP 地址。

```
aws ec2 create-coip-cidr \  
  --cidr 15.0.0.0/24 \  
  --coip-pool-id ipv4pool-coip-1234567890abcdefg
```

输出：

```
{  
  "CoipCidr": {  
    "Cidr": "15.0.0.0/24",  
    "CoipPoolId": "ipv4pool-coip-1234567890abcdefg",  
    "LocalGatewayRouteTableId": "lgw-rtb-abcdefg1234567890"  
  }  
}
```

有关更多信息，请参阅 [《AWS Outposts 用户指南》](#) 中的客户拥有的 IP 地址。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCoipCidr](#)中的。

## create-coip-pool

以下代码示例演示了如何使用 create-coip-pool。

### AWS CLI

创建客户拥有的 IP (CoIP) 地址池

以下create-coip-pool示例为指定的本地网关路由表中的 CoIP 地址创建 CoIP 池。

```
aws ec2 create-coip-pool \  
  --local-gateway-route-table-id lgw-rtb-abcdefg1234567890
```

输出：

```
{
  "CoipPool": {
    "PoolId": "ipv4pool-coip-1234567890abcdefg",
    "LocalGatewayRouteTableId": "lgw-rtb-abcdefg1234567890",
    "PoolArn": "arn:aws:ec2:us-west-2:123456789012:coip-pool/ipv4pool-
coip-1234567890abcdefg"
  }
}
```

有关更多信息，请参阅 [《AWS Outposts 用户指南》](#) 中的客户拥有的 IP 地址。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCoipPool](#)中的。

## create-customer-gateway

以下代码示例演示了如何使用 create-customer-gateway。

### AWS CLI

#### 创建客户网关

此示例为其外部接口创建了一个具有指定 IP 地址的客户网关。

命令：

```
aws ec2 create-customer-gateway --type ipsec.1 --public-ip 12.1.2.3 --bgp-asn 65534
```

输出：

```
{
  "CustomerGateway": {
    "CustomerGatewayId": "cgw-0e11f167",
    "IpAddress": "12.1.2.3",
    "State": "available",
    "Type": "ipsec.1",
    "BgpAsn": "65534"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCustomerGateway](#)中的。

## create-default-subnet

以下代码示例演示了如何使用 `create-default-subnet`。

### AWS CLI

#### 创建默认子网

此示例在可用区中创建默认子网 `us-east-2a`。

命令:

```
aws ec2 create-default-subnet --availability-zone us-east-2a
```

```
{
  "Subnet": {
    "AvailabilityZone": "us-east-2a",
    "Tags": [],
    "AvailableIpAddressCount": 4091,
    "DefaultForAz": true,
    "Ipv6CidrBlockAssociationSet": [],
    "VpcId": "vpc-1a2b3c4d",
    "State": "available",
    "MapPublicIpOnLaunch": true,
    "SubnetId": "subnet-1122aabb",
    "CidrBlock": "172.31.32.0/20",
    "AssignIpv6AddressOnCreation": false
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDefaultSubnet](#)中的。

## create-default-vpc

以下代码示例演示了如何使用 `create-default-vpc`。

### AWS CLI

#### 创建默认 VPC

此示例创建了一个默认 VPC。

命令:



```
aws ec2 create-default-vpc
```

输出：

```
{
  "Vpc": {
    "VpcId": "vpc-8eaae5ea",
    "InstanceTenancy": "default",
    "Tags": [],
    "Ipv6CidrBlockAssociationSet": [],
    "State": "pending",
    "DhcpOptionsId": "dopt-af0c32c6",
    "CidrBlock": "172.31.0.0/16",
    "IsDefault": true
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDefaultVpc](#)中的。

## create-dhcp-options

以下代码示例演示了如何使用 create-dhcp-options。

### AWS CLI

#### 创建一组 DHCP 选项

以下create-dhcp-options示例创建了一组 DHCP 选项，用于指定域名、域名服务器和 NetBIOS 节点类型。

```
aws ec2 create-dhcp-options \
  --dhcp-configuration \
    "Key=domain-name-servers,Values=10.2.5.1,10.2.5.2" \
    "Key=domain-name,Values=example.com" \
    "Key=netbios-node-type,Values=2"
```

输出：

```
{
  "DhcpOptions": {
    "DhcpConfigurations": [
```

```
{
  "Key": "domain-name",
  "Values": [
    {
      "Value": "example.com"
    }
  ]
},
{
  "Key": "domain-name-servers",
  "Values": [
    {
      "Value": "10.2.5.1"
    },
    {
      "Value": "10.2.5.2"
    }
  ]
},
{
  "Key": "netbios-node-type",
  "Values": [
    {
      "Value": "2"
    }
  ]
}
],
"DhcpOptionsId": "dopt-06d52773eff4c55f3"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDhcpOptions](#)中的。

## create-egress-only-internet-gateway

以下代码示例演示了如何使用 `create-egress-only-internet-gateway`。

### AWS CLI

创建仅限出口的互联网网关

此示例为指定的 VPC 创建仅限出站的互联网网关。

命令:

```
aws ec2 create-egress-only-internet-gateway --vpc-id vpc-0c62a468
```

输出:

```
{
  "EgressOnlyInternetGateway": {
    "EgressOnlyInternetGatewayId": "eigw-015e0e244e24dfe8a",
    "Attachments": [
      {
        "State": "attached",
        "VpcId": "vpc-0c62a468"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateEgressOnlyInternetGateway](#)中的。

## create-fleet

以下代码示例演示了如何使用 create-fleet。

### AWS CLI

示例 1：创建启动竞价型实例作为默认购买模式的 EC2 队列

以下create-fleet示例使用启动队列所需的最低参数创建 EC2 队列：启动模板、目标容量和默认购买模式。启动模板由其启动模板 ID 和版本号标识。队列的目标容量为 2 个实例，默认购买模式为spot，这会导致队列启动 2 个竞价型实例。

在创建 EC2 队列时，可以使用 JSON 文件指定要启动的实例的有关信息。

```
aws ec2 create-fleet \
  --cli-input-json file:///file_name.json
```

file\_name.json 的内容：

```
{
```

```
"LaunchTemplateConfigs": [  
  {  
    "LaunchTemplateSpecification": {  
      "LaunchTemplateId": "lt-0e8c754449b27161c",  
      "Version": "1"  
    }  
  },  
],  
"TargetCapacitySpecification": {  
  "TotalTargetCapacity": 2,  
  "DefaultTargetCapacityType": "spot"  
}  
}
```

输出：

```
{  
  "FleetId": "fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE"  
}
```

## 示例 2：创建启动按需实例作为默认购买模式的 EC2 队列

以下 `create-fleet` 示例使用启动队列所需的最低参数创建 EC2 队列：启动模板、目标容量和默认购买模式。启动模板由其启动模板 ID 和版本号标识。队列的目标容量为 2 个实例，默认购买模式为 `on-demand`，这会导致队列启动 2 个按需实例。

在创建 EC2 队列时，可以使用 JSON 文件指定要启动的实例的有关信息。

```
aws ec2 create-fleet \  
  --cli-input-json file:///file_name.json
```

`file_name.json` 的内容：

```
{  
  "LaunchTemplateConfigs": [  
    {  
      "LaunchTemplateSpecification": {  
        "LaunchTemplateId": "lt-0e8c754449b27161c",  
        "Version": "1"  
      }  
    }  
  ],  
}
```

```
"TargetCapacitySpecification": {
  "TotalTargetCapacity": 2,
  "DefaultTargetCapacityType": "on-demand"
}
```

输出：

```
{
  "FleetId": "fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE"
}
```

### 示例 3：创建以按需实例作为主容量启动的 EC2 队列

以下 `create-fleet` 示例创建了一个 EC2 队列，该队列指定队列的总目标容量为 2 个实例，目标容量为 1 个按需实例。默认的购买模式是 `spot`。队列按指定启动 1 个按需实例，但需要再启动一个实例才能达到目标总容量。差额的购买模型计算为 `TotalTargetCapacity-OnDemandTargetCapacity = DefaultTargetCapacityType`，这会导致队列启动 1 个竞价型实例。

在创建 EC2 队列时，可以使用 JSON 文件指定要启动的实例的有关信息。

```
aws ec2 create-fleet \
  --cli-input-json file://file_name.json
```

`file_name.json` 的内容：

```
{
  "LaunchTemplateConfigs": [
    {
      "LaunchTemplateSpecification": {
        "LaunchTemplateId": "lt-0e8c754449b27161c",
        "Version": "1"
      }
    }
  ],
  "TargetCapacitySpecification": {
    "TotalTargetCapacity": 2,
    "OnDemandTargetCapacity": 1,
    "DefaultTargetCapacityType": "spot"
  }
}
```

```
}
```

输出：

```
{
  "FleetId": "fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE"
}
```

#### 示例 4：创建使用最低价格分配策略启动竞价型实例的 EC2 队列

如果未指定 Spot 实例的分配策略，则使用默认分配策略 lowest-price。以下 create-fleet 示例使用 lowest-price 分配策略创建 EC2 队列。覆盖启动模板的三个启动规范有不同的实例类型，但有相同的权重容量和子网。目标总容量为 2 个实例，默认购买模式为 spot。EC2 实例集按启动规范的最低价格实例类型启动两个竞价型实例。

在创建 EC2 队列时，可以使用 JSON 文件指定要启动的实例的有关信息。

```
aws ec2 create-fleet \
  --cli-input-json file://file_name.jsonContents of file_name.json::

{
  "LaunchTemplateConfigs": [
    {
      "LaunchTemplateSpecification": {
        "LaunchTemplateId": "lt-0e8c754449b27161c",
        "Version": "1"
      },
      "Overrides": [
        {
          "InstanceType": "c4.large",
          "WeightedCapacity": 1,
          "SubnetId": "subnet-a4f6c5d3"
        },
        {
          "InstanceType": "c3.large",
          "WeightedCapacity": 1,
          "SubnetId": "subnet-a4f6c5d3"
        },
        {
          "InstanceType": "c5.large",
          "WeightedCapacity": 1,
          "SubnetId": "subnet-a4f6c5d3"
        }
      ]
    }
  ]
}
```

```

    }
  ]
}
],
"TargetCapacitySpecification": {
  "TotalTargetCapacity": 2,
  "DefaultTargetCapacityType": "spot"
}
}

```

输出：

```

{
  "FleetId": "fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFleet](#)中的。

## create-flow-logs

以下代码示例演示了如何使用 create-flow-logs。

### AWS CLI

#### 示例 1：创建流日志

以下create-flow-logs示例创建了一个流日志，用于捕获指定网络接口的所有被拒绝的流量。使用指定 IAM 角色中的权限将流 CloudWatch 日志传输到 Logs 中的日志组。

```

aws ec2 create-flow-logs \
  --resource-type NetworkInterface \
  --resource-ids eni-11223344556677889 \
  --traffic-type REJECT \
  --log-group-name my-flow-logs \
  --deliver-logs-permission-arn arn:aws:iam::123456789101:role/publishFlowLogs

```

输出：

```

{
  "ClientToken": "so0eNA2uSHUN1HI0S2cJ305GuIX1CezaRdGtexample",
  "FlowLogIds": [

```

```
    "f1-12345678901234567"  
  ],  
  "Unsuccessful": []  
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 [VPC 流日志](#)。

#### 示例 2：使用自定义格式创建流日志

以下create-flow-logs示例创建了一个流日志，用于捕获指定 VPC 的所有流量并将流日志传输到 Amazon S3 存储桶。--log-format 参数指定流日志记录的自定义格式。要在 Windows 上运行此命令，请将单引号 (') 更改为双引号 (")。

```
aws ec2 create-flow-logs \  
  --resource-type VPC \  
  --resource-ids vpc-00112233344556677 \  
  --traffic-type ALL \  
  --log-destination-type s3 \  
  --log-destination arn:aws:s3:::flow-log-bucket/my-custom-flow-logs/ \  
  --log-format '${version} ${vpc-id} ${subnet-id} ${instance-id} ${srcaddr}  
  ${dstaddr} ${srcport} ${dstport} ${protocol} ${tcp-flags} ${type} ${pkt-srcaddr}  
  ${pkt-dstaddr}'
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 [VPC 流日志](#)。

#### 示例 3：创建最大聚合间隔为一分钟的流日志

以下create-flow-logs示例创建了一个流日志，用于捕获指定 VPC 的所有流量并将流日志传输到 Amazon S3 存储桶。该--max-aggregation-interval参数将最大聚合间隔指定为 60 秒（1 分钟）。

```
aws ec2 create-flow-logs \  
  --resource-type VPC \  
  --resource-ids vpc-00112233344556677 \  
  --traffic-type ALL \  
  --log-destination-type s3 \  
  --log-destination arn:aws:s3:::flow-log-bucket/my-custom-flow-logs/ \  
  --max-aggregation-interval 60
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 [VPC 流日志](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFlowLogs](#)中的。



## create-fpga-image

以下代码示例演示了如何使用 create-fpga-image。

### AWS CLI

#### 创建 Amazon FPGA 镜像

此示例使用指定存储桶中的指定压缩包创建 AFI。

命令:

```
aws ec2 create-fpga-image --name my-afi --description test-afi --input-storage-location Bucket=my-fpga-bucket,Key=dcg/17_12_22-103226.Developer_CL.tar --logs-storage-location Bucket=my-fpga-bucket,Key=logs
```

输出 :

```
{
  "FpgaImageId": "afi-0d123e123bfc85abc",
  "FpgaImageGlobalId": "agfi-123cb27b5e84a0abc"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFpgaImage](#)中的。

## create-image

以下代码示例演示了如何使用 create-image。

### AWS CLI

示例 1：从 Amazon EBS 支持的实例创建 AMI

以下create-image示例从指定实例创建 AMI。

```
aws ec2 create-image \
  --instance-id i-1234567890abcdef0 \
  --name "My server" \
  --description "An AMI for my server"
```

输出 :

```
{
  "ImageId": "ami-abcdef01234567890"
}
```

有关为 AMI 指定块储存设备映射的更多信息，请参阅 Amazon EC2 用户指南中的[为 AMI 指定块储存设备映射](#)。

示例 2：在不重启的情况下从 Amazon EBS 支持的实例创建 AMI

以下 `create-image` 示例创建了一个 AMI 并设置了 `--no-reboot` 参数，这样在创建映像之前就不会重启实例。

```
aws ec2 create-image \
  --instance-id i-1234567890abcdef0 \
  --name "My server" \
  --no-reboot
```

输出：

```
{
  "ImageId": "ami-abcdef01234567890"
}
```

有关为 AMI 指定块储存设备映射的更多信息，请参阅 Amazon EC2 用户指南中的[为 AMI 指定块储存设备映射](#)。

示例 3：在创建时标记 AMI 和快照

以下 `create-image` 示例创建了一个 AMI，并使用相同的标签为 AMI 和快照添加了标签 `cost-center=cc123`

```
aws ec2 create-image \
  --instance-id i-1234567890abcdef0 \
  --name "My server" \
  --tag-specifications "ResourceType=image,Tags=[{Key=cost-center,Value=cc123}]"
  "ResourceType=snapshot,Tags=[{Key=cost-center,Value=cc123}]"
```

输出：

```
{
  "ImageId": "ami-abcdef01234567890"
}
```

```
}
```

有关在创建时为资源[添加标签的更多信息](#)，请参阅 [Amazon EC2 用户指南中的在资源创建时添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateImage](#)中的。

## create-instance-connect-endpoint

以下代码示例演示了如何使用 create-instance-connect-endpoint。

### AWS CLI

#### 创建 EC2 实例 Connect 终端节点

以下create-instance-connect-endpoint示例在指定子网中创建 EC2 Instance Connect 终端节点。

```
aws ec2 create-instance-connect-endpoint \  
  --region us-east-1 \  
  --subnet-id subnet-0123456789example
```

输出：

```
{  
  "VpcId": "vpc-0123abcd",  
  "InstanceConnectEndpointArn": "arn:aws:ec2:us-east-1:111111111111:instance-  
connect-endpoint/eice-0123456789example",  
  "AvailabilityZone": "us-east-1a",  
  "NetworkInterfaceIds": [  
    "eni-0123abcd"  
  ],  
  "PreserveClientIp": true,  
  "Tags": [],  
  "FipsDnsName": "eice-0123456789example.0123abcd.fips.ec2-instance-connect-  
endpoint.us-east-1.amazonaws.com",  
  "StateMessage": "",  
  "State": "create-complete",  
  "DnsName": "eice-0123456789example.0123abcd.ec2-instance-connect-endpoint.us-  
east-1.amazonaws.com",  
  "SubnetId": "subnet-0123abcd",  
  "OwnerId": "111111111111",  
  "SecurityGroupIds": [  
    "sg-0123abcd"  
  ]  
}
```

```
    "sg-0123abcd"
  ],
  "InstanceConnectEndpointId": "eice-0123456789example",
  "CreatedAt": "2023-04-07T15:43:53.000Z"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的创建 EC2 Instance [Connect 终端节点](#)。

• 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateInstanceConnectEndpoint](#) 中的。

## create-instance-event-window

以下代码示例演示了如何使用 create-instance-event-window。

### AWS CLI

示例 1：创建具有时间范围的事件窗口

以下 create-instance-event-window 示例创建了一个具有时间范围的事件窗口。您不能同时指定 cron-expression 参数。

```
aws ec2 create-instance-event-window \
  --region us-east-1 \
  --time-range StartWeekDay=monday,StartHour=2,EndWeekDay=wednesday,EndHour=8 \
  --tag-specifications "ResourceType=instance-event-
window,Tags=[{Key=K1,Value=V1}]" \
  --name myEventWindowName
```

输出：

```
{
  "InstanceEventWindow": {
    "InstanceEventWindowId": "iew-0abcdef1234567890",
    "TimeRanges": [
      {
        "StartWeekDay": "monday",
        "StartHour": 2,
        "EndWeekDay": "wednesday",
        "EndHour": 8
      }
    ]
  },
  "Name": "myEventWindowName",
  "State": "creating",
}
```

```
    "Tags": [
      {
        "Key": "K1",
        "Value": "V1"
      }
    ]
  }
}
```

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中计划事件部分的[注意事项](#)。

#### 示例 2：使用 cron 表达式创建事件窗口

以下 `create-instance-event-window` 示例创建了一个带有 cron 表达式的事件窗口。您不能同时指定 `time-range` 参数。

```
aws ec2 create-instance-event-window \
  --region us-east-1 \
  --cron-expression "* 21-23 * * 2,3" \
  --tag-specifications "ResourceType=instance-event-
window,Tags=[{Key=K1,Value=V1}]" \
  --name myEventWindowName
```

输出：

```
{
  "InstanceEventWindow": {
    "InstanceEventWindowId": "iew-0abcdef1234567890",
    "Name": "myEventWindowName",
    "CronExpression": "* 21-23 * * 2,3",
    "State": "creating",
    "Tags": [
      {
        "Key": "K1",
        "Value": "V1"
      }
    ]
  }
}
```

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中计划事件部分的[注意事项](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateInstanceEventWindow](#) 中的。

## create-instance-export-task

以下代码示例演示了如何使用 `create-instance-export-task`。

### AWS CLI

#### 导出实例

此示例命令创建一个任务，用于将实例 `i-1234567890abcdef0` 导出到 Amazon S3 存储桶 `myexportbucket`。

#### 命令:

```
aws ec2 create-instance-export-task --description "RHEL5 instance" --instance-id i-1234567890abcdef0 --target-environment vmware --export-to-s3-task DiskImageFormat=vmdk,ContainerFormat=ova,S3Bucket=myexportbucket,S3Prefix=RHEL5
```

#### 输出:

```
{
  "ExportTask": {
    "State": "active",
    "InstanceExportDetails": {
      "InstanceId": "i-1234567890abcdef0",
      "TargetEnvironment": "vmware"
    },
    "ExportToS3Task": {
      "S3Bucket": "myexportbucket",
      "S3Key": "RHEL5export-i-fh8sjjsq.ova",
      "DiskImageFormat": "vmdk",
      "ContainerFormat": "ova"
    },
    "Description": "RHEL5 instance",
    "ExportTaskId": "export-i-fh8sjjsq"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateInstanceExportTask](#)中的。

## create-internet-gateway

以下代码示例演示了如何使用 `create-internet-gateway`。

## AWS CLI

### 创建互联网网关

以下`create-internet-gateway`示例使用标签创建互联网网关Name=my-igw。

```
aws ec2 create-internet-gateway \  
  --tag-specifications ResourceType=internet-gateway,Tags=[{Key=Name,Value=my-igw}]
```

输出：

```
{  
  "InternetGateway": {  
    "Attachments": [],  
    "InternetGatewayId": "igw-0d0fb496b3994d755",  
    "OwnerId": "123456789012",  
    "Tags": [  
      {  
        "Key": "Name",  
        "Value": "my-igw"  
      }  
    ]  
  }  
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的[互联网网关](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateInternetGateway](#)中的。

### **create-ipam-pool**

以下代码示例演示了如何使用 `create-ipam-pool`。

## AWS CLI

### 创建 IPAM 池

以下`create-ipam-pool`示例创建了一个 IPAM 池。

(Linux)：

```
aws ec2 create-ipam-pool \
  --ipam-scope-id ipam-scope-02fc38cd4c48e7d38 \
  --address-family ipv4 \
  --auto-import \
  --allocation-min-netmask-length 16 \
  --allocation-max-netmask-length 26 \
  --allocation-default-netmask-length 24 \
  --allocation-resource-tags "Key=Environment,Value=Preprod" \
  --tag-specifications 'ResourceType=ipam-pool,Tags=[{Key=Name,Value="Preprod
pool"}]'
```

( 视窗 ) :

```
aws ec2 create-ipam-pool ^
  --ipam-scope-id ipam-scope-02fc38cd4c48e7d38 ^
  --address-family ipv4 ^
  --auto-import ^
  --allocation-min-netmask-length 16 ^
  --allocation-max-netmask-length 26 ^
  --allocation-default-netmask-length 24 ^
  --allocation-resource-tags "Key=Environment,Value=Preprod" ^
  --tag-specifications ResourceType=ipam-pool,Tags=[{Key=Name,Value="Preprod
pool"}]
```

输出 :

```
{
  "IpamPool": {
    "OwnerId": "123456789012",
    "IpamPoolId": "ipam-pool-0533048da7d823723",
    "IpamPoolArn": "arn:aws:ec2::123456789012:ipam-pool/ipam-
pool-0533048da7d823723",
    "IpamScopeArn": "arn:aws:ec2::123456789012:ipam-scope/ipam-
scope-02fc38cd4c48e7d38",
    "IpamScopeType": "private",
    "IpamArn": "arn:aws:ec2::123456789012:ipam/ipam-08440e7a3acde3908",
    "IpamRegion": "us-east-1",
    "Locale": "None",
    "PoolDepth": 1,
    "State": "create-in-progress",
    "AutoImport": true,
    "AddressFamily": "ipv4",
```



```
    "AllocationMinNetmaskLength": 16,
    "AllocationMaxNetmaskLength": 26,
    "AllocationDefaultNetmaskLength": 24,
    "AllocationResourceTags": [
      {
        "Key": "Environment",
        "Value": "Preprod"
      }
    ],
    "Tags": [
      {
        "Key": "Name",
        "Value": "Preprod pool"
      }
    ]
  }
}
```

有关更多信息，请参阅 Amazon VPC IPAM 用户 [指南中的 IP 地址配置计划](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateIpamPool](#) 中的。

## create-ipam-resource-discovery

以下代码示例演示了如何使用 create-ipam-resource-discovery。

### AWS CLI

#### 创建资源发现

在此示例中，您是一名委派的 IPAM 管理员，他想要创建资源发现并与另一个 AWS 组织的 IPAM 管理员共享，以便其他组织中的管理员可以管理和监控您组织中资源的 IP 地址。

#### 重要提示

此示例包括 --region 和 --operating-regions 选项，因为虽然它们是可选的，但必须以特定的方式配置它们才能成功地将资源发现与 IPAM 集成。\* --operating-regions 必须与您拥有希望 IPAM 发现的资源的区域相匹配。如果您不希望 IPAM 管理 IP 地址的区域（例如出于合规性原因），请不要将其包括在内。\* --region 必须与您要关联的 IPAM 的主区域匹配。您必须在创建 IPAM 的同一区域中创建资源发现。例如，如果您要关联的 IPAM 是在 us-east-1 中创建的，请在请求 --region us-east-1 中包含这些选项。如果您未指定，则 --region 和 --operating-regions 选项均默认为您运行命令的区域。

在此示例中，我们要整合的 IPAM 的运行区域包括 us-west-1、us-west-2 和 ap-south-1。当我们创建资源发现时，我们希望 IPAM 发现 us-west-1 和 ap-south-1 中的资源 IP 地址，us-west-2 但不是 ap-south-1。因此，我们只包含 `--operating-regions RegionName='us-west-1' RegionName='us-west-2'` 在请求中。

以下 `create-ipam-resource-discovery` 示例创建 IPAM 资源发现。

```
aws ec2 create-ipam-resource-discovery \
  --description 'Example-resource-discovery' \
  --tag-specifications 'ResourceType=ipam-resource-discovery,Tags=[{Key=cost-center,Value=cc123}]' \
  --operating-regions RegionName='us-west-1' RegionName='us-west-2' \
  --region us-east-1
```

输出：

```
{
  "IpamResourceDiscovery": {
    "OwnerId": "149977607591",
    "IpamResourceDiscoveryId": "ipam-res-disco-0257046d8aa78b8bc",
    "IpamResourceDiscoveryArn": "arn:aws:ec2::149977607591:ipam-resource-discovery/ipam-res-disco-0257046d8aa78b8bc",
    "IpamResourceDiscoveryRegion": "us-east-1",
    "Description": "'Example-resource-discovery'",
    "OperatingRegions": [
      {"RegionName": "us-west-1"},
      {"RegionName": "us-west-2"},
      {"RegionName": "us-east-1"}
    ],
    "IsDefault": false,
    "State": "create-in-progress",
    "Tags": [
      {
        "Key": "cost-center",
        "Value": "cc123"
      }
    ]
  }
}
```

创建资源发现后，您可能需要与其他 IPAM 委托管理员共享，您可以这样做。[create-resource-share](#) 有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的 [将 IPAM 与组织以外的账户集成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateIpamResourceDiscovery](#) 中的。

## create-ipam-scope

以下代码示例演示了如何使用 create-ipam-scope。

### AWS CLI

#### 创建 IPAM 作用域

以下 create-ipam-scope 示例创建 IPAM 作用域。

(Linux) :

```
aws ec2 create-ipam-scope \  
  --ipam-id ipam-08440e7a3acde3908 \  
  --description "Example description" \  
  --tag-specifications 'ResourceType=ipam-scope,Tags=[{Key=Name,Value="Example  
name value"}]'
```

( 视窗 ) :

```
aws ec2 create-ipam-scope ^  
  --ipam-id ipam-08440e7a3acde3908 ^  
  --description "Example description" ^  
  --tag-specifications ResourceType=ipam-scope,Tags=[{Key=Name,Value="Example name  
value"}]
```

输出 :

```
{  
  "IpamScope": {  
    "OwnerId": "123456789012",  
    "IpamScopeId": "ipam-scope-01c1ebab2b63bd7e4",  
    "IpamScopeArn": "arn:aws:ec2::123456789012:ipam-scope/ipam-  
scope-01c1ebab2b63bd7e4",  
    "IpamArn": "arn:aws:ec2::123456789012:ipam/ipam-08440e7a3acde3908",  
    "IpamRegion": "us-east-1",  
    "IpamScopeType": "private",  
    "IsDefault": false,  
    "Description": "Example description",  
    "PoolCount": 0,  
    "State": "create-in-progress",  
    "Tags": [  
      {
```

```

        "Key": "Name",
        "Value": "Example name value"
    }
  ]
}

```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的[创建其他范围](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateIpamScope](#) 中的。

## create-ipam

以下代码示例演示了如何使用 create-ipam。

### AWS CLI

#### 创建 IPAM

以下 create-ipam 示例创建了一个 IPAM。

(Linux) :

```

aws ec2 create-ipam \
  --description "Example description" \
  --operating-regions "RegionName=us-east-2" "RegionName=us-west-1" \
  --tag-specifications 'ResourceType=ipam,Tags=[{Key=Name,Value=ExampleIPAM}]'

```

( 视窗 ) :

```

aws ec2 create-ipam ^
  --description "Example description" ^
  --operating-regions "RegionName=us-east-2" "RegionName=us-west-1" ^
  --tag-specifications ResourceType=ipam,Tags=[{Key=Name,Value=ExampleIPAM}]

```

输出 :

```

{
  "Ipam": {
    "OwnerId": "123456789012",
    "IpamId": "ipam-036486dfa6af58ee0",
    "IpamArn": "arn:aws:ec2::123456789012:ipam/ipam-036486dfa6af58ee0",
    "IpamRegion": "us-east-1",
  }
}

```

```

    "PublicDefaultScopeId": "ipam-scope-071b8042b0195c183",
    "PrivateDefaultScopeId": "ipam-scope-0807405dece705a30",
    "ScopeCount": 2,
    "OperatingRegions": [
      {
        "RegionName": "us-east-2"
      },
      {
        "RegionName": "us-west-1"
      },
      {
        "RegionName": "us-east-1"
      }
    ],
    "State": "create-in-progress",
    "Tags": [
      {
        "Key": "Name",
        "Value": "ExampleIPAM"
      }
    ]
  }
}

```

有关更多信息，请参阅 Amazon V [PC IPAM 用户指南中的创建 IPAM](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateIpam](#) 中的。

## create-key-pair

以下代码示例演示了如何使用 create-key-pair。

### AWS CLI

#### 创建密钥对

本示例将创建一个名为 MyKeyPair 的密钥对。

命令:

```
aws ec2 create-key-pair --key-name MyKeyPair
```

输出是私有密钥和密钥指纹的 ASCII 版本。需要将密钥保存到文件中。

有关更多信息，请参阅《AWS 命令行界面用户指南》中的“使用密钥对”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateKeyPair](#)中的。

## create-launch-template-version

以下代码示例演示了如何使用 create-launch-template-version。

### AWS CLI

#### 创建启动模板版本

此示例基于启动模板的版本 1 创建新的启动模板版本并指定了不同的 AMI ID。

命令:

```
aws ec2 create-launch-template-version --launch-template-id lt-0abcd290751193123
--version-description WebVersion2 --source-version 1 --launch-template-data
'{"ImageId":"ami-c998b6b2"}'
```

输出:

```
{
  "LaunchTemplateVersion": {
    "VersionDescription": "WebVersion2",
    "LaunchTemplateId": "lt-0abcd290751193123",
    "LaunchTemplateName": "WebServers",
    "VersionNumber": 2,
    "CreatedBy": "arn:aws:iam::123456789012:root",
    "LaunchTemplateData": {
      "ImageId": "ami-c998b6b2",
      "InstanceType": "t2.micro",
      "NetworkInterfaces": [
        {
          "Ipv6Addresses": [
            {
              "Ipv6Address": "2001:db8:1234:1a00::123"
            }
          ],
          "DeviceIndex": 0,
          "SubnetId": "subnet-7b16de0c",
          "AssociatePublicIpAddress": true
        }
      ]
    }
  }
}
```

```

    ]
  },
  "DefaultVersion": false,
  "CreateTime": "2017-12-01T13:35:46.000Z"
}
}

```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateLaunchTemplateVersion](#) 中的。

## create-launch-template

以下代码示例演示了如何使用 create-launch-template。

### AWS CLI

#### 示例 1：创建启动模板

以下 create-launch-template 示例创建一个启动模板，该模板指定启动实例的子网，为实例分配公有 IP 地址和 IPv6 地址，并为实例创建标签。

```

aws ec2 create-launch-template \
  --launch-template-name TemplateForWebServer \
  --version-description WebVersion1 \
  --launch-template-data '{"NetworkInterfaces":
[{"AssociatePublicIpAddress":true,"DeviceIndex":0,"Ipv6AddressCount":1,"SubnetId":"subnet-7b
[{"ResourceType":"instance","Tags":[{"Key":"purpose","Value":"webserver"}]}]}'

```

输出：

```

{
  "LaunchTemplate": {
    "LatestVersionNumber": 1,
    "LaunchTemplateId": "lt-01238c059e3466abc",
    "LaunchTemplateName": "TemplateForWebServer",
    "DefaultVersionNumber": 1,
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
    "CreateTime": "2019-01-27T09:13:24.000Z"
  }
}

```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的“从启动模板启动实例”。有关引用 JSON 格式参数的信息，请参阅《AWS 命令行界面用户指南》中的“引用字符串”。

## 示例 2：为 Amazon EC2 Auto Scaling 创建启动模板

以下 `create-launch-template` 示例创建一个具有多个标签和块设备映射的启动模板，以在实例启动时指定一个附加 EBS 卷。为 `Groups` 指定一个值，该值与 VPC 的安全组对应，自动扩缩组会将实例启动到该安全组中。指定 VPC 和子网作为自动扩缩组的属性。

```
aws ec2 create-launch-template \  
  --launch-template-name TemplateForAutoScaling \  
  --version-description AutoScalingVersion1 \  
  --launch-template-data '{"NetworkInterfaces":  
 [{"DeviceIndex":0,"AssociatePublicIpAddress":true,"Groups":  
 ["sg-7c227019,sg-903004f8"],"DeleteOnTermination":true}], "ImageId":"ami-  
b42209de","InstanceType":"m4.large","TagSpecifications":  
 [{"ResourceType":"instance","Tags":[{"Key":"environment","Value":"production"},  
 {"Key":"purpose","Value":"webserver"}]},{ "ResourceType":"volume","Tags":  
 [{"Key":"environment","Value":"production"}, {"Key":"cost-  
center","Value":"cc123"}]}}' --region us-east-1
```

输出：

```
{  
  "LaunchTemplate": {  
    "LatestVersionNumber": 1,  
    "LaunchTemplateId": "lt-0123c79c33a54e0abc",  
    "LaunchTemplateName": "TemplateForAutoScaling",  
    "DefaultVersionNumber": 1,  
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",  
    "CreateTime": "2019-04-30T18:16:06.000Z"  
  }  
}
```

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的“为自动扩缩组创建启动模板”。有关引用 JSON 格式参数的信息，请参阅《AWS 命令行界面用户指南》中的“引用字符串”。

## 示例 3：创建指定对 EBS 卷进行加密的启动模板

以下 `create-launch-template` 示例创建一个启动模板，其中包括根据未加密的快照创建的加密 EBS 卷。同时还会在创建过程中标记卷。如果默认情况下禁用了加密，则必须指定以下示例中所示的 `"Encrypted"` 选项。如果使用 `"KmsKeyId"` 选项来指定客户托管的 CMK，则即使默认情况下启用了加密，也必须指定 `"Encrypted"` 选项。



```
aws ec2 create-launch-template \  
  --launch-template-name TemplateForEncryption \  
  --launch-template-data file://config.json
```

config.json 的内容：

```
{  
  "BlockDeviceMappings":[  
    {  
      "DeviceName":"/dev/sda1",  
      "Ebs":{"  
        "VolumeType":"gp2",  
        "DeleteOnTermination":true,  
        "SnapshotId":"snap-066877671789bd71b",  
        "Encrypted":true,  
        "KmsKeyId":"arn:aws:kms:us-east-1:012345678910:key/abcd1234-  
a123-456a-a12b-a123b4cd56ef"  
      }  
    }  
  ],  
  "ImageId":"ami-00068cd7555f543d5",  
  "InstanceType":"c5.large",  
  "TagSpecifications":[  
    {  
      "ResourceType":"volume",  
      "Tags":[  
        {  
          "Key":"encrypted",  
          "Value":"yes"  
        }  
      ]  
    }  
  ]  
}
```

输出：

```
{  
  "LaunchTemplate": {  
    "LatestVersionNumber": 1,  
    "LaunchTemplateId": "lt-0d5bd51bcf8530abc",  
    "LaunchTemplateName": "TemplateForEncryption",
```

```

    "DefaultVersionNumber": 1,
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
    "CreateTime": "2020-01-07T19:08:36.000Z"
  }
}

```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的“从快照恢复 Amazon EBS 卷且默认情况下加密”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateLaunchTemplate](#) 中的。

## create-local-gateway-route-table-virtual-interface-group-association

以下代码示例演示了如何使用 `create-local-gateway-route-table-virtual-interface-group-association`。

### AWS CLI

将本地网关路由表与虚拟接口 (VIF) 组关联

以下 `create-local-gateway-route-table-virtual-interface-group-association` 示例在指定的本地网关路由表和 VIF 组之间创建关联。

```

aws ec2 create-local-gateway-route-table-virtual-interface-group-association \
  --local-gateway-route-table-id lgw-rtb-exampleidabcd1234 \
  --local-gateway-virtual-interface-group-id lgw-vif-grp-exampleid0123abcd

```

输出：

```

{
  "LocalGatewayRouteTableVirtualInterfaceGroupAssociation": {
    "LocalGatewayRouteTableVirtualInterfaceGroupAssociationId": "lgw-vif-grp-
assoc-exampleid12345678",
    "LocalGatewayVirtualInterfaceGroupId": "lgw-vif-grp-exampleid0123abcd",
    "LocalGatewayId": "lgw-exampleid11223344",
    "LocalGatewayRouteTableId": "lgw-rtb-exampleidabcd1234",
    "LocalGatewayRouteTableArn": "arn:aws:ec2:us-west-2:111122223333:local-
gateway-route-table/lgw-rtb-exampleidabcd1234",
    "OwnerId": "111122223333",
    "State": "pending",
    "Tags": []
  }
}

```

```
}
```

有关更多信息，请参阅 [O utpo AWS st s 用户指南中的 VIF 群组关联](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `CreateLocalGatewayRouteTableVirtualInterfaceGroupAssociation`](#) 中的。

## **create-local-gateway-route-table-vpc-association**

以下代码示例演示了如何使用 `create-local-gateway-route-table-vpc-association`。

### AWS CLI

将 VPC 与路由表关联

以下 `create-local-gateway-route-table-vpc-association` 示例将指定的 VPC 与指定的本地网关路由表相关联。

```
aws ec2 create-local-gateway-route-table-vpc-association \
  --local-gateway-route-table-id lgw-rtb-059615ef7dEXAMPLE \
  --vpc-id vpc-07ef66ac71EXAMPLE
```

输出：

```
{
  "LocalGatewayRouteTableVpcAssociation": {
    "LocalGatewayRouteTableVpcAssociationId": "lgw-vpc-assoc-0ee765bcc8EXAMPLE",
    "LocalGatewayRouteTableId": "lgw-rtb-059615ef7dEXAMPLE",
    "LocalGatewayId": "lgw-09b493aa7cEXAMPLE",
    "VpcId": "vpc-07ef66ac71EXAMPLE",
    "State": "associated"
  }
}
```

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `CreateLocalGatewayRouteTableVpcAssociation`](#) 中的。

## **create-local-gateway-route-table**

以下代码示例演示了如何使用 `create-local-gateway-route-table`。

## AWS CLI

### 创建本地网关路由表

以下`create-local-gateway-route-table`示例使用直接 VPC 路由模式创建本地网关路由表。

```
aws ec2 create-local-gateway-route-table \  
  --local-gateway-id lgw-1a2b3c4d5e6f7g8h9 \  
  --mode direct-vpc-routing
```

输出：

```
{  
  "LocalGatewayRouteTable": {  
    "LocalGatewayRouteTableId": "lgw-rtb-abcdefg1234567890",  
    "LocalGatewayRouteTableArn": "arn:aws:ec2:us-west-2:111122223333:local-gateway-route-table/lgw-rtb-abcdefg1234567890",  
    "LocalGatewayId": "lgw-1a2b3c4d5e6f7g8h9",  
    "OutpostArn": "arn:aws:outposts:us-west-2:111122223333:outpost/op-021345abcdef67890",  
    "OwnerId": "111122223333",  
    "State": "pending",  
    "Tags": [],  
    "Mode": "direct-vpc-routing"  
  }  
}
```

有关更多信息，请参阅《AWS Outposts 用户指南》中的[本地网关路由表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLocalGatewayRouteTable](#)中的。

### **create-local-gateway-route**

以下代码示例演示了如何使用 `create-local-gateway-route`。

## AWS CLI

为本地网关路由表创建静态路由

以下`create-local-gateway-route`示例在指定的本地网关路由表中创建指定的路由。

```
aws ec2 create-local-gateway-route \  
  --destination-cidr-block 0.0.0.0/0 \  
  --local-gateway-route-table-id lgw-rtb-059615ef7dEXAMPLE
```

输出：

```
{  
  "Route": {  
    "DestinationCidrBlock": "0.0.0.0/0",  
    "LocalGatewayVirtualInterfaceGroupId": "lgw-vif-grp-07145b276bEXAMPLE",  
    "Type": "static",  
    "State": "deleted",  
    "LocalGatewayRouteTableId": "lgw-rtb-059615ef7dEXAMPLE"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLocalGatewayRoute](#)中的。

## create-managed-prefix-list

以下代码示例演示了如何使用 create-managed-prefix-list。

### AWS CLI

#### 创建前缀列表

以下create-managed-prefix-list示例创建了一个最多包含 10 个条目的 IPv4 前缀列表，并在前缀列表中创建 2 个条目。

```
aws ec2 create-managed-prefix-list \  
  --address-family IPv4 \  
  --max-entries 10 \  
  --entries Cidr=10.0.0.0/16,Description=vpc-a Cidr=10.2.0.0/16,Description=vpc-b \  
  --prefix-list-name vpc-cidrs
```

输出：

```
{  
  "PrefixList": {
```

```

    "PrefixListId": "pl-0123456abcabcabc1",
    "AddressFamily": "IPv4",
    "State": "create-in-progress",
    "PrefixListArn": "arn:aws:ec2:us-west-2:123456789012:prefix-list/
pl-0123456abcabcabc1",
    "PrefixListName": "vpc-cidrs",
    "MaxEntries": 10,
    "Version": 1,
    "Tags": [],
    "OwnerId": "123456789012"
  }
}

```

有关更多信息，请参阅 Amazon VPC 用户指南中的[托管前缀列表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateManagedPrefixList](#)中的。

## create-nat-gateway

以下代码示例演示了如何使用 create-nat-gateway。

### AWS CLI

#### 示例 1：创建公有 NAT 网关

以下 create-nat-gateway 示例在指定子网中创建公有 NAT 网关，并将弹性 IP 地址与指定的分配 ID 关联起来。创建公有 NAT 网关时，必须关联弹性 IP 地址。

```

aws ec2 create-nat-gateway \
  --subnet-id subnet-0250c25a1fEXAMPLE \
  --allocation-id eipalloc-09ad461b0dEXAMPLE

```

输出：

```

{
  "NatGateway": {
    "CreateTime": "2021-12-01T22:22:38.000Z",
    "NatGatewayAddresses": [
      {
        "AllocationId": "eipalloc-09ad461b0dEXAMPLE"
      }
    ],
  },
}

```

```

    "NatGatewayId": "nat-0c61bf8a12EXAMPLE",
    "State": "pending",
    "SubnetId": "subnet-0250c25a1fEXAMPLE",
    "VpcId": "vpc-0a60eb65b4EXAMPLE",
    "ConnectivityType": "public"
  }
}

```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 [NAT 网关](#)。

## 示例 2：创建私有 NAT 网关

以下 `create-nat-gateway` 示例在指定子网中创建私有 NAT 网关。私有 NAT 网关没有关联的弹性 IP 地址。

```

aws ec2 create-nat-gateway \
  --subnet-id subnet-0250c25a1fEXAMPLE \
  --connectivity-type private

```

输出：

```

{
  "NatGateway": {
    "CreateTime": "2021-12-01T22:26:00.000Z",
    "NatGatewayAddresses": [
      {}
    ],
    "NatGatewayId": "nat-011b568379EXAMPLE",
    "State": "pending",
    "SubnetId": "subnet-0250c25a1fEXAMPLE",
    "VpcId": "vpc-0a60eb65b4EXAMPLE",
    "ConnectivityType": "private"
  }
}

```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 [NAT 网关](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateNatGateway](#) 中的。

## create-network-acl-entry

以下代码示例演示了如何使用 `create-network-acl-entry`。

## AWS CLI

### 创建网络 ACL 条目

此示例为指定的网络 ACL 创建了一个条目。该规则允许从 UDP 端口 53 (DNS) 上的任何 IPv4 地址 (0.0.0.0/0) 进入任何关联子网的入口流量。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 create-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-number 100 --protocol udp --port-range From=53,To=53 --cidr-block 0.0.0.0/0 --rule-action allow
```

此示例为指定的网络 ACL 创建了一条规则，该规则允许通过 TCP 端口 80 (HTTP) 来自任何 IPv6 地址 (:: /0) 的入口流量。

命令:

```
aws ec2 create-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-number 120 --protocol tcp --port-range From=80,To=80 --ipv6-cidr-block ::/0 --rule-action allow
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateNetworkAclEntry](#)中的。

## create-network-acl

以下代码示例演示了如何使用 create-network-acl。

### AWS CLI

#### 创建网络 ACL

此示例为指定的 VPC 创建网络 ACL。

命令:

```
aws ec2 create-network-acl --vpc-id vpc-a01106c2
```

输出 :



```
{
  "NetworkAcl": {
    "Associations": [],
    "NetworkAclId": "acl-5fb85d36",
    "VpcId": "vpc-a01106c2",
    "Tags": [],
    "Entries": [
      {
        "CidrBlock": "0.0.0.0/0",
        "RuleNumber": 32767,
        "Protocol": "-1",
        "Egress": true,
        "RuleAction": "deny"
      },
      {
        "CidrBlock": "0.0.0.0/0",
        "RuleNumber": 32767,
        "Protocol": "-1",
        "Egress": false,
        "RuleAction": "deny"
      }
    ],
    "IsDefault": false
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateNetworkAcl](#)中的。

## create-network-insights-access-scope

以下代码示例演示了如何使用 create-network-insights-access-scope。

### AWS CLI

#### 创建网络访问范围

以下create-network-insights-access-scope示例创建了网络访问范围。

```
aws ec2 create-network-insights-access-scope \
  --cli-input-json file://access-scope-file.json
```

access-scope-file.json 的内容：

```
{
  "MatchPaths": [
    {
      "Source": {
        "ResourceStatement": {
          "Resources": [
            "vpc-abcd12e3"
          ]
        }
      }
    }
  ],
  "ExcludePaths": [
    {
      "Source": {
        "ResourceStatement": {
          "ResourceTypes": [
            "AWS::EC2::InternetGateway"
          ]
        }
      }
    }
  ]
}
```

输出：

```
{
  "NetworkInsightsAccessScope": {
    "NetworkInsightsAccessScopeId": "nis-123456789abc01234",
    "NetworkInsightsAccessScopeArn": "arn:aws:ec2:us-east-1:123456789012:network-insights-access-scope/nis-123456789abc01234",
    "CreateDate": "2022-01-25T19:20:28.796000+00:00",
    "UpdateDate": "2022-01-25T19:20:28.797000+00:00"
  },
  "NetworkInsightsAccessScopeContent": {
    "NetworkInsightsAccessScopeId": "nis-123456789abc01234",
    "MatchPaths": [
      {
        "Source": {
          "ResourceStatement": {
            "Resources": [
              "vpc-abcd12e3"
            ]
          }
        }
      }
    ]
  }
}
```

```
        ]
      }
    }
  ],
  "ExcludePaths": [
    {
      "Source": {
        "ResourceStatement": {
          "ResourceTypes": [
            "AWS::EC2::InternetGateway"
          ]
        }
      }
    }
  ]
}
```

有关更多信息，请参阅《[网络访问分析器指南](#)》中的“[使用 AWS CLI 开始使用网络访问分析器](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateNetworkInsightsAccessScope](#) 中的。

## create-network-insights-path

以下代码示例演示了如何使用 `create-network-insights-path`。

### AWS CLI

#### 创建路径

以下 `create-network-insights-path` 示例创建了一个路径。源是指定的互联网网关，目的地是指定的 EC2 实例。要确定是否可以使用指定的协议和端口到达目的地，请使用 `start-network-insights-analysis` 命令分析路径。

```
aws ec2 create-network-insights-path \
  --source igw-0797cccdc9d73b0e5 \
  --destination i-0495d385ad28331c7 \
  --destination-port 22 \
  --protocol TCP
```

输出：

```
{
  "NetworkInsightsPaths": {
    "NetworkInsightsPathId": "nip-0b26f224f1d131fa8",
    "NetworkInsightsPathArn": "arn:aws:ec2:us-east-1:123456789012:network-
insights-path/nip-0b26f224f1d131fa8",
    "CreateDate": "2021-01-20T22:43:46.933Z",
    "Source": "igw-0797cccdc9d73b0e5",
    "Destination": "i-0495d385ad28331c7",
    "Protocol": "tcp"
  }
}
```

有关更多信息，请参阅 [《Reachability Analyzer 指南》](#) 中的 [AWS CLI 使用入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateNetworkInsightsPath](#) 中的。

## create-network-interface-permission

以下代码示例演示了如何使用 `create-network-interface-permission`。

### AWS CLI

#### 创建网络接口权限

此示例向账户授予将网络接口附加到实例的权限。

命令:

```
aws ec2 create-network-interface-permission --network-interface-id eni-1a2b3c4d --
aws-account-id 123456789012 --permission INSTANCE-ATTACH
```

输出:

```
{
  "InterfacePermission": {
    "PermissionState": {
      "State": "GRANTED"
    },
    "NetworkInterfacePermissionId": "eni-perm-06fd19020ede149ea",
    "NetworkInterfaceId": "eni-1a2b3c4d",
    "Permission": "INSTANCE-ATTACH",
    "AwsAccountId": "123456789012"
  }
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateNetworkInterfacePermission](#)中的。

## create-network-interface

以下代码示例演示了如何使用 create-network-interface。

### AWS CLI

示例 1：为网络接口指定 IPv4 地址

以下create-network-interface示例使用指定的主 IPv4 地址为指定子网创建网络接口。

```
aws ec2 create-network-interface \  
  --subnet-id subnet-00a24d0d67acf6333 \  
  --description "my network interface" \  
  --groups sg-09dfba7ed20cda78b \  
  --private-ip-address 10.0.8.17
```

输出：

```
{  
  "NetworkInterface": {  
    "AvailabilityZone": "us-west-2a",  
    "Description": "my network interface",  
    "Groups": [  
      {  
        "GroupName": "my-security-group",  
        "GroupId": "sg-09dfba7ed20cda78b"  
      }  
    ],  
    "InterfaceType": "interface",  
    "Ipv6Addresses": [],  
    "MacAddress": "06:6a:0f:9a:49:37",  
    "NetworkInterfaceId": "eni-0492b355f0cf3b3f8",  
    "OwnerId": "123456789012",  
    "PrivateDnsName": "ip-10-0-8-18.us-west-2.compute.internal",  
    "PrivateIpAddress": "10.0.8.17",  
    "PrivateIpAddresses": [  
      {  
        "Primary": true,
```

```

        "PrivateDnsName": "ip-10-0-8-17.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.17"
    }
],
"RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
"RequesterManaged": false,
"SourceDestCheck": true,
"Status": "pending",
"SubnetId": "subnet-00a24d0d67acf6333",
"TagSet": [],
"VpcId": "vpc-02723a0feeeb9d57b"
}
}

```

## 示例 2：创建具有 IPv4 地址和 IPv6 地址的网络接口

以下 `create-network-interface` 示例使用由 Amazon EC2 选择的 IPv4 地址和 IPv6 地址为指定子网创建网络接口。

```

aws ec2 create-network-interface \
  --subnet-id subnet-00a24d0d67acf6333 \
  --description "my dual stack network interface" \
  --ipv6-address-count 1 \
  --groups sg-09dfba7ed20cda78b

```

输出：

```

{
  "NetworkInterface": {
    "AvailabilityZone": "us-west-2a",
    "Description": "my dual stack network interface",
    "Groups": [
      {
        "GroupName": "my-security-group",
        "GroupId": "sg-09dfba7ed20cda78b"
      }
    ],
    "InterfaceType": "interface",
    "Ipv6Addresses": [
      {
        "Ipv6Address": "2600:1f13:cfe:3650:a1dc:237c:393a:4ba7",
        "IsPrimaryIpv6": false
      }
    ]
  }
}

```

```

    ],
    "MacAddress": "06:b8:68:d2:b2:2d",
    "NetworkInterfaceId": "eni-05da417453f9a84bf",
    "OwnerId": "123456789012",
    "PrivateDnsName": "ip-10-0-8-18.us-west-2.compute.internal",
    "PrivateIpAddress": "10.0.8.18",
    "PrivateIpAddresses": [
      {
        "Primary": true,
        "PrivateDnsName": "ip-10-0-8-18.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.18"
      }
    ],
    "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
    "RequesterManaged": false,
    "SourceDestCheck": true,
    "Status": "pending",
    "SubnetId": "subnet-00a24d0d67acf6333",
    "TagSet": [],
    "VpcId": "vpc-02723a0feeb9d57b",
    "Ipv6Address": "2600:1f13:cfe:3650:a1dc:237c:393a:4ba7"
  }
}

```

### 示例 3：使用连接跟踪配置选项创建网络接口

以下create-network-interface示例创建了一个网络接口并配置了空闲连接跟踪超时。

```

aws ec2 create-network-interface \
  --subnet-id subnet-00a24d0d67acf6333 \
  --groups sg-02e57dbcfe0331c1b \
  --connection-tracking-specification TcpEstablishedTimeout=86400,UdpTimeout=60

```

输出：

```

{
  "NetworkInterface": {
    "AvailabilityZone": "us-west-2a",
    "ConnectionTrackingConfiguration": {
      "TcpEstablishedTimeout": 86400,
      "UdpTimeout": 60
    },
    "Description": ""
  }
}

```

```

    "Groups": [
      {
        "GroupName": "my-security-group",
        "GroupId": "sg-02e57dbcfe0331c1b"
      }
    ],
    "InterfaceType": "interface",
    "Ipv6Addresses": [],
    "MacAddress": "06:4c:53:de:6d:91",
    "NetworkInterfaceId": "eni-0c133586e08903d0b",
    "OwnerId": "123456789012",
    "PrivateDnsName": "ip-10-0-8-94.us-west-2.compute.internal",
    "PrivateIpAddress": "10.0.8.94",
    "PrivateIpAddresses": [
      {
        "Primary": true,
        "PrivateDnsName": "ip-10-0-8-94.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.94"
      }
    ],
    "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
    "RequesterManaged": false,
    "SourceDestCheck": true,
    "Status": "pending",
    "SubnetId": "subnet-00a24d0d67acf6333",
    "TagSet": [],
    "VpcId": "vpc-02723a0feeeb9d57b"
  }
}

```

#### 示例 4：创建弹性结构适配器

以下 `create-network-interface` 示例创建了 EFA。

```

aws ec2 create-network-interface \
  --interface-type efa \
  --subnet-id subnet-00a24d0d67acf6333 \
  --description "my efa" \
  --groups sg-02e57dbcfe0331c1b

```

输出：

```
{
```



```
"NetworkInterface": {
  "AvailabilityZone": "us-west-2a",
  "Description": "my efa",
  "Groups": [
    {
      "GroupName": "my-efa-sg",
      "GroupId": "sg-02e57dbcfe0331c1b"
    }
  ],
  "InterfaceType": "efa",
  "Ipv6Addresses": [],
  "MacAddress": "06:d7:a4:f7:4d:57",
  "NetworkInterfaceId": "eni-034acc2885e862b65",
  "OwnerId": "123456789012",
  "PrivateDnsName": "ip-10-0-8-180.us-west-2.compute.internal",
  "PrivateIpAddress": "10.0.8.180",
  "PrivateIpAddresses": [
    {
      "Primary": true,
      "PrivateDnsName": "ip-10-0-8-180.us-west-2.compute.internal",
      "PrivateIpAddress": "10.0.8.180"
    }
  ],
  "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
  "RequesterManaged": false,
  "SourceDestCheck": true,
  "Status": "pending",
  "SubnetId": "subnet-00a24d0d67acf6333",
  "TagSet": [],
  "VpcId": "vpc-02723a0feeeb9d57b"
}
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[弹性网络接口](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateNetworkInterface](#) 中的。

## create-placement-group

以下代码示例演示了如何使用 create-placement-group。

### AWS CLI

#### 创建置放群组

此示例命令使用指定名称创建置放群组。

命令:

```
aws ec2 create-placement-group --group-name my-cluster --strategy cluster
```

创建分区置放群组

此示例命令创建一个名HDFS-Group-A为五个分区的分区置放组。

命令:

```
aws ec2 create-placement-group --group-name HDFS-Group-A --strategy partition --partition-count 5
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePlacementGroup](#)中的。

## create-replace-root-volume-task

以下代码示例演示了如何使用 create-replace-root-volume-task。

AWS CLI

示例 1：将根卷恢复到其初始启动状态

以下create-replace-root-volume-task示例将实例 i-0123456789abcdefa 的根卷恢复到其初始启动状态。

```
aws ec2 create-replace-root-volume-task \  
--instance-id i-0123456789abcdefa
```

输出：

```
{  
  "ReplaceRootVolumeTask":  
  {  
    "InstanceId": "i-0123456789abcdefa",  
    "ReplaceRootVolumeTaskId": "replacevol-0111122223333abcd",  
    "TaskState": "pending",  
    "StartTime": "2022-03-14T15:06:38Z",  
    "Tags": []  
  }  
}
```

```
}  
}
```

有关更多信息，请参阅 Amazon 弹性计算云用户指南中的[替换根卷](#)。

示例 2：将根卷恢复到特定快照

以下 `create-replace-root-volume-task` 示例将实例 `i-0123456789abcdefa` 的根卷恢复到快照 `snap-0abcdef1234567890`。

```
aws ec2 create-replace-root-volume-task \  
  --instance-id i-0123456789abcdefa \  
  --snapshot-id snap-0abcdef1234567890
```

输出：

```
{  
  "ReplaceRootVolumeTask":  
  {  
    "InstanceId": "i-0123456789abcdefa",  
    "ReplaceRootVolumeTaskId": "replacevol-0555566667777abcd",  
    "TaskState": "pending",  
    "StartTime": "2022-03-14T15:16:28Z",  
    "Tags": []  
  }  
}
```

有关更多信息，请参阅 Amazon 弹性计算云用户指南中的[替换根卷](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateReplaceRootVolumeTask](#) 中的。

## create-reserved-instances-listing

以下代码示例演示了如何使用 `create-reserved-instances-listing`。

### AWS CLI

在预留实例 Marketplace 上架预留实例

以下 `create-reserved-instances-listing` 示例在预留实例 Marketplace 中创建了指定预留实例的清单。

```
aws ec2 create-reserved-instances-listing \  
  --reserved-instances-id 5ec28771-05ff-4b9b-aa31-9e57dexample \  
  --instance-count 3 \  
  --price-schedules CurrencyCode=USD,Price=25.50 \  
  --client-token 550e8400-e29b-41d4-a716-446655440000
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateReservedInstancesListing](#)中的。

## create-restore-image-task

以下代码示例演示了如何使用 create-restore-image-task。

### AWS CLI

#### 从 S3 存储桶恢复 AMI

以下create-restore-image-task示例从 S3 存储桶恢复 AMI。使用describe-store-image-tasks输出中的值，指定 AMI 的对象密钥和将 AMI 复制到的 S3 存储桶的名称，然后为已恢复的 AMI 指定名称。S3objectKey 和 Bucket名称对该账户在该区域中的 AMI 必须唯一。恢复的 AMI 将收到一个新的 AMI ID。

```
aws ec2 create-restore-image-task \  
  --object-key ami-1234567890abcdef0.bin \  
  --bucket my-ami-bucket \  
  --name "New AMI Name"
```

输出：

```
{  
  "ImageId": "ami-0eab20fe36f83e1a8"  
}
```

有关使用 S3 存储和恢复 AMI 的更多信息，请参阅亚马逊 EC2 用户指南中的使用 S3 存储和恢复 AMI < <https://docs.aws.amazon.com/AWS ec2/latest/UserGuide/ami-store-restore.html>>。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRestoreImageTask](#)中的。

## create-route-table

以下代码示例演示了如何使用 create-route-table。

## AWS CLI

### 创建路由表

本示例为指定的 VPC 创建路由表。

命令:

```
aws ec2 create-route-table --vpc-id vpc-a01106c2
```

输出:

```
{
  "RouteTable": {
    "Associations": [],
    "RouteTableId": "rtb-22574640",
    "VpcId": "vpc-a01106c2",
    "PropagatingVgws": [],
    "Tags": [],
    "Routes": [
      {
        "GatewayId": "local",
        "DestinationCidrBlock": "10.0.0.0/16",
        "State": "active"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRouteTable](#)中的。

### create-route

以下代码示例演示了如何使用 create-route。

## AWS CLI

### 创建路线

此示例为指定的路由表创建路由。该路由匹配所有 IPv4 流量 (0.0.0.0/0)，并将其路由到指定的互联网网关。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 create-route --route-table-id rtb-22574640 --destination-cidr-block 0.0.0.0/0 --gateway-id igw-c0a643a9
```

此示例命令在路由表 `rtb-g8ff4ea2` 中创建路由。该路由匹配 IPv4 CIDR 块 `10.0.0.0/16` 的流量，并将其路由到 VPC 对等连接 `pcx-111aaa22`。通过此路由，可以将流量定向到 VPC 对等连接中的对等 VPC。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 create-route --route-table-id rtb-g8ff4ea2 --destination-cidr-block 10.0.0.0/16 --vpc-peering-connection-id pcx-1a2b3c4d
```

此示例在指定的路由表中创建一个匹配所有 IPv6 流量 (`::/0`) 的路由，并将其路由到指定的仅限出口 Internet 网关。

命令:

```
aws ec2 create-route --route-table-id rtb-dce620b8 --destination-ipv6-cidr-block ::/0 --egress-only-internet-gateway-id eigw-01eadbd45ecd7943f
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateRoute](#) 中的。

## create-security-group

以下代码示例演示了如何使用 `create-security-group`。

### AWS CLI

为 EC2-Classical 创建安全组

本示例将创建一个名为 `MySecurityGroup` 的安全组。

命令:

```
aws ec2 create-security-group --group-name MySecurityGroup --description "My security group"
```

输出:

```
{
  "GroupId": "sg-903004f8"
}
```

为 EC2-VPC 创建安全组

本示例为指定的 VPC 创建名为 MySecurityGroup 的安全组。

命令:

```
aws ec2 create-security-group --group-name MySecurityGroup --description "My
security group" --vpc-id vpc-1a2b3c4d
```

输出:

```
{
  "GroupId": "sg-903004f8"
}
```

有关更多信息，请参阅《AWS 命令行界面用户指南》中的“使用安全组”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSecurityGroup](#)中的。

## create-snapshot

以下代码示例演示了如何使用 create-snapshot。

AWS CLI

创建快照

此示例命令创建卷的快照，卷 ID 为，vol-1234567890abcdef0并附上用于识别快照的简短描述。

命令:

```
aws ec2 create-snapshot --volume-id vol-1234567890abcdef0 --description "This is my
root volume snapshot"
```

输出:

```
{
  "Description": "This is my root volume snapshot",
  "Tags": [],
  "Encrypted": false,
  "VolumeId": "vol-1234567890abcdef0",
  "State": "pending",
  "VolumeSize": 8,
  "StartTime": "2018-02-28T21:06:01.000Z",
  "Progress": "",
  "OwnerId": "012345678910",
  "SnapshotId": "snap-066877671789bd71b"
}
```

### 创建带有标签的快照

此示例命令创建快照并应用两个标签：purpose=prod 和 costcenter=123。

命令：

```
aws ec2 create-snapshot --volume-id vol-1234567890abcdef0 --description 'Prod
backup' --tag-specifications 'ResourceType=snapshot,Tags=[{Key=purpose,Value=prod},
{Key=costcenter,Value=123}]'
```

输出：

```
{
  "Description": "Prod backup",
  "Tags": [
    {
      "Value": "prod",
      "Key": "purpose"
    },
    {
      "Value": "123",
      "Key": "costcenter"
    }
  ],
  "Encrypted": false,
  "VolumeId": "vol-1234567890abcdef0",
  "State": "pending",
  "VolumeSize": 8,
  "StartTime": "2018-02-28T21:06:06.000Z",
```



```
"Progress": "",
"OwnerId": "012345678910",
"SnapshotId": "snap-09ed24a70bc19bbe4"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSnapshot](#)中的。

## create-snapshots

以下代码示例演示了如何使用 create-snapshots。

### AWS CLI

#### 示例 1：创建多卷快照

以下create-snapshots示例创建连接到指定实例的所有卷的快照。

```
aws ec2 create-snapshots \
  --instance-specification InstanceId=i-1234567890abcdef0 \
  --description "This is snapshot of a volume from my-instance"
```

输出：

```
{
  "Snapshots": [
    {
      "Description": "This is a snapshot of a volume from my-instance",
      "Tags": [],
      "Encrypted": false,
      "VolumeId": "vol-0a01d2d5a34697479",
      "State": "pending",
      "VolumeSize": 16,
      "StartTime": "2019-08-05T16:58:19.000Z",
      "Progress": "",
      "OwnerId": "123456789012",
      "SnapshotId": "snap-07f30e3909aa0045e"
    },
    {
      "Description": "This is a snapshot of a volume from my-instance",
      "Tags": [],
      "Encrypted": false,
      "VolumeId": "vol-02d0d4947008cb1a2",
```

```

        "State": "pending",
        "VolumeSize": 20,
        "StartTime": "2019-08-05T16:58:19.000Z",
        "Progress": "",
        "OwnerId": "123456789012",
        "SnapshotId": "snap-0ec20b602264aad48"
    },
    ...
]
}

```

## 示例 2：使用源卷中的标签创建多卷快照

以下 `create-snapshots` 示例创建连接到指定实例的所有卷的快照，并将标签从每个卷复制到其对应的快照中。

```

aws ec2 create-snapshots \
  --instance-specification InstanceId=i-1234567890abcdef0 \
  --copy-tags-from-source volume \
  --description "This is snapshot of a volume from my-instance"

```

输出：

```

{
  "Snapshots": [
    {
      "Description": "This is a snapshot of a volume from my-instance",
      "Tags": [
        {
          "Key": "Name",
          "Value": "my-volume"
        }
      ],
      "Encrypted": false,
      "VolumeId": "vol-02d0d4947008cb1a2",
      "State": "pending",
      "VolumeSize": 20,
      "StartTime": "2019-08-05T16:53:04.000Z",
      "Progress": "",
      "OwnerId": "123456789012",
      "SnapshotId": "snap-053bfaeb821a458dd"
    }
  ]
  ...
}

```

```
]
}
```

### 示例 3：创建不包括根卷的多卷快照

以下create-snapshots示例创建了除根卷之外连接到指定实例的所有卷的快照。

```
aws ec2 create-snapshots \
  --instance-specification InstanceId=i-1234567890abcdef0,ExcludeBootVolume=true
```

有关输出示例，请参阅示例 1。

### 示例 4：创建多卷快照并添加标签

以下create-snapshots示例创建附加到指定实例的所有卷的快照，并为每个快照添加两个标签。

```
aws ec2 create-snapshots \
  --instance-specification InstanceId=i-1234567890abcdef0 \
  --tag-specifications 'ResourceType=snapshot,Tags=[{Key=Name,Value=backup},
  {Key=costcenter,Value=123}]'
```

有关输出示例，请参阅示例 1。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSnapshots](#)中的。

## create-spot-datafeed-subscription

以下代码示例演示了如何使用 create-spot-datafeed-subscription。

### AWS CLI

#### 创建竞价型实例数据源

以下create-spot-datafeed-subscription示例创建了竞价型实例数据馈送。

```
aws ec2 create-spot-datafeed-subscription \
  --bucket my-bucket \
  --prefix spot-data-feed
```

输出：

```
{
  "SpotDatafeedSubscription": {
    "Bucket": "my-bucket",
    "OwnerId": "123456789012",
    "Prefix": "spot-data-feed",
    "State": "Active"
  }
}
```

数据源存储在您指定的 Amazon S3 存储桶中。此数据馈送的文件名采用以下格式。

```
my-bucket.s3.amazonaws.com/spot-data-feed/123456789012.YYYY-MM-DD-HH.n.abcd1234.gz
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的竞价型实例[数据源](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateSpotDatafeedSubscription](#)中的。

## create-store-image-task

以下代码示例演示了如何使用 create-store-image-task。

### AWS CLI

将 AMI 存储在 S3 存储桶中

以下 create-store-image-task 示例将 AMI 存储在 S3 存储桶中。指定 AMI 的 ID 以及要在其中存储 AMI 的 S3 存储桶的名称。

```
aws ec2 create-store-image-task \
  --image-id ami-1234567890abcdef0 \
  --bucket my-ami-bucket
```

输出：

```
{
  "ObjectKey": "ami-1234567890abcdef0.bin"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[使用 S3 存储和恢复 AMI](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateStoreImageTask](#)中的。

## create-subnet-cidr-reservation

以下代码示例演示了如何使用 `create-subnet-cidr-reservation`。

### AWS CLI

#### 创建子网 CIDR 预留

以下`create-subnet-cidr-reservation`示例为指定的子网和 CIDR 范围创建子网 CIDR 预留。

```
aws ec2 create-subnet-cidr-reservation \
  --subnet-id subnet-03c51e2eEXAMPLE \
  --reservation-type prefix \
  --cidr 10.1.0.20/26
```

输出：

```
{
  "SubnetCidrReservation": {
    "SubnetCidrReservationId": "scr-044f977c4eEXAMPLE",
    "SubnetId": "subnet-03c51e2e6cEXAMPLE",
    "Cidr": "10.1.0.16/28",
    "ReservationType": "prefix",
    "OwnerId": "123456789012"
  }
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的[子网 CIDR 预留](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSubnetCidrReservation](#)中的。

## create-subnet

以下代码示例演示了如何使用 `create-subnet`。

### AWS CLI

#### 示例 1：创建仅具有 IPv4 CIDR 块的子网

以下 `create-subnet` 示例在指定的 VPC 中创建具有指定 IPv4 CIDR 块的子网。

```
aws ec2 create-subnet \  
  --vpc-id vpc-081ec835f3EXAMPLE \  
  --cidr-block 10.0.0.0/24 \  
  --tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv4-only-  
subnet}]
```

输出：

```
{  
  "Subnet": {  
    "AvailabilityZone": "us-west-2a",  
    "AvailabilityZoneId": "usw2-az2",  
    "AvailableIpAddressCount": 251,  
    "CidrBlock": "10.0.0.0/24",  
    "DefaultForAz": false,  
    "MapPublicIpOnLaunch": false,  
    "State": "available",  
    "SubnetId": "subnet-0e99b93155EXAMPLE",  
    "VpcId": "vpc-081ec835f3EXAMPLE",  
    "OwnerId": "123456789012",  
    "AssignIpv6AddressOnCreation": false,  
    "Ipv6CidrBlockAssociationSet": [],  
    "Tags": [  
      {  
        "Key": "Name",  
        "Value": "my-ipv4-only-subnet"  
      }  
    ],  
    "SubnetArn": "arn:aws:ec2:us-west-2:123456789012:subnet/  
subnet-0e99b93155EXAMPLE"  
  }  
}
```

示例 2：创建同时具有 IPv4 和 IPv6 CIDR 块的子网

以下 `create-subnet` 示例在指定的 VPC 中，创建同时具有指定 IPv4 和 IPv6 CIDR 块的子网。

```
aws ec2 create-subnet \  
  --vpc-id vpc-081ec835f3EXAMPLE \  
  --cidr-block 10.0.0.0/24 \  
  --ipv6-cidr-block 2600:1f16:cfe:3660::/64 \  
  --tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv4-and-ipv6-only-subnet}]
```

```
--tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv4-ipv6-
subnet}]
```

输出：

```
{
  "Subnet": {
    "AvailabilityZone": "us-west-2a",
    "AvailabilityZoneId": "usw2-az2",
    "AvailableIpAddressCount": 251,
    "CidrBlock": "10.0.0.0/24",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-0736441d38EXAMPLE",
    "VpcId": "vpc-081ec835f3EXAMPLE",
    "OwnerId": "123456789012",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [
      {
        "AssociationId": "subnet-cidr-assoc-06c5f904499fcc623",
        "Ipv6CidrBlock": "2600:1f13:cfe:3660::/64",
        "Ipv6CidrBlockState": {
          "State": "associating"
        }
      }
    ],
    "Tags": [
      {
        "Key": "Name",
        "Value": "my-ipv4-ipv6-subnet"
      }
    ],
    "SubnetArn": "arn:aws:ec2:us-west-2:123456789012:subnet/
subnet-0736441d38EXAMPLE"
  }
}
```

示例 3：创建仅具有 IPv6 CIDR 块的子网

以下 create-subnet 示例在指定的 VPC 中创建具有指定 IPv6 CIDR 块的子网。

```
aws ec2 create-subnet \
```

```
--vpc-id vpc-081ec835f3EXAMPLE \  
--ipv6-native \  
--ipv6-cidr-block 2600:1f16:115:200::/64 \  
--tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv6-only-  
subnet}]
```

输出：

```
{  
  "Subnet": {  
    "AvailabilityZone": "us-west-2a",  
    "AvailabilityZoneId": "usw2-az2",  
    "AvailableIpAddressCount": 0,  
    "DefaultForAz": false,  
    "MapPublicIpOnLaunch": false,  
    "State": "available",  
    "SubnetId": "subnet-03f720e7deEXAMPLE",  
    "VpcId": "vpc-081ec835f3EXAMPLE",  
    "OwnerId": "123456789012",  
    "AssignIpv6AddressOnCreation": true,  
    "Ipv6CidrBlockAssociationSet": [  
      {  
        "AssociationId": "subnet-cidr-assoc-01ef639edde556709",  
        "Ipv6CidrBlock": "2600:1f13:cfe:3660::/64",  
        "Ipv6CidrBlockState": {  
          "State": "associating"  
        }  
      }  
    ],  
    "Tags": [  
      {  
        "Key": "Name",  
        "Value": "my-ipv6-only-subnet"  
      }  
    ],  
    "SubnetArn": "arn:aws:ec2:us-west-2:123456789012:subnet/  
subnet-03f720e7deEXAMPLE"  
  }  
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的 [VPC 和子网](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateSubnet](#) 中的。



## create-tags

以下代码示例演示了如何使用 create-tags。

### AWS CLI

#### 示例 1：为资源添加标签

以下 create-tags 示例将标签 Stack=production 添加到指定的映像，或者覆盖 AMI 的现有标签（其中标签键为 Stack）。

```
aws ec2 create-tags \  
  --resources ami-1234567890abcdef0 \  
  --tags Key=Stack,Value=production
```

有关更多信息，请参阅 [《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的主题标题](#)。

#### 示例 2：为多个资源添加标签

以下 create-tags 示例为 AMI 和实例添加（或覆盖）两个标签。其中一个标签有一个键（webserver），但没有值（值设置为空字符串）。另一个标签有一个键（stack）和一个值（Production）。

```
aws ec2 create-tags \  
  --resources ami-1a2b3c4d i-1234567890abcdef0 \  
  --tags Key=webserver,Value= Key=stack,Value=Production
```

有关更多信息，请参阅 [《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的主题标题](#)。

#### 示例 3：添加包含特殊字符的标签

以下 create-tags 示例为实例添加标签 [Group]=test。方括号（[ 和 ]）是特殊字符，必须对其进行转义。以下示例还使用适用于每个环境的行延续字符。

如果使用的是 Windows，请用双引号（"）将具有特殊字符的元素引起来，然后在每个双引号字符前面添加反斜杠（\），如下所示：

```
aws ec2 create-tags ^  
  --resources i-1234567890abcdef0 ^  
  --tags Key=\"[Group]\",Value=test
```

如果您使用的是 Windows PowerShell，请在元素中使用双引号 (") 将带有特殊字符的值括起来，在每个双引号字符前面加一个反斜杠 (\)，然后用单引号 (') 将整个键和值结构括起来，如下所示：

```
aws ec2 create-tags `
  --resources i-1234567890abcdef0 `
  --tags 'Key="\[Group]\",Value=test'
```

如果使用的是 Linux 或 OS X，请使用双引号 (") 将具有特殊字符的元素引起来，然后使用单引号 (') 将整个键和值结构引起来，如下所示：

```
aws ec2 create-tags \
  --resources i-1234567890abcdef0 \
  --tags 'Key="[Group]",Value=test'
```

有关更多信息，请参阅 [《适用于 Linux 实例的 Amazon 弹性计算云用户指南》](#) 中的主题标题。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateTags](#) 中的。

## create-traffic-mirror-filter-rule

以下代码示例演示了如何使用 create-traffic-mirror-filter-rule。

### AWS CLI

为传入的 TCP 流量创建过滤规则

以下 create-traffic-mirror-filter-rule 示例创建了一条规则，您可以使用该规则镜像所有传入的 TCP 流量。在运行此命令之前，create-traffic-mirror-filter 请使用创建流量镜像过滤器。

```
aws ec2 create-traffic-mirror-filter-rule \
  --description "TCP Rule" \
  --destination-cidr-block 0.0.0.0/0 \
  --protocol 6 \
  --rule-action accept \
  --rule-number 1 \
  --source-cidr-block 0.0.0.0/0 \
  --traffic-direction ingress \
  --traffic-mirror-filter-id tmf-04812ff784b25ae67
```

输出：

```
{
  "TrafficMirrorFilterRule": {
    "DestinationCidrBlock": "0.0.0.0/0",
    "TrafficMirrorFilterId": "tmf-04812ff784b25ae67",
    "TrafficMirrorFilterRuleId": "tmfr-02d20d996673f3732",
    "SourceCidrBlock": "0.0.0.0/0",
    "TrafficDirection": "ingress",
    "Description": "TCP Rule",
    "RuleNumber": 1,
    "RuleAction": "accept",
    "Protocol": 6
  },
  "ClientToken": "4752b573-40a6-4eac-a8a4-a72058761219"
}
```

有关更多信息，请参阅《[流量镜像指南](#)》中的[创建AWS 流量镜像过滤器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTrafficMirrorFilterRule](#)中的。

## create-traffic-mirror-filter

以下代码示例演示了如何使用 create-traffic-mirror-filter。

### AWS CLI

#### 创建流量镜像过滤器

以下create-traffic-mirror-filter示例创建了流量镜像过滤器。创建筛选器后，使用create-traffic-mirror-filter-rule向过滤器添加规则。

```
aws ec2 create-traffic-mirror-filter \
  --description "TCP Filter"
```

输出：

```
{
  "ClientToken": "28908518-100b-4987-8233-8c744EXAMPLE",
  "TrafficMirrorFilter": {
    "TrafficMirrorFilterId": "tmf-04812ff784EXAMPLE",
    "Description": "TCP Filter",
    "EgressFilterRules": [],
    "IngressFilterRules": [],
  }
}
```

```
    "Tags": [],
    "NetworkServices": []
  }
}
```

有关更多信息，请参阅《[流量镜像指南](#)》中的[创建AWS 流量镜像过滤器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTrafficMirrorFilter](#)中的。

## create-traffic-mirror-session

以下代码示例演示了如何使用 create-traffic-mirror-session。

### AWS CLI

#### 创建流量镜像会话

以下create-traffic-mirror-session命令为数据包的 25 字节的指定源和目标创建流量镜像会话。

```
aws ec2 create-traffic-mirror-session \
  --description "example session" \
  --traffic-mirror-target-id tmt-07f75d8feeEXAMPLE \
  --network-interface-id eni-070203f901EXAMPLE \
  --session-number 1 \
  --packet-length 25 \
  --traffic-mirror-filter-id tmf-04812ff784EXAMPLE
```

输出：

```
{
  "TrafficMirrorSession": {
    "TrafficMirrorSessionId": "tms-08a33b1214EXAMPLE",
    "TrafficMirrorTargetId": "tmt-07f75d8feeEXAMPLE",
    "TrafficMirrorFilterId": "tmf-04812ff784EXAMPLE",
    "NetworkInterfaceId": "eni-070203f901EXAMPLE",
    "OwnerId": "111122223333",
    "PacketLength": 25,
    "SessionNumber": 1,
    "VirtualNetworkId": 7159709,
    "Description": "example session",
    "Tags": []
  },
}
```

```
"ClientToken": "5236cffc-ee13-4a32-bb5b-388d9da09d96"
}
```

有关更多信息，请参阅《[流量镜像指南](#)》中的[创建AWS 流量镜像会话](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTrafficMirrorSession](#)中的。

## create-traffic-mirror-target

以下代码示例演示了如何使用 create-traffic-mirror-target。

### AWS CLI

创建 Network Load Balancer 流量镜像目标

以下create-traffic-mirror-target示例创建了 Network Load Balancer 流量镜像目标。

```
aws ec2 create-traffic-mirror-target \
  --description "Example Network Load Balancer Target" \
  --network-load-balancer-arn arn:aws:elasticloadbalancing:us-
east-1:111122223333:loadbalancer/net/NLB/7cdec873EXAMPLE
```

输出：

```
{
  "TrafficMirrorTarget": {
    "Type": "network-load-balancer",
    "Tags": [],
    "Description": "Example Network Load Balancer Target",
    "OwnerId": "111122223333",
    "NetworkLoadBalancerArn": "arn:aws:elasticloadbalancing:us-
east-1:724145273726:loadbalancer/net/NLB/7cdec873EXAMPLE",
    "TrafficMirrorTargetId": "tmt-0dabe9b0a6EXAMPLE"
  },
  "ClientToken": "d5c090f5-8a0f-49c7-8281-72c796a21f72"
}
```

创建网络流量镜像目标

以下create-traffic-mirror-target示例创建了一个网络接口流量镜像目标。

```
aws ec2 create-traffic-mirror-target — 描述“网络接口目标” — eni-eni-01f6f631eExample network-
interface-id
```

输出：

```
{
  "ClientToken": "5289a345-0358-4e62-93d5-47ef3061d65e",
  "TrafficMirrorTarget": {
    "Description": "Network interface target",
    "NetworkInterfaceId": "eni-01f6f631eEXAMPLE",
    "TrafficMirrorTargetId": "tmt-02dcdb2abEXAMPLE",
    "OwnerId": "111122223333",
    "Type": "network-interface",
    "Tags": []
  }
}
```

有关更多信息，请参阅《[流量镜像指南](#)》中的[创建AWS 流量镜像目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTrafficMirrorTarget](#)中的。

## create-transit-gateway-connect-peer

以下代码示例演示了如何使用 `create-transit-gateway-connect-peer`。

### AWS CLI

创建 Transit Gateway Connect 对等

以下`create-transit-gateway-connect-peer`示例创建了一个 Connect 对等体。

```
aws ec2 create-transit-gateway-connect-peer \
  --transit-gateway-attachment-id tgw-attach-0f0927767cEXAMPLE \
  --peer-address 172.31.1.11 \
  --inside-cidr-blocks 169.254.6.0/29
```

输出：

```
{
  "TransitGatewayConnectPeer": {
    "TransitGatewayAttachmentId": "tgw-attach-0f0927767cEXAMPLE",
    "TransitGatewayConnectPeerId": "tgw-connect-peer-0666adbac4EXAMPLE",
    "State": "pending",
    "CreationTime": "2021-10-13T03:35:17.000Z",
    "ConnectPeerConfiguration": {
```

```
    "TransitGatewayAddress": "10.0.0.234",
    "PeerAddress": "172.31.1.11",
    "InsideCidrBlocks": [
      "169.254.6.0/29"
    ],
    "Protocol": "gre",
    "BgpConfigurations": [
      {
        "TransitGatewayAsn": 64512,
        "PeerAsn": 64512,
        "TransitGatewayAddress": "169.254.6.2",
        "PeerAddress": "169.254.6.1",
        "BgpStatus": "down"
      },
      {
        "TransitGatewayAsn": 64512,
        "PeerAsn": 64512,
        "TransitGatewayAddress": "169.254.6.3",
        "PeerAddress": "169.254.6.1",
        "BgpStatus": "down"
      }
    ]
  }
}
```

有关更多信息，请参阅《公交网关指南》中的[公交网关 Connect 附件和 Transit Gateway Connect 对等体](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTransitGatewayConnectPeer](#)中的。

## create-transit-gateway-connect

以下代码示例演示了如何使用 create-transit-gateway-connect。

### AWS CLI

#### 创建公交网关 Connect 附件

以下create-transit-gateway-connect示例使用“gre”协议为指定的附件创建一个 Connect 附件。

```
aws ec2 create-transit-gateway-connect \
```

```
--transport-transit-gateway-attachment-id tgw-attach-0a89069f57EXAMPLE \  
--options "Protocol=gre"
```

输出：

```
{  
  "TransitGatewayConnect": {  
    "TransitGatewayAttachmentId": "tgw-attach-037012e5dcEXAMPLE",  
    "TransportTransitGatewayAttachmentId": "tgw-attach-0a89069f57EXAMPLE",  
    "TransitGatewayId": "tgw-02f776b1a7EXAMPLE",  
    "State": "pending",  
    "CreationTime": "2021-03-09T19:59:17+00:00",  
    "Options": {  
      "Protocol": "gre"  
    }  
  }  
}
```

有关更多信息，请参阅《公交网关指南》中的[公交网关 Connect 附件和 Transit Gateway Connect 对等体](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTransitGatewayConnect](#)中的。

## create-transit-gateway-multicast-domain

以下代码示例演示了如何使用 create-transit-gateway-multicast-domain。

### AWS CLI

#### 示例 1：创建 IGMP 组播域

以下create-transit-gateway-multicast-domain示例为指定的传输网关创建多播域。禁用静态源后，与多播域关联的子网中的任何实例都可以发送多播流量。如果至少有一个成员使用IGMP 协议，则必须启用 IGMPv2 支持。

```
aws ec2 create-transit-gateway-multicast-domain \  
  --transit-gateway-id tgw-0bf0bffefaEXAMPLE \  
  --options StaticSourcesSupport=disable,Igmpv2Support=enable
```

输出：

```
{
```



```

    "TransitGatewayMulticastDomain": {
      "TransitGatewayMulticastDomainId": "tgw-mcast-domain-0c9e29e2a7EXAMPLE",
      "TransitGatewayId": "tgw-0bf0bffefaEXAMPLE",
      "TransitGatewayMulticastDomainArn": "arn:aws:ec2:us-
west-2:123456789012:transit-gateway-multicast-domain/tgw-mcast-
domain-0c9e29e2a7EXAMPLE",
      "OwnerId": "123456789012",
      "Options": {
        "Icmpv2Support": "enable",
        "StaticSourcesSupport": "disable",
        "AutoAcceptSharedAssociations": "disable"
      },
      "State": "pending",
      "CreationTime": "2021-09-29T22:17:13.000Z"
    }
  }
}

```

## 示例 2：创建静态多播域

以下 `create-transit-gateway-multicast-domain` 示例为指定的传输网关创建多播域。启用静态源后，必须静态添加源。

```

aws ec2 create-transit-gateway-multicast-domain \
  --transit-gateway-id tgw-0bf0bffefaEXAMPLE \
  --options StaticSourcesSupport=enable,Icmpv2Support=disable

```

输出：

```

{
  "TransitGatewayMulticastDomain": {
    "TransitGatewayMulticastDomainId": "tgw-mcast-domain-000fb24d04EXAMPLE",
    "TransitGatewayId": "tgw-0bf0bffefaEXAMPLE",
    "TransitGatewayMulticastDomainArn": "arn:aws:ec2:us-
west-2:123456789012:transit-gateway-multicast-domain/tgw-mcast-
domain-000fb24d04EXAMPLE",
    "OwnerId": "123456789012",
    "Options": {
      "Icmpv2Support": "disable",
      "StaticSourcesSupport": "enable",
      "AutoAcceptSharedAssociations": "disable"
    },
    "State": "pending",
    "CreationTime": "2021-09-29T22:20:19.000Z"
  }
}

```

```
}  
}
```

有关更多信息，请参阅《传输网关指南》中的[管理多播域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTransitGatewayMulticastDomain](#)中的。

## create-transit-gateway-peering-attachment

以下代码示例演示了如何使用 create-transit-gateway-peering-attachment。

### AWS CLI

#### 创建公交网关对等连接

以下create-transit-gateway-peering-attachment示例在两个指定的中转网关之间创建对等连接请求。

```
aws ec2 create-transit-gateway-peering-attachment \  
  --transit-gateway-id tgw-123abc05e04123abc \  
  --peer-transit-gateway-id tgw-11223344aabbcc112 \  
  --peer-account-id 123456789012 \  
  --peer-region us-east-2
```

输出：

```
{  
  "TransitGatewayPeeringAttachment": {  
    "TransitGatewayAttachmentId": "tgw-attach-4455667788aabbccd",  
    "RequesterTgwInfo": {  
      "TransitGatewayId": "tgw-123abc05e04123abc",  
      "OwnerId": "123456789012",  
      "Region": "us-west-2"  
    },  
    "AcceptorTgwInfo": {  
      "TransitGatewayId": "tgw-11223344aabbcc112",  
      "OwnerId": "123456789012",  
      "Region": "us-east-2"  
    },  
    "State": "initiatingRequest",  
    "CreationTime": "2019-12-09T11:38:05.000Z"  
  }  
}
```

```
}
```

有关更多信息，请参阅《[公交网关指南](#)》中的 [Transit Gateway 对等连接附件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTransitGatewayPeeringAttachment](#)中的。

## create-transit-gateway-policy-table

以下代码示例演示了如何使用 `create-transit-gateway-policy-table`。

### AWS CLI

创建传输网关策略表

以下`create-transit-gateway-policy-table`示例为指定的中转网关创建传输网关策略表。

```
aws ec2 create-transit-gateway-policy-table \
  --transit-gateway-id tgw-067f8505c18f0bd6e
```

输出：

```
{
  "TransitGatewayPolicyTable": {
    "TransitGatewayPolicyTableId": "tgw-ptb-0a16f134b78668a81",
    "TransitGatewayId": "tgw-067f8505c18f0bd6e",
    "State": "pending",
    "CreationTime": "2023-11-28T16:36:43+00:00"
  }
}
```

有关更多信息，请参阅 [Transit Gateway 用户指南中的公交网关策略表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTransitGatewayPolicyTable](#)中的。

## create-transit-gateway-prefix-list-reference

以下代码示例演示了如何使用 `create-transit-gateway-prefix-list-reference`。

### AWS CLI

创建对前缀列表的引用

以下 `create-transit-gateway-prefix-list-reference` 示例创建了对指定公网网关路由表中指定前缀列表的引用。

```
aws ec2 create-transit-gateway-prefix-list-reference \  
  --transit-gateway-route-table-id tgw-rtb-0123456789abcd123 \  
  --prefix-list-id pl-1111112222222333 \  
  --transit-gateway-attachment-id tgw-attach-aaaaaabbbbb11111
```

输出：

```
{  
  "TransitGatewayPrefixListReference": {  
    "TransitGatewayRouteTableId": "tgw-rtb-0123456789abcd123",  
    "PrefixListId": "pl-1111112222222333",  
    "PrefixListOwnerId": "123456789012",  
    "State": "pending",  
    "Blackhole": false,  
    "TransitGatewayAttachment": {  
      "TransitGatewayAttachmentId": "tgw-attach-aaaaaabbbbb11111",  
      "ResourceType": "vpc",  
      "ResourceId": "vpc-112233445566aabbcc"  
    }  
  }  
}
```

有关更多信息，请参阅 *T transit Gateways 指南* 中的 [前缀列表参考](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateTransitGatewayPrefixListReference](#) 中的。

## create-transit-gateway-route-table

以下代码示例演示了如何使用 `create-transit-gateway-route-table`。

### AWS CLI

#### 创建 Transit Gateway 路由表

以下 `create-transit-gateway-route-table` 示例为指定的中转网关创建路由表。

```
aws ec2 create-transit-gateway-route-table \  
  --transit-gateway-id tgw-0262a0e521EXAMPLE
```

输出：

```
{
  "TransitGatewayRouteTable": {
    "TransitGatewayRouteTableId": "tgw-rtb-0960981be7EXAMPLE",
    "TransitGatewayId": "tgw-0262a0e521EXAMPLE",
    "State": "pending",
    "DefaultAssociationRouteTable": false,
    "DefaultPropagationRouteTable": false,
    "CreationTime": "2019-07-10T19:01:46.000Z"
  }
}
```

有关更多信息，请参阅 [Transit Gateways 指南中的创建中转网关路由表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateTransitGatewayRouteTable](#) 中的。

## create-transit-gateway-route

以下代码示例演示了如何使用 create-transit-gateway-route。

### AWS CLI

#### 创建网关路由

以下 create-transit-gateway-route 示例为指定路由表创建具有指定目的地的路由。

```
aws ec2 create-transit-gateway-route \
  --destination-cidr-block 10.0.2.0/24 \
  --transit-gateway-route-table-id tgw-rtb-0b6f6aaa01EXAMPLE \
  --transit-gateway-attachment-id tgw-attach-0b5968d3b6EXAMPLE
```

输出：

```
{
  "Route": {
    "DestinationCidrBlock": "10.0.2.0/24",
    "TransitGatewayAttachments": [
      {
        "ResourceId": "vpc-0065acced4EXAMPLE",
        "TransitGatewayAttachmentId": "tgw-attach-0b5968d3b6EXAMPLE",
        "ResourceType": "vpc"
      }
    ]
  }
}
```

```
    ],
    "Type": "static",
    "State": "active"
  }
}
```

有关更多信息，请参阅《[公交网关指南](#)》中的 [Transit Gateway 路由表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateTransitGatewayRoute](#) 中的。

## create-transit-gateway-vpc-attachment

以下代码示例演示了如何使用 `create-transit-gateway-vpc-attachment`。

### AWS CLI

示例 1：将传输网关与 VPC 关联

以下 `create-transit-gateway-vpc-attachment` 示例创建与指定 VPC 的中转网关连接。

```
aws ec2 create-transit-gateway-vpc-attachment \
  --transit-gateway-id tgw-0262a0e521EXAMPLE \
  --vpc-id vpc-07e8ffd50f49335df \
  --subnet-id subnet-0752213d59EXAMPLE
```

输出：

```
{
  "TransitGatewayVpcAttachment": {
    "TransitGatewayAttachmentId": "tgw-attach-0a34fe6b4fEXAMPLE",
    "TransitGatewayId": "tgw-0262a0e521EXAMPLE",
    "VpcId": "vpc-07e8ffd50fEXAMPLE",
    "VpcOwnerId": "111122223333",
    "State": "pending",
    "SubnetIds": [
      "subnet-0752213d59EXAMPLE"
    ],
    "CreationTime": "2019-07-10T17:33:46.000Z",
    "Options": {
      "DnsSupport": "enable",
      "Ipv6Support": "disable"
    }
  }
}
```

```
}
```

有关更多信息，请参阅 [《传输网关指南》中的创建连接到 VPC 的中转网关](#)。

示例 2：将传输网关与 VPC 中的多个子网关联

以下 `create-transit-gateway-vpc-attachment` 示例创建与指定 VPC 和子网的中转网关连接。

```
aws ec2 create-transit-gateway-vpc-attachment \  
  --transit-gateway-id tgw-02f776b1a7EXAMPLE \  
  --vpc-id vpc-3EXAMPLE \  
  --subnet-ids "subnet-dEXAMPLE" "subnet-6EXAMPLE"
```

输出：

```
{  
  "TransitGatewayVpcAttachment": {  
    "TransitGatewayAttachmentId": "tgw-attach-0e141e0bebEXAMPLE",  
    "TransitGatewayId": "tgw-02f776b1a7EXAMPLE",  
    "VpcId": "vpc-3EXAMPLE",  
    "VpcOwnerId": "111122223333",  
    "State": "pending",  
    "SubnetIds": [  
      "subnet-6EXAMPLE",  
      "subnet-dEXAMPLE"  
    ],  
    "CreationTime": "2019-12-17T20:07:52.000Z",  
    "Options": {  
      "DnsSupport": "enable",  
      "Ipv6Support": "disable"  
    }  
  }  
}
```

有关更多信息，请参阅 [《传输网关指南》中的创建连接到 VPC 的中转网关](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateTransitGatewayVpcAttachment](#) 中的。

## create-transit-gateway

以下代码示例演示了如何使用 `create-transit-gateway`。

## AWS CLI

### 创建公交网关

以下create-transit-gateway示例创建了一个传输网关。

```
aws ec2 create-transit-gateway \  
  --description MyTGW \  
  --options  
  AmazonSideAsn=64516,AutoAcceptSharedAttachments=enable,DefaultRouteTableAssociation=enable,
```

输出：

```
{  
  "TransitGateway": {  
    "TransitGatewayId": "tgw-0262a0e521EXAMPLE",  
    "TransitGatewayArn": "arn:aws:ec2:us-east-2:111122223333:transit-gateway/  
tgw-0262a0e521EXAMPLE",  
    "State": "pending",  
    "OwnerId": "111122223333",  
    "Description": "MyTGW",  
    "CreationTime": "2019-07-10T14:02:12.000Z",  
    "Options": {  
      "AmazonSideAsn": 64516,  
      "AutoAcceptSharedAttachments": "enable",  
      "DefaultRouteTableAssociation": "enable",  
      "AssociationDefaultRouteTableId": "tgw-rtb-018774adf3EXAMPLE",  
      "DefaultRouteTablePropagation": "enable",  
      "PropagationDefaultRouteTableId": "tgw-rtb-018774adf3EXAMPLE",  
      "VpnEcmpSupport": "enable",  
      "DnsSupport": "enable"  
    }  
  }  
}
```

有关更多信息，请参阅 [《传输网关指南》中的创建传输网关](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTransitGateway](#)中的。

### create-verified-access-endpoint

以下代码示例演示了如何使用 create-verified-access-endpoint。



## AWS CLI

### 创建已验证访问终端节点

以下`create-verified-access-endpoint`示例为指定的已验证访问权限组创建已验证访问权限终端节点。指定的网络接口和安全组必须属于同一 VPC。

```
aws ec2 create-verified-access-endpoint \
  --verified-access-group-id vagr-0dbe967baf14b7235 \
  --endpoint-type network-interface \
  --attachment-type vpc \
  --domain-certificate-arn arn:aws:acm:us-east-2:123456789012:certificate/
eb065ea0-26f9-4e75-a6ce-0a1a7EXAMPLE \
  --application-domain example.com \
  --endpoint-domain-prefix my-ava-app \
  --security-group-ids sg-004915970c4c8f13a \
  --network-interface-options
NetworkInterfaceId=eni-0aec70418c8d87a0f,Protocol=https,Port=443 \
  --tag-specifications ResourceType=verified-access-
endpoint,Tags=[{Key=Name,Value=my-va-endpoint}]
```

输出：

```
{
  "VerifiedAccessEndpoint": {
    "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",
    "VerifiedAccessGroupId": "vagr-0dbe967baf14b7235",
    "VerifiedAccessEndpointId": "vae-066fac616d4d546f2",
    "ApplicationDomain": "example.com",
    "EndpointType": "network-interface",
    "AttachmentType": "vpc",
    "DomainCertificateArn": "arn:aws:acm:us-east-2:123456789012:certificate/
eb065ea0-26f9-4e75-a6ce-0a1a7EXAMPLE",
    "EndpointDomain": "my-ava-
app.edge-00c3372d53b1540bb.vai-0ce000c0b7643abea.prod.verified-access.us-
east-2.amazonaws.com",
    "SecurityGroupIds": [
      "sg-004915970c4c8f13a"
    ],
    "NetworkInterfaceOptions": {
      "NetworkInterfaceId": "eni-0aec70418c8d87a0f",
      "Protocol": "https",
      "Port": 443
    }
  }
}
```

```

    },
    "Status": {
      "Code": "pending"
    },
    "Description": "",
    "CreationTime": "2023-08-25T20:54:43",
    "LastUpdatedTime": "2023-08-25T20:54:43",
    "Tags": [
      {
        "Key": "Name",
        "Value": "my-va-endpoint"
      }
    ]
  }
}

```

有关更多信息，请参阅 [《已验证访问用户指南》](#) 中的 [AWS 已验证访问终端节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateVerifiedAccessEndpoint](#) 中的。

## create-verified-access-group

以下代码示例演示了如何使用 `create-verified-access-group`。

### AWS CLI

创建“已验证访问权限”组

以下 `create-verified-access-group` 示例为指定的“已验证访问权限”实例创建已验证访问权限组。

```

aws ec2 create-verified-access-group \
  --verified-access-instance-id vai-0ce000c0b7643abea \
  --tag-specifications ResourceType=verified-access-
group,Tags=[{Key=Name,Value=my-va-group}]

```

输出：

```

{
  "VerifiedAccessGroup": {
    "VerifiedAccessGroupId": "vagr-0dbe967baf14b7235",
    "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",

```

```
    "Description": "",
    "Owner": "123456789012",
    "VerifiedAccessGroupArn": "arn:aws:ec2:us-east-2:123456789012:verified-
access-group/vagr-0dbe967baf14b7235",
    "CreationTime": "2023-08-25T19:55:19",
    "LastUpdatedTime": "2023-08-25T19:55:19",
    "Tags": [
      {
        "Key": "Name",
        "Value": "my-va-group"
      }
    ]
  }
}
```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVerifiedAccessGroup](#)中的。

## create-verified-access-instance

以下代码示例演示了如何使用 create-verified-access-instance。

### AWS CLI

#### 创建已验证访问权限实例

以下create-verified-access-instance示例创建了一个带有 Name 标签的已验证访问实例。

```
aws ec2 create-verified-access-instance \
  --tag-specifications ResourceType=verified-access-
instance,Tags=[{Key=Name,Value=my-va-instance}]
```

输出：

```
{
  "VerifiedAccessInstance": {
    "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",
    "Description": "",
    "VerifiedAccessTrustProviders": [],
    "CreationTime": "2023-08-25T18:27:56",
```

```
    "LastUpdatedTime": "2023-08-25T18:27:56",
    "Tags": [
      {
        "Key": "Name",
        "Value": "my-va-instance"
      }
    ]
  }
}
```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateVerifiedAccessInstance](#) 中的。

## create-verified-access-trust-provider

以下代码示例演示了如何使用 create-verified-access-trust-provider。

### AWS CLI

#### 创建已验证访问信任提供商

以下 create-verified-access-trust-provider 示例使用 Identity Center 设置了经过验证的访问信任提供商。

```
aws ec2 create-verified-access-trust-provider \
  --trust-provider-type user \
  --user-trust-provider-type iam-identity-center \
  --policy-reference-name idc \
  --tag-specifications ResourceType=verified-access-trust-
  provider,Tags=[{Key=Name,Value=my-va-trust-provider}]
```

输出：

```
{
  "VerifiedAccessTrustProvider": {
    "VerifiedAccessTrustProviderId": "vatp-0bb32de759a3e19e7",
    "Description": "",
    "TrustProviderType": "user",
    "UserTrustProviderType": "iam-identity-center",
    "PolicyReferenceName": "idc",
    "CreationTime": "2023-08-25T18:40:36",
```

```
    "LastUpdatedTime": "2023-08-25T18:40:36",
    "Tags": [
      {
        "Key": "Name",
        "Value": "my-va-trust-provider"
      }
    ]
  }
}
```

有关更多信息，请参阅《已验证访问用户指南》中的“AWS 验证访问权限的[信任提供商](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateVerifiedAccessTrustProvider](#) 中的。

## create-volume

以下代码示例演示了如何使用 create-volume。

### AWS CLI

创建空的通用固态硬盘 (gp2) 卷

以下 create-volume 示例在指定的可用区创建一个 80 GiB 的通用型 SSD (gp2) 卷。请注意，当前区域必须是 us-east-1，或者您可以添加 --region 参数来为命令指定区域。

```
aws ec2 create-volume \
  --volume-type gp2 \
  --size 80 \
  --availability-zone us-east-1a
```

输出：

```
{
  "AvailabilityZone": "us-east-1a",
  "Tags": [],
  "Encrypted": false,
  "VolumeType": "gp2",
  "VolumeId": "vol-1234567890abcdef0",
  "State": "creating",
  "Iops": 240,
  "SnapshotId": "",
  "CreateTime": "YYYY-MM-DDTHH:MM:SS.000Z",
```

```
    "Size": 80
  }
```

如果您未指定卷类型，则默认卷类型为gp2。

```
aws ec2 create-volume \  
  --size 80 \  
  --availability-zone us-east-1a
```

示例 2：使用快照创建预配置 IOPS 固态硬盘 (io1) 卷

以下create-volume示例使用指定的快照在指定可用区中创建一个预配置 IOPS SSD (io1) 卷，该卷具有 1000 个预配置 IOPS SSD (io1)。

```
aws ec2 create-volume \  
  --volume-type io1 \  
  --iops 1000 \  
  --snapshot-id snap-066877671789bd71b \  
  --availability-zone us-east-1a
```

输出：

```
{  
  "AvailabilityZone": "us-east-1a",  
  "Tags": [],  
  "Encrypted": false,  
  "VolumeType": "io1",  
  "VolumeId": "vol-1234567890abcdef0",  
  "State": "creating",  
  "Iops": 1000,  
  "SnapshotId": "snap-066877671789bd71b",  
  "CreateTime": "YYYY-MM-DDTHH:MM:SS.000Z",  
  "Size": 500  
}
```

示例 3：创建加密卷

以下create-volume示例使用用于 EBS 加密的默认 CMK 创建加密卷。如果默认情况下加密处于禁用状态，则必须按以下方式指定--encrypted参数。

```
aws ec2 create-volume \  
  --encrypted
```

```
--size 80 \  
--encrypted \  
--availability-zone us-east-1a
```

输出：

```
{  
  "AvailabilityZone": "us-east-1a",  
  "Tags": [],  
  "Encrypted": true,  
  "VolumeType": "gp2",  
  "VolumeId": "vol-1234567890abcdef0",  
  "State": "creating",  
  "Iops": 240,  
  "SnapshotId": "",  
  "CreateTime": "YYYY-MM-DDTHH:MM:SS.000Z",  
  "Size": 80  
}
```

如果默认启用了加密，则以下示例命令将创建加密卷，即使没有--encrypted参数也是如此。

```
aws ec2 create-volume \  
  --size 80 \  
  --availability-zone us-east-1a
```

如果您使用--kms-key-id参数来指定客户管理的 CMK，则即使默认启用了加密，也必须指定该--encrypted参数。

```
aws ec2 create-volume \  
  --volume-type gp2 \  
  --size 80 \  
  --encrypted \  
  --kms-key-id 0ea3fef3-80a7-4778-9d8c-1c0c6EXAMPLE \  
  --availability-zone us-east-1a
```

示例 4：创建带有标签的卷

以下create-volume示例创建了一个卷并添加了两个标签。

```
aws ec2 create-volume \  
  --size 80 \  
  --availability-zone us-east-1a
```

```
--availability-zone us-east-1a \  
--volume-type gp2 \  
--size 80 \  
--tag-specifications 'ResourceType=volume,Tags=[{Key=purpose,Value=production},  
{Key=cost-center,Value=cc123}]'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVolume](#)中的。

## create-vpc-endpoint-connection-notification

以下代码示例演示了如何使用 create-vpc-endpoint-connection-notification。

### AWS CLI

#### 创建端点连接通知

此示例为特定的终端节点服务创建通知，当接口终端节点已连接到您的服务以及您的服务已接受终端节点时，该通知会提醒您。

命令：

```
aws ec2 create-vpc-endpoint-connection-notification --connection-notification-arn  
arn:aws:sns:us-east-2:123456789012:VpceNotification --connection-events Connect  
Accept --service-id vpce-svc-1237881c0d25a3abc
```

输出：

```
{  
  "ConnectionNotification": {  
    "ConnectionNotificationState": "Enabled",  
    "ConnectionNotificationType": "Topic",  
    "ServiceId": "vpce-svc-1237881c0d25a3abc",  
    "ConnectionEvents": [  
      "Accept",  
      "Connect"  
    ],  
    "ConnectionNotificationId": "vpce-nfn-008776de7e03f5abc",  
    "ConnectionNotificationArn": "arn:aws:sns:us-  
east-2:123456789012:VpceNotification"  
  }  
}
```



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVpcEndpointConnectionNotification](#)中的。

## create-vpc-endpoint-service-configuration

以下代码示例演示了如何使用 create-vpc-endpoint-service-configuration。

### AWS CLI

示例 1：为接口终端节点创建终端节点服务配置

以下create-vpc-endpoint-service-configuration示例使用 Network Load Balancer 创建 VPC 终端节点服务配置nlb-vpce。此示例还指定必须接受通过接口终端节点连接到服务的请求。

```
aws ec2 create-vpc-endpoint-service-configuration \
  --network-load-balancer-arns arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/nlb-vpce/e94221227f1ba532 \
  --acceptance-required
```

输出：

```
{
  "ServiceConfiguration": {
    "ServiceType": [
      {
        "ServiceType": "Interface"
      }
    ],
    "NetworkLoadBalancerArns": [
      "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/net/
nlb-vpce/e94221227f1ba532"
    ],
    "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-03d5ebb7d9579a2b3",
    "ServiceState": "Available",
    "ServiceId": "vpce-svc-03d5ebb7d9579a2b3",
    "AcceptanceRequired": true,
    "AvailabilityZones": [
      "us-east-1d"
    ],
    "BaseEndpointDnsNames": [
```

```

        "vpce-svc-03d5ebb7d9579a2b3.us-east-1.vpce.amazonaws.com"
    ]
}
}

```

## 示例 2：为 Gateway Load Balancer 终端节点创建终端节点服务配置

以下 `create-vpc-endpoint-service-configuration` 示例使用 Gateway Load Balancer 创建 VPC 终端节点服务配置 `GWLBService`。系统会自动接受通过 Gateway Load Balancer 端点连接到服务的请求。

```

aws ec2 create-vpc-endpoint-service-configuration \
  --gateway-load-balancer-arns arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/gwy/GWLBService/123123123123abcc \
  --no-acceptance-required

```

输出：

```

{
  "ServiceConfiguration": {
    "ServiceType": [
      {
        "ServiceType": "GatewayLoadBalancer"
      }
    ],
    "ServiceId": "vpce-svc-123123a1c43abc123",
    "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123",
    "ServiceState": "Available",
    "AvailabilityZones": [
      "us-east-1d"
    ],
    "AcceptanceRequired": false,
    "ManagesVpcEndpoints": false,
    "GatewayLoadBalancerArns": [
      "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/gwy/
GWLBService/123123123123abcc"
    ]
  }
}

```

有关更多信息，请参阅 Amazon VPC 用户指南中的 VPC [终端节点服务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVpcEndpointServiceConfiguration](#)中的。

## create-vpc-endpoint

以下代码示例演示了如何使用 create-vpc-endpoint。

### AWS CLI

#### 示例 1：创建网关终端节点

以下create-vpc-endpoint示例us-east-1在该区域的 VPC vpc-1a2b3c4d 和 Amazon S3 之间创建网关 VPC 终端节点，并将路由表rtb-11aa22bb与该终端节点关联起来。

```
aws ec2 create-vpc-endpoint \  
  --vpc-id vpc-1a2b3c4d \  
  --service-name com.amazonaws.us-east-1.s3 \  
  --route-table-ids rtb-11aa22bb
```

输出：

```
{  
  "VpcEndpoint": {  
    "PolicyDocument": "{\n\"Version\": \"2008-10-17\",\n\"Statement\": [\n{\n\"Sid\": \"\",  
\n\"Effect\": \"Allow\",\n\"Principal\": \"*\",\n\"Action\": \"*\",\n\"Resource\": \"*\"}\n]}",  
    "VpcId": "vpc-1a2b3c4d",  
    "State": "available",  
    "ServiceName": "com.amazonaws.us-east-1.s3",  
    "RouteTableIds": [  
      "rtb-11aa22bb"  
    ],  
    "VpcEndpointId": "vpc-1a2b3c4d",  
    "CreationTimestamp": "2015-05-15T09:40:50Z"  
  }  
}
```

有关更多信息，请参阅AWS PrivateLink 指南中的[创建网关终端节点](#)。

#### 示例 2：创建接口终端节点

以下create-vpc-endpoint示例us-east-1在该区域的 VPC vpc-1a2b3c4d 和 Amazon S3 之间创建了一个接口 VPC 终端节点。该命令在子网中创建终端节点subnet-1a2b3c4d，将其与安全组关联sg-1a2b3c4d，并添加密钥为“服务”、值为“S3”的标签。

```
aws ec2 create-vpc-endpoint \  
  --vpc-id vpc-1a2b3c4d \  
  --vpc-endpoint-type Interface \  
  --service-name com.amazonaws.us-east-1.s3 \  
  --subnet-ids subnet-7b16de0c \  
  --security-group-id sg-1a2b3c4d \  
  --tag-specifications ResourceType=vpc-endpoint,Tags=[{Key=service,Value=S3}]
```

输出：

```
{  
  "VpcEndpoint": {  
    "VpcEndpointId": "vpce-1a2b3c4d5e6f1a2b3",  
    "VpcEndpointType": "Interface",  
    "VpcId": "vpc-1a2b3c4d",  
    "ServiceName": "com.amazonaws.us-east-1.s3",  
    "State": "pending",  
    "RouteTableIds": [],  
    "SubnetIds": [  
      "subnet-1a2b3c4d"  
    ],  
    "Groups": [  
      {  
        "GroupId": "sg-1a2b3c4d",  
        "GroupName": "default"  
      }  
    ],  
    "PrivateDnsEnabled": false,  
    "RequesterManaged": false,  
    "NetworkInterfaceIds": [  
      "eni-0b16f0581c8ac6877"  
    ],  
    "DnsEntries": [  
      {  
        "DnsName": "*.vpce-1a2b3c4d5e6f1a2b3-9hnenorg.s3.us-  
east-1.vpce.amazonaws.com",  
        "HostedZoneId": "Z7HUB22UULQXV"  
      },  
      {  
        "DnsName": "*.vpce-1a2b3c4d5e6f1a2b3-9hnenorg-us-east-1c.s3.us-  
east-1.vpce.amazonaws.com",  
        "HostedZoneId": "Z7HUB22UULQXV"  
      }  
    ]  
  }  
}
```

```

    ],
    "CreationTimestamp": "2021-03-05T14:46:16.030000+00:00",
    "Tags": [
      {
        "Key": "service",
        "Value": "S3"
      }
    ],
    "OwnerId": "123456789012"
  }
}

```

有关更多信息，请参阅《用户指南》中的[创建接口端点](#) AWS PrivateLink。

### 示例 3：创建 Gateway Load Balancer 端点

以下create-vpc-endpoint示例在 VPC vpc-111122223333aabbcc 和之间创建网关负载均衡器终端节点，以及使用网关负载均衡器配置的服务。

```

aws ec2 create-vpc-endpoint \
  --service-name com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123 \
  --vpc-endpoint-type GatewayLoadBalancer \
  --vpc-id vpc-111122223333aabbcc \
  --subnet-ids subnet-0011aabbcc2233445

```

输出：

```

{
  "VpcEndpoint": {
    "VpcEndpointId": "vpce-aabbaabbaabbaabba",
    "VpcEndpointType": "GatewayLoadBalancer",
    "VpcId": "vpc-111122223333aabbcc",
    "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123",
    "State": "pending",
    "SubnetIds": [
      "subnet-0011aabbcc2233445"
    ],
    "RequesterManaged": false,
    "NetworkInterfaceIds": [
      "eni-01010120203030405"
    ],
    "CreationTimestamp": "2020-11-11T08:06:03.522Z",
    "OwnerId": "123456789012"
  }
}

```

```
}  
}
```

有关更多信息，请参阅《用户指南》中的 [Gateway Load Balancer 终端节点 AWS PrivateLink](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateVpcEndpoint](#) 中的。

## create-vpc-peering-connection

以下代码示例演示了如何使用 create-vpc-peering-connection。

### AWS CLI

在您的 VPC 之间创建 VPC 对等连接

此示例请求您的 vpc-1a2b3c4d 和 vpc-11122233 之间建立对等连接。

命令:

```
aws ec2 create-vpc-peering-connection --vpc-id vpc-1a2b3c4d --peer-vpc-id  
vpc-11122233
```

输出:

```
{  
  "VpcPeeringConnection": {  
    "Status": {  
      "Message": "Initiating Request to 444455556666",  
      "Code": "initiating-request"  
    },  
    "Tags": [],  
    "RequesterVpcInfo": {  
      "OwnerId": "444455556666",  
      "VpcId": "vpc-1a2b3c4d",  
      "CidrBlock": "10.0.0.0/28"  
    },  
    "VpcPeeringConnectionId": "pcx-111aaa111",  
    "ExpirationTime": "2014-04-02T16:13:36.000Z",  
    "AccepterVpcInfo": {  
      "OwnerId": "444455556666",  
      "VpcId": "vpc-11122233"  
    }  
  }  
}
```

```
}
```

### 与其他账户中的 VPC 创建 VPC 对等连接

此示例请求在您的 VPC (vpc-1a2b3c4d) 和属于账户 123456789012 的 VPC (vpc-11122233) 之间建立对等连接。AWS

命令:

```
aws ec2 create-vpc-peering-connection --vpc-id vpc-1a2b3c4d --peer-vpc-id  
vpc-11122233 --peer-owner-id 123456789012
```

### 与其他区域的 VPC 创建 VPC 对等连接

此示例请求在当前区域 (vpc-1a2b3c4d) 的 VPC 与您在该区域的账户中的 VPC (vpc-11122233) 之间建立对等连接。us-west-2

命令:

```
aws ec2 create-vpc-peering-connection --vpc-id vpc-1a2b3c4d --peer-vpc-id  
vpc-11122233 --peer-region us-west-2
```

此示例请求在当前区域 (vpc-1a2b3c4d) 的 VPC 与属于该区域账户 123456789012 的 VPC (vpc-11122233) 之间建立对等连接。AWS us-west-2

命令:

```
aws ec2 create-vpc-peering-connection --vpc-id vpc-1a2b3c4d --peer-vpc-id  
vpc-11122233 --peer-owner-id 123456789012 --peer-region us-west-2
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVpcPeeringConnection](#)中的。

## create-vpc

以下代码示例演示了如何使用 create-vpc。

### AWS CLI

#### 示例 1：创建 VPC

以下 create-vpc 示例创建具有指定 IPv4 CIDR 块和名称标签的 VPC。

```
aws ec2 create-vpc \  
  --cidr-block 10.0.0.0/16 \  
  --tag-specification ResourceType=vpc,Tags=[{Key=Name,Value=MyVpc}]
```

输出：

```
{  
  "Vpc": {  
    "CidrBlock": "10.0.0.0/16",  
    "DhcpOptionsId": "dopt-5EXAMPLE",  
    "State": "pending",  
    "VpcId": "vpc-0a60eb65b4EXAMPLE",  
    "OwnerId": "123456789012",  
    "InstanceTenancy": "default",  
    "Ipv6CidrBlockAssociationSet": [],  
    "CidrBlockAssociationSet": [  
      {  
        "AssociationId": "vpc-cidr-assoc-07501b79ecEXAMPLE",  
        "CidrBlock": "10.0.0.0/16",  
        "CidrBlockState": {  
          "State": "associated"  
        }  
      }  
    ],  
    "IsDefault": false,  
    "Tags": [  
      {  
        "Key": "Name",  
        "Value": "MyVpc"  
      }  
    ]  
  }  
}
```

## 示例 2：创建具有专用租赁的 VPC

以下 create-vpc 示例创建具有指定 IPv4 CIDR 块和专用租赁的 VPC。

```
aws ec2 create-vpc \  
  --cidr-block 10.0.0.0/16 \  
  --instance-tenancy dedicated
```



输出：

```
{
  "Vpc": {
    "CidrBlock": "10.0.0.0/16",
    "DhcpOptionsId": "dopt-19edf471",
    "State": "pending",
    "VpcId": "vpc-0a53287fa4EXAMPLE",
    "OwnerId": "111122223333",
    "InstanceTenancy": "dedicated",
    "Ipv6CidrBlockAssociationSet": [],
    "CidrBlockAssociationSet": [
      {
        "AssociationId": "vpc-cidr-assoc-00b24cc1c2EXAMPLE",
        "CidrBlock": "10.0.0.0/16",
        "CidrBlockState": {
          "State": "associated"
        }
      }
    ],
    "IsDefault": false
  }
}
```

示例 3：创建具有 IPv6 CIDR 块的 VPC

以下 `create-vpc` 示例创建具有 Amazon 提供的 IPv6 CIDR 块的 VPC。

```
aws ec2 create-vpc \
  --cidr-block 10.0.0.0/16 \
  --amazon-provided-ipv6-cidr-block
```

输出：

```
{
  "Vpc": {
    "CidrBlock": "10.0.0.0/16",
    "DhcpOptionsId": "dopt-dEXAMPLE",
    "State": "pending",
    "VpcId": "vpc-0fc5e3406bEXAMPLE",
    "OwnerId": "123456789012",
    "InstanceTenancy": "default",
    "Ipv6CidrBlockAssociationSet": [
```

```

    {
      "AssociationId": "vpc-cidr-assoc-068432c60bEXAMPLE",
      "Ipv6CidrBlock": "",
      "Ipv6CidrBlockState": {
        "State": "associating"
      },
      "Ipv6Pool": "Amazon",
      "NetworkBorderGroup": "us-west-2"
    }
  ],
  "CidrBlockAssociationSet": [
    {
      "AssociationId": "vpc-cidr-assoc-0669f8f9f5EXAMPLE",
      "CidrBlock": "10.0.0.0/16",
      "CidrBlockState": {
        "State": "associated"
      }
    }
  ],
  "IsDefault": false
}
}

```

#### 示例 4：从 IPAM 池中创建具有 CIDR 的 VPC

以下 `create-vpc` 示例从 Amazon VPC IP 地址管理器 (IPAM) 池，创建具有 CIDR 的 VPC。

Linux 和 macOS：

```

aws ec2 create-vpc \
  --ipv4-ipam-pool-id ipam-pool-0533048da7d823723 \
  --tag-specifications ResourceType=vpc,Tags='[{"Key=Environment,Value="Preprod"}, {"Key=Owner,Value="Build Team"}]'
```

Windows:

```

aws ec2 create-vpc ^
  --ipv4-ipam-pool-id ipam-pool-0533048da7d823723 ^
  --tag-specifications ResourceType=vpc,Tags=[{"Key=Environment,Value="Preprod"}, {"Key=Owner,Value="Build Team"}]
```

输出：

```
{
  "Vpc": {
    "CidrBlock": "10.0.1.0/24",
    "DhcpOptionsId": "dopt-2afccf50",
    "State": "pending",
    "VpcId": "vpc-010e1791024eb0af9",
    "OwnerId": "123456789012",
    "InstanceTenancy": "default",
    "Ipv6CidrBlockAssociationSet": [],
    "CidrBlockAssociationSet": [
      {
        "AssociationId": "vpc-cidr-assoc-0a77de1d803226d4b",
        "CidrBlock": "10.0.1.0/24",
        "CidrBlockState": {
          "State": "associated"
        }
      }
    ],
    "IsDefault": false,
    "Tags": [
      {
        "Key": "Environment",
        "Value": "Preprod"
      },
      {
        "Key": "Owner",
        "Value": "Build Team"
      }
    ]
  }
}
```

有关更多信息，请参阅《Amazon VPC IPAM 用户指南》中的[创建使用 IPAM 池 CIDR 的 VPC](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVpc](#)中的。

## create-vpn-connection-route

以下代码示例演示了如何使用 create-vpn-connection-route。

### AWS CLI

为 VPN 连接创建静态路由

此示例为指定的 VPN 连接创建静态路由。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 create-vpn-connection-route --vpn-connection-id vpn-40f41529 --destination-cidr-block 11.12.0.0/16
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVpnConnectionRoute](#)中的。

## create-vpn-connection

以下代码示例演示了如何使用 create-vpn-connection。

### AWS CLI

示例 1：使用动态路由创建 VPN 连接

以下create-vpn-connection示例在指定的虚拟专用网关和指定的客户网关之间创建 VPN 连接，并将标签应用于 VPN 连接。输出包括 XML 格式的客户网关设备的配置信息。

```
aws ec2 create-vpn-connection \  
  --type ipsec.1 \  
  --customer-gateway-id cgw-001122334455aabbcc \  
  --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \  
  --tag-specification 'ResourceType=vpn-connection,Tags=[{Key=Name,Value=BGP-VPN}]'
```

输出：

```
{  
  "VpnConnection": {  
    "CustomerGatewayConfiguration": "...configuration information...",  
    "CustomerGatewayId": "cgw-001122334455aabbcc",  
    "Category": "VPN",  
    "State": "pending",  
    "VpnConnectionId": "vpn-123123123123abcab",  
    "VpnGatewayId": "vgw-1a1a1a1a1a1a2b2b2",  
    "Options": {  
      "EnableAcceleration": false,  
      "StaticRoutesOnly": false,  
      "LocalIpv4NetworkCidr": "0.0.0.0/0",
```

```

    "RemoteIpv4NetworkCidr": "0.0.0.0/0",
    "TunnelInsideIpVersion": "ipv4",
    "TunnelOptions": [
        {},
        {}
    ]
  },
  "Routes": [],
  "Tags": [
    {
      "Key": "Name",
      "Value": "BGP-VPN"
    }
  ]
}
}

```

有关更多信息，请参阅站点到站点 VPN 用户[指南中的 AWS 站点到站点 VPN 的工作原理](#)。AWS

#### 示例 2：使用静态路由创建 VPN 连接

以下create-vpn-connection示例在指定的虚拟专用网关和指定的客户网关之间创建 VPN 连接。这些选项指定静态路由。输出包括 XML 格式的客户网关设备的配置信息。

```

aws ec2 create-vpn-connection \
  --type ipsec.1 \
  --customer-gateway-id cgw-001122334455aabbc \
  --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
  --options "{\"StaticRoutesOnly\":true}"

```

输出：

```

{
  "VpnConnection": {
    "CustomerGatewayConfiguration": "..configuration information...",
    "CustomerGatewayId": "cgw-001122334455aabbc",
    "Category": "VPN",
    "State": "pending",
    "VpnConnectionId": "vpn-123123123123abcab",
    "VpnGatewayId": "vgw-1a1a1a1a1a1a2b2b2",
    "Options": {
      "EnableAcceleration": false,

```

```

        "StaticRoutesOnly": true,
        "LocalIpv4NetworkCidr": "0.0.0.0/0",
        "RemoteIpv4NetworkCidr": "0.0.0.0/0",
        "TunnelInsideIpVersion": "ipv4",
        "TunnelOptions": [
            {},
            {}
        ]
    },
    "Routes": [],
    "Tags": []
}
}

```

有关更多信息，请参阅站点到站点 VPN 用户[指南中的 AWS 站点到站点 VPN 的工作原理](#)。AWS

示例 3：创建 VPN 连接并指定自己的内部 CIDR 和预共享密钥

以下 `create-vpn-connection` 示例创建 VPN 连接，并为每个隧道指定内部 IP 地址 CIDR 块和自定义预共享密钥。指定的值将在 `CustomerGatewayConfiguration` 信息中返回。

```

aws ec2 create-vpn-connection \
  --type ipsec.1 \
  --customer-gateway-id cgw-001122334455aabbcc \
  --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
  --options
  TunnelOptions='[{TunnelInsideCidr=169.254.12.0/30,PreSharedKey=ExamplePreSharedKey1},
  {TunnelInsideCidr=169.254.13.0/30,PreSharedKey=ExamplePreSharedKey2}]'

```

输出：

```

{
  "VpnConnection": {
    "CustomerGatewayConfiguration": "..configuration information...",
    "CustomerGatewayId": "cgw-001122334455aabbcc",
    "Category": "VPN",
    "State": "pending",
    "VpnConnectionId": "vpn-123123123123abcab",
    "VpnGatewayId": "vgw-1a1a1a1a1a1a2b2b2",
    "Options": {
      "EnableAcceleration": false,
      "StaticRoutesOnly": false,
    }
  }
}

```

```

    "LocalIpv4NetworkCidr": "0.0.0.0/0",
    "RemoteIpv4NetworkCidr": "0.0.0.0/0",
    "TunnelInsideIpVersion": "ipv4",
    "TunnelOptions": [
      {
        "OutsideIpAddress": "203.0.113.3",
        "TunnelInsideCidr": "169.254.12.0/30",
        "PreSharedKey": "ExamplePreSharedKey1"
      },
      {
        "OutsideIpAddress": "203.0.113.5",
        "TunnelInsideCidr": "169.254.13.0/30",
        "PreSharedKey": "ExamplePreSharedKey2"
      }
    ]
  },
  "Routes": [],
  "Tags": []
}

```

有关更多信息，请参阅站点到站点 VPN 用户 [指南中的 AWS 站点到站点 VPN 的工作原理](#)。AWS

#### 示例 4：创建支持 IPv6 流量的 VPN 连接

以下 `create-vpn-connection` 示例创建了一个 VPN 连接，该连接支持指定传输网关和指定客户网关之间的 IPv6 流量。两条隧道的隧道选项都指定了 AWS 必须启动 IKE 协商的隧道。

```

aws ec2 create-vpn-connection \
  --type ipsec.1 \
  --transit-gateway-id tgw-12312312312312312 \
  --customer-gateway-id cgw-001122334455aabbc \
  --options TunnelInsideIpVersion=ipv6,TunnelOptions=[{StartupAction=start},
  {StartupAction=start}]

```

输出：

```

{
  "VpnConnection": {
    "CustomerGatewayConfiguration": "..configuration information...",
    "CustomerGatewayId": "cgw-001122334455aabbc",
    "Category": "VPN",

```

```
"State": "pending",
"VpnConnectionId": "vpn-1111111122222222",
"TransitGatewayId": "tgw-12312312312312312",
"Options": {
  "EnableAcceleration": false,
  "StaticRoutesOnly": false,
  "LocalIpv6NetworkCidr": "::/0",
  "RemoteIpv6NetworkCidr": "::/0",
  "TunnelInsideIpVersion": "ipv6",
  "TunnelOptions": [
    {
      "OutsideIpAddress": "203.0.113.3",
      "StartupAction": "start"
    },
    {
      "OutsideIpAddress": "203.0.113.5",
      "StartupAction": "start"
    }
  ]
},
"Routes": [],
"Tags": []
}
```

有关更多信息，请参阅站点到站点 VPN 用户指南中的 [AWS 站点到站点 VPN 的工作原理](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateVpnConnection](#) 中的。

## create-vpn-gateway

以下代码示例演示了如何使用 create-vpn-gateway。

### AWS CLI

#### 创建虚拟专用网关

此示例创建了一个虚拟专用网关。

命令:

```
aws ec2 create-vpn-gateway --type ipsec.1
```



输出：

```
{
  "VpnGateway": {
    "AmazonSideAsn": 64512,
    "State": "available",
    "Type": "ipsec.1",
    "VpnGatewayId": "vgw-9a4cacf3",
    "VpcAttachments": []
  }
}
```

使用特定亚马逊端 ASN 创建虚拟专用网关

此示例创建了一个虚拟私有网关，并为 BGP 会话的 Amazon 端指定自治系统编号 (ASN)。

命令：

```
aws ec2 create-vpn-gateway --type ipsec.1 --amazon-side-asn 65001
```

输出：

```
{
  "VpnGateway": {
    "AmazonSideAsn": 65001,
    "State": "available",
    "Type": "ipsec.1",
    "VpnGatewayId": "vgw-9a4cacf3",
    "VpcAttachments": []
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVpnGateway](#)中的。

## delete-carrier-gateway

以下代码示例演示了如何使用 delete-carrier-gateway。

### AWS CLI

删除您的运营商网关

以下`delete-carrier-gateway`示例删除了指定的运营商网关。

```
aws ec2 delete-carrier-gateway \  
  --carrier-gateway-id cagw-0465cdEXAMPLE1111
```

输出：

```
{  
  "CarrierGateway": {  
    "CarrierGatewayId": "cagw-0465cdEXAMPLE1111",  
    "VpcId": "vpc-0c529aEXAMPLE1111",  
    "State": "deleting",  
    "OwnerId": "123456789012"  
  }  
}
```

有关更多信息，请参阅 Amazon Virtual Private Cloud 用户指南中的[运营商网关](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCarrierGateway](#)中的。

## delete-client-vpn-endpoint

以下代码示例演示了如何使用 `delete-client-vpn-endpoint`。

AWS CLI

删除 Client VPN 端点

以下`delete-client-vpn-endpoint`示例删除指定的 Client VPN 端点。

```
aws ec2 delete-client-vpn-endpoint \  
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde
```

输出：

```
{  
  "Status": {  
    "Code": "deleting"  
  }  
}
```

有关更多信息，请参阅《[客户端 VPN 管理员指南](#)》中的[AWS Client VPN 端点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteClientVpnEndpoint](#)中的。

## delete-client-vpn-route

以下代码示例演示了如何使用 delete-client-vpn-route。

### AWS CLI

删除 Client VPN 终端节点的路由

以下delete-client-vpn-route示例删除了 Client VPN 终端节点的指定子网的0.0.0.0/0路由。

```
aws ec2 delete-client-vpn-route \  
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde \  
  --destination-cidr-block 0.0.0.0/0 \  
  --target-vpc-subnet-id subnet-0123456789abcabca
```

输出：

```
{  
  "Status": {  
    "Code": "deleting"  
  }  
}
```

有关更多信息，请参阅《[AWS Client VPN 管理员指南](#)》中的[路由](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteClientVpnRoute](#)中的。

## delete-coip-cidr

以下代码示例演示了如何使用 delete-coip-cidr。

### AWS CLI

删除一系列客户拥有的 IP (CoIP) 地址

以下delete-coip-cidr示例删除指定 CoIP 池中指定范围的 CoIP 地址。

```
aws ec2 delete-coip-cidr \  
  --cidr 14.0.0.0/24 \  
  --coip-pool-id ipv4pool-coip-1234567890abcdefg
```

输出：

```
{  
  "CoipCidr": {  
    "Cidr": "14.0.0.0/24",  
    "CoipPoolId": "ipv4pool-coip-1234567890abcdefg",  
    "LocalGatewayRouteTableId": "lgw-rtb-abcdefg1234567890"  
  }  
}
```

有关更多信息，请参阅 [《AWS Outposts 用户指南》](#) 中的客户拥有的 IP 地址。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteCoipCidr](#) 中的。

## delete-coip-pool

以下代码示例演示了如何使用 delete-coip-pool。

### AWS CLI

删除客户拥有的 IP (CoIP) 地址池

以下 delete-coip-pool 示例删除了由 CoIP 地址组成的 CoIP 地址池。

```
aws ec2 delete-coip-pool \  
  --coip-pool-id ipv4pool-coip-1234567890abcdefg
```

输出：

```
{  
  "CoipPool": {  
    "PoolId": "ipv4pool-coip-1234567890abcdefg",  
    "LocalGatewayRouteTableId": "lgw-rtb-abcdefg1234567890",  
    "PoolArn": "arn:aws:ec2:us-west-2:123456789012:coip-pool/ipv4pool-  
coip-1234567890abcdefg"  
  }  
}
```

```
}
```

有关更多信息，请参阅 [《AWS Outposts 用户指南》](#) 中的客户拥有的 IP 地址。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCoipPool](#)中的。

## **delete-customer-gateway**

以下代码示例演示了如何使用 delete-customer-gateway。

### AWS CLI

#### 删除客户网关

此示例删除了指定的客户网关。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-customer-gateway --customer-gateway-id cgw-0e11f167
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCustomerGateway](#)中的。

## **delete-dhcp-options**

以下代码示例演示了如何使用 delete-dhcp-options。

### AWS CLI

#### 删除 DHCP 选项集

此示例删除指定的 DHCP 选项集。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-dhcp-options --dhcp-options-id dopt-d9070ebb
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDhcpOptions](#)中的。

## **delete-egress-only-internet-gateway**

以下代码示例演示了如何使用 delete-egress-only-internet-gateway。

## AWS CLI

### 删除仅限出口的互联网网关

此示例删除指定的仅限出口 Internet 网关。

命令:

```
aws ec2 delete-egress-only-internet-gateway --egress-only-internet-gateway-id
eigw-01eadbd45ecd7943f
```

输出:

```
{
  "ReturnCode": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteEgressOnlyInternetGateway](#)中的。

## delete-fleets

以下代码示例演示了如何使用 delete-fleets。

### AWS CLI

示例 1：删除 EC2 队列并终止关联的实例

以下delete-fleets示例删除指定的 EC2 队列并终止相关的按需实例和竞价型实例。

```
aws ec2 delete-fleets \
  --fleet-ids fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE \
  --terminate-instances
```

输出:

```
{
  "SuccessfulFleetDeletions": [
    {
      "CurrentFleetState": "deleted_terminating",
      "PreviousFleetState": "active",
    }
  ]
}
```

```
        "FleetId": "fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE"
      }
    ],
    "UnsuccessfulFleetDeletions": []
  }
```

有关更多信息，请参阅《适用于 Linux 实例的亚马逊弹性计算云用户指南》中的[删除 EC2 队列](#)。

示例 2：在不终止关联实例的情况下删除 EC2 队列

以下delete-fleets示例在不终止关联的按需实例和竞价型实例的情况下删除指定的 EC2 队列。

```
aws ec2 delete-fleets \
  --fleet-ids fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE \
  --no-terminate-instances
```

输出：

```
{
  "SuccessfulFleetDeletions": [
    {
      "CurrentFleetState": "deleted_running",
      "PreviousFleetState": "active",
      "FleetId": "fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE"
    }
  ],
  "UnsuccessfulFleetDeletions": []
}
```

有关更多信息，请参阅《适用于 Linux 实例的亚马逊弹性计算云用户指南》中的[删除 EC2 队列](#)。

• 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFleets](#)中的。

## delete-flow-logs

以下代码示例演示了如何使用 delete-flow-logs。

### AWS CLI

#### 删除流日志

以下delete-flow-logs示例删除了指定的流日志。

```
aws ec2 delete-flow-logs --flow-log-id fl-11223344556677889
```

输出：

```
{
  "Unsuccessful": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFlowLogs](#)中的。

## delete-fpga-image

以下代码示例演示了如何使用 delete-fpga-image。

AWS CLI

删除 Amazon FPGA 镜像

此示例删除指定的 AFI。

命令：

```
aws ec2 delete-fpga-image --fpga-image-id afi-06b12350a123fbabc
```

输出：

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFpgaImage](#)中的。

## delete-instance-connect-endpoint

以下代码示例演示了如何使用 delete-instance-connect-endpoint。

AWS CLI

删除 EC2 实例 Connect 终端节点



以下`delete-instance-connect-endpoint`示例删除指定的 EC2 Instance Connect 终端节点。

```
aws ec2 delete-instance-connect-endpoint \  
  --instance-connect-endpoint-id eice-03f5e49b83924bbc7
```

输出：

```
{  
  "InstanceConnectEndpoint": {  
    "OwnerId": "111111111111",  
    "InstanceConnectEndpointId": "eice-0123456789example",  
    "InstanceConnectEndpointArn": "arn:aws:ec2:us-east-1:111111111111:instance-  
connect-endpoint/eice-0123456789example",  
    "State": "delete-in-progress",  
    "StateMessage": "",  
    "NetworkInterfaceIds": [],  
    "VpcId": "vpc-0123abcd",  
    "AvailabilityZone": "us-east-1d",  
    "CreatedAt": "2023-02-07T12:05:37+00:00",  
    "SubnetId": "subnet-0123abcd"  
  }  
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的移除 EC2 Instance [Connect 终端节点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInstanceConnectEndpoint](#)中的。

## `delete-instance-event-window`

以下代码示例演示了如何使用 `delete-instance-event-window`。

### AWS CLI

示例 1：删除事件窗口

以下`delete-instance-event-window`示例删除了一个事件窗口。

```
aws ec2 delete-instance-event-window \  
  --region us-east-1 \  
  --instance-event-window-id iew-0abcdef1234567890
```

输出：

```
{
  "InstanceEventWindowState": {
    "InstanceEventWindowId": "iew-0abcdef1234567890",
    "State": "deleting"
  }
}
```

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中计划事件部分的[注意事项](#)。

### 示例 2：强制删除事件窗口

如果事件窗口当前与目标相关联，则以下delete-instance-event-window示例强制删除该事件窗口。

```
aws ec2 delete-instance-event-window \
  --region us-east-1 \
  --instance-event-window-id iew-0abcdef1234567890 \
  --force-delete
```

输出：

```
{
  "InstanceEventWindowState": {
    "InstanceEventWindowId": "iew-0abcdef1234567890",
    "State": "deleting"
  }
}
```

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中计划事件部分的[注意事项](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInstanceEventWindow](#)中的。

## delete-internet-gateway

以下代码示例演示了如何使用 delete-internet-gateway。

### AWS CLI

#### 删除互联网网关

以下delete-internet-gateway示例删除了指定的互联网网关。

```
aws ec2 delete-internet-gateway \  
  --internet-gateway-id igw-0d0fb496b3EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon VPC 用户指南》中的[互联网网关](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInternetGateway](#)中的。

## delete-ipam-pool

以下代码示例演示了如何使用 delete-ipam-pool。

### AWS CLI

#### 删除 IPAM 池

在此示例中，您是 IPAM 委托的管理员，想要删除不再需要的 IPAM 池，但该池已为其配置了 CIDR。如果池已配置了 CIDR，则无法将其删除，除非您使用该--cascade选项，因此您将使用--cascade

要完成此请求，请执行以下操作：

你需要可以获得的 IPAM 池 ID [describe-ipam-pools](#)。--region必须是 IPAM 主区域。

以下delete-ipam-pool示例删除了您 AWS 账户中的 IPAM 池。

```
aws ec2 delete-ipam-pool \  
  --ipam-pool-id ipam-pool-050c886a3ca41cd5b \  
  --cascade \  
  --region us-east-1
```

输出：

```
{  
  "IpamPool": {  
    "OwnerId": "320805250157",  
    "IpamPoolId": "ipam-pool-050c886a3ca41cd5b",
```

```
    "IpamPoolArn": "arn:aws:ec2::320805250157:ipam-pool/ipam-
pool-050c886a3ca41cd5b",
    "IpamScopeArn": "arn:aws:ec2::320805250157:ipam-scope/ipam-
scope-0a158dde35c51107b",
    "IpamScopeType": "private",
    "IpamArn": "arn:aws:ec2::320805250157:ipam/ipam-005f921c17ebd5107",
    "IpamRegion": "us-east-1",
    "Locale": "None",
    "PoolDepth": 1,
    "State": "delete-in-progress",
    "Description": "example",
    "AutoImport": false,
    "AddressFamily": "ipv4",
    "AllocationMinNetmaskLength": 0,
    "AllocationMaxNetmaskLength": 32
  }
}
```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的[删除池](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteIpamPool](#)中的。

## delete-ipam-resource-discovery

以下代码示例演示了如何使用 delete-ipam-resource-discovery。

### AWS CLI

#### 删除资源发现

在此示例中，您是 IPAM 委托的管理员，想要删除您在将 IPAM 与组织外部帐户集成的过程中为与其他 IPAM 管理员共享而创建的非默认资源发现。

要完成此请求，请执行以下操作：

--region 必须是您创建资源发现的区域。如果出现以下情况，则无法删除默认资源发现。"IsDefault": true 默认资源发现是在创建 IPAM 的帐户中自动创建的资源发现。要删除默认资源发现，必须删除 IPAM。

以下 delete-ipam-resource-discovery 示例删除了资源发现。

```
aws ec2 delete-ipam-resource-discovery \
```

```
--ipam-resource-discovery-id ipam-res-disco-0e39761475298ee0f \  
--region us-east-1
```

输出：

```
{  
  "IpamResourceDiscovery": {  
    "OwnerId": "149977607591",  
    "IpamResourceDiscoveryId": "ipam-res-disco-0e39761475298ee0f",  
    "IpamResourceDiscoveryArn": "arn:aws:ec2::149977607591:ipam-resource-  
discovery/ipam-res-disco-0e39761475298ee0f",  
    "IpamResourceDiscoveryRegion": "us-east-1",  
    "OperatingRegions": [  
      {  
        "RegionName": "us-east-1"  
      }  
    ],  
    "IsDefault": false,  
    "State": "delete-in-progress"  
  }  
}
```

有关资源发现的更多信息，请参阅 Amazon VPC IPAM 用户指南中的使用[资源发现](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteIpamResourceDiscovery](#)中的。

## delete-ipam-scope

以下代码示例演示了如何使用 delete-ipam-scope。

### AWS CLI

删除 IPAM 作用域

以下delete-ipam-scope示例删除了 IPAM。

```
aws ec2 delete-ipam-scope \  
--ipam-scope-id ipam-scope-01c1ebab2b63bd7e4
```

输出：

```
{
```

```
"IpamScope": {
  "OwnerId": "123456789012",
  "IpamScopeId": "ipam-scope-01c1ebab2b63bd7e4",
  "IpamScopeArn": "arn:aws:ec2::123456789012:ipam-scope/ipam-
scope-01c1ebab2b63bd7e4",
  "IpamArn": "arn:aws:ec2::123456789012:ipam/ipam-08440e7a3acde3908",
  "IpamRegion": "us-east-1",
  "IpamScopeType": "private",
  "IsDefault": false,
  "Description": "Example description",
  "PoolCount": 0,
  "State": "delete-in-progress"
}
}
```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的[删除范围](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteIpamScope](#) 中的。

## delete-ipam

以下代码示例演示了如何使用 delete-ipam。

### AWS CLI

#### 删除 IPAM

以下 delete-ipam 示例删除了 IPAM。

```
aws ec2 delete-ipam \
  --ipam-id ipam-036486dfa6af58ee0
```

输出：

```
{
  "Ipam": {
    "OwnerId": "123456789012",
    "IpamId": "ipam-036486dfa6af58ee0",
    "IpamArn": "arn:aws:ec2::123456789012:ipam/ipam-036486dfa6af58ee0",
    "IpamRegion": "us-east-1",
    "PublicDefaultScopeId": "ipam-scope-071b8042b0195c183",
    "PrivateDefaultScopeId": "ipam-scope-0807405dece705a30",
```

```
    "ScopeCount": 2,
    "OperatingRegions": [
      {
        "RegionName": "us-east-1"
      },
      {
        "RegionName": "us-east-2"
      },
      {
        "RegionName": "us-west-1"
      }
    ],
    "State": "delete-in-progress"
  }
}
```

有关更多信息，请参阅 Amazon V [PC IPAM 用户指南中的删除 IPAM](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteIpam](#) 中的。

## delete-key-pair

以下代码示例演示了如何使用 delete-key-pair。

### AWS CLI

#### 删除密钥对

以下 delete-key-pair 示例删除指定的 key pair。

```
aws ec2 delete-key-pair \
  --key-name my-key-pair
```

输出：

```
{
  "Return": true,
  "KeyId": "key-03c8d3aceb53b507"
}
```

有关更多信息，请参阅《AWS 命令行界面用户指南》中的 [创建和删除密钥对](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteKeyPair](#)中的。

## delete-launch-template-versions

以下代码示例演示了如何使用 delete-launch-template-versions。

### AWS CLI

删除启动模板版本

此示例删除了指定的启动模板版本。

命令:

```
aws ec2 delete-launch-template-versions --launch-template-id lt-0abcd290751193123 --versions 1
```

输出:

```
{
  "UnsuccessfullyDeletedLaunchTemplateVersions": [],
  "SuccessfullyDeletedLaunchTemplateVersions": [
    {
      "LaunchTemplateName": "TestVersion",
      "VersionNumber": 1,
      "LaunchTemplateId": "lt-0abcd290751193123"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLaunchTemplateVersions](#)中的。

## delete-launch-template

以下代码示例演示了如何使用 delete-launch-template。

### AWS CLI

删除启动模板

本示例将删除指定的启动模板。



**命令:**

```
aws ec2 delete-launch-template --launch-template-id lt-0abcd290751193123
```

**输出:**

```
{
  "LaunchTemplate": {
    "LatestVersionNumber": 2,
    "LaunchTemplateId": "lt-0abcd290751193123",
    "LaunchTemplateName": "TestTemplate",
    "DefaultVersionNumber": 2,
    "CreatedBy": "arn:aws:iam::123456789012:root",
    "CreateTime": "2017-11-23T16:46:25.000Z"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLaunchTemplate](#)中的。

**delete-local-gateway-route-table-virtual-interface-group-association**

以下代码示例演示了如何使用 `delete-local-gateway-route-table-virtual-interface-group-association`。

**AWS CLI**

取消本地网关路由表与虚拟接口 (VIF) 组的关联

以下`delete-local-gateway-route-table-virtual-interface-group-association`示例删除了指定的本地网关路由表和 VIF 组之间的关联。

```
aws ec2 delete-local-gateway-route-table-virtual-interface-group-association \
  --local-gateway-route-table-virtual-interface-group-association-id lgw-vif-grp-
  assoc-exampleid12345678
```

**输出:**

```
{
  "LocalGatewayRouteTableVirtualInterfaceGroupAssociation": {
    "LocalGatewayRouteTableVirtualInterfaceGroupAssociationId": "lgw-vif-grp-
    assoc-exampleid12345678",
  }
}
```

```

    "LocalGatewayVirtualInterfaceGroupId": "lgw-vif-grp-exampleid0123abcd",
    "LocalGatewayId": "lgw-exampleid11223344",
    "LocalGatewayRouteTableId": "lgw-rtb-exampleidabcd1234",
    "LocalGatewayRouteTableArn": "arn:aws:ec2:us-west-2:111122223333:local-
gateway-route-table/lgw-rtb-exampleidabcd1234",
    "OwnerId": "111122223333",
    "State": "disassociating",
    "Tags": []
  }
}

```

有关更多信息，请参阅 [O utpo AWS st s 用户指南中的 VIF 群组关联](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 DeleteLocalGatewayRouteTableVirtualInterfaceGroupAssociation](#) 中的。

## delete-local-gateway-route-table-vpc-association

以下代码示例演示了如何使用 `delete-local-gateway-route-table-vpc-association`。

### AWS CLI

取消本地网关路由表与 VPC 的关联

以下 `delete-local-gateway-route-table-vpc-association` 示例删除了指定的本地网关路由表和 VPC 之间的关联。

```

aws ec2 delete-local-gateway-route-table-vpc-association \
  --local-gateway-route-table-vpc-association-id vpc-example0123456789

```

输出：

```

{
  "LocalGatewayRouteTableVpcAssociation": {
    "LocalGatewayRouteTableVpcAssociationId": "lgw-vpc-assoc-abcd1234wxyz56789",
    "LocalGatewayRouteTableId": "lgw-rtb-abcdefg1234567890",
    "LocalGatewayRouteTableArn": "arn:aws:ec2:us-west-2:555555555555:local-
gateway-route-table/lgw-rtb-abcdefg1234567890",
    "LocalGatewayId": "lgw-exampleid01234567",
    "VpcId": "vpc-example0123456789",
    "OwnerId": "555555555555",
    "State": "disassociating"
  }
}

```

```
}  
}
```

有关更多信息，请参阅 AWS Outposts 用户指南中的 [VPC 关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteLocalGatewayRouteTableVpcAssociation](#) 中的。

## delete-local-gateway-route-table

以下代码示例演示了如何使用 delete-local-gateway-route-table。

### AWS CLI

删除本地网关路由表

以下 delete-local-gateway-route-table 示例使用直接 VPC 路由模式创建本地网关路由表。

```
aws ec2 delete-local-gateway-route-table \  
  --local-gateway-route-table-id lgw-rtb-abcdefg1234567890
```

输出：

```
{  
  "LocalGatewayRouteTable": {  
    "LocalGatewayRouteTableId": "lgw-rtb-abcdefg1234567890",  
    "LocalGatewayRouteTableArn": "arn:aws:ec2:us-west-2:111122223333:local-gateway-route-table/lgw-rtb-abcdefg1234567890",  
    "LocalGatewayId": "lgw-1a2b3c4d5e6f7g8h9",  
    "OutpostArn": "arn:aws:outposts:us-west-2:111122223333:outpost/op-021345abcdef67890",  
    "OwnerId": "111122223333",  
    "State": "deleting",  
    "Tags": [],  
    "Mode": "direct-vpc-routing"  
  }  
}
```

有关更多信息，请参阅《AWS Outposts 用户指南》中的 [本地网关路由表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteLocalGatewayRouteTable](#) 中的。

## delete-local-gateway-route

以下代码示例演示了如何使用 delete-local-gateway-route。

### AWS CLI

从本地网关路由表中删除路由

以下delete-local-gateway-route示例从指定的本地网关路由表中删除指定的路由。

```
aws ec2 delete-local-gateway-route \
  --destination-cidr-block 0.0.0.0/0 \
  --local-gateway-route-table-id lgw-rtb-059615ef7dEXAMPLE
```

输出：

```
{
  "Route": {
    "DestinationCidrBlock": "0.0.0.0/0",
    "LocalGatewayVirtualInterfaceGroupId": "lgw-vif-grp-07145b276bEXAMPLE",
    "Type": "static",
    "State": "deleted",
    "LocalGatewayRouteTableId": "lgw-rtb-059615ef7EXAMPLE"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLocalGatewayRoute](#)中的。

## delete-managed-prefix-list

以下代码示例演示了如何使用 delete-managed-prefix-list。

### AWS CLI

删除前缀列表

以下delete-managed-prefix-list示例删除了指定的前缀列表。

```
aws ec2 delete-managed-prefix-list \
  --prefix-list-id pl-0123456abcabc1
```

输出：

```
{
  "PrefixList": {
    "PrefixListId": "pl-0123456abcabc1",
    "AddressFamily": "IPv4",
    "State": "delete-in-progress",
    "PrefixListArn": "arn:aws:ec2:us-west-2:123456789012:prefix-list/pl-0123456abcabc1",
    "PrefixListName": "test",
    "MaxEntries": 10,
    "Version": 1,
    "OwnerId": "123456789012"
  }
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的[托管前缀列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteManagedPrefixList](#)中的。

## delete-nat-gateway

以下代码示例演示了如何使用 delete-nat-gateway。

### AWS CLI

#### 删除 NAT 网关

此示例删除 NAT 网关nat-04ae55e711cec5680。

命令：

```
aws ec2 delete-nat-gateway --nat-gateway-id nat-04ae55e711cec5680
```

输出：

```
{
  "NatGatewayId": "nat-04ae55e711cec5680"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNatGateway](#)中的。

## delete-network-acl-entry

以下代码示例演示了如何使用 `delete-network-acl-entry`。

### AWS CLI

#### 删除网络 ACL 条目

此示例从指定的网络 ACL 中删除了编号为 100 的入口规则。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-number 100
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNetworkAclEntry](#)中的。

## delete-network-acl

以下代码示例演示了如何使用 `delete-network-acl`。

### AWS CLI

#### 删除网络 ACL

此示例删除指定的网络 ACL。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-network-acl --network-acl-id acl-5fb85d36
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNetworkAcl](#)中的。

## delete-network-insights-access-scope-analysis

以下代码示例演示了如何使用 `delete-network-insights-access-scope-analysis`。

### AWS CLI

#### 删除网络访问范围分析

以下`delete-network-insights-access-scope-analysis`示例删除了指定的网络访问范围分析。

```
aws ec2 delete-network-insights-access-scope-analysis \  
--network-insights-access-scope-analysis-id nisa-01234567891abcdef
```

输出：

```
{  
  "NetworkInsightsAccessScopeAnalysisId": "nisa-01234567891abcdef"  
}
```

有关更多信息，请参阅《[网络访问分析器指南](#)》中的“[使用 AWS CLI 开始使用网络访问分析器](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNetworkInsightsAccessScopeAnalysis](#)中的。

## delete-network-insights-access-scope

以下代码示例演示了如何使用 delete-network-insights-access-scope。

AWS CLI

删除网络访问范围

以下delete-network-insights-access-scope示例删除了指定的网络访问范围。

```
aws ec2 delete-network-insights-access-scope \  
--network-insights-access-scope-id nis-123456789abc01234
```

输出：

```
{  
  "NetworkInsightsAccessScopeId": "nis-123456789abc01234"  
}
```

有关更多信息，请参阅《[网络访问分析器指南](#)》中的“[使用 AWS CLI 开始使用网络访问分析器](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNetworkInsightsAccessScope](#)中的。

## delete-network-insights-analysis

以下代码示例演示了如何使用 delete-network-insights-analysis。

## AWS CLI

### 删除路径分析

以下delete-network-insights-analysis示例删除了指定的分析。

```
aws ec2 delete-network-insights-analysis \  
  --network-insights-analysis-id nia-02207aa13eb480c7a
```

输出：

```
{  
  "NetworkInsightsAnalysisId": "nia-02207aa13eb480c7a"  
}
```

有关更多信息，请参阅 [《Reach ability Analyzer 指南》中的 AWS CLI 使用入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNetworkInsightsAnalysis](#)中的。

## delete-network-insights-path

以下代码示例演示了如何使用 delete-network-insights-path。

### AWS CLI

#### 删除路径

以下delete-network-insights-path示例删除了指定的路径。在删除路径之前，必须使用delete-network-insights-analysis命令删除其所有分析。

```
aws ec2 delete-network-insights-path \  
  --network-insights-path-id nip-0b26f224f1d131fa8
```

输出：

```
{  
  "NetworkInsightsPathId": "nip-0b26f224f1d131fa8"  
}
```

有关更多信息，请参阅 [《Reach ability Analyzer 指南》中的 AWS CLI 使用入门](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNetworkInsightsPath](#)中的。

## delete-network-interface-permission

以下代码示例演示了如何使用 delete-network-interface-permission。

### AWS CLI

#### 删除网络接口权限

此示例删除了指定的网络接口权限。

命令:

```
aws ec2 delete-network-interface-permission --network-interface-permission-id eni-perm-06fd19020ede149ea
```

输出:

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNetworkInterfacePermission](#)中的。

## delete-network-interface

以下代码示例演示了如何使用 delete-network-interface。

### AWS CLI

#### 删除网络接口

此示例删除了指定的网络接口。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-network-interface --network-interface-id eni-e5aa89a3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNetworkInterface](#)中的。

## delete-placement-group

以下代码示例演示了如何使用 delete-placement-group。

### AWS CLI

#### 删除置放群组

此示例命令删除指定的置放群组。

命令:

```
aws ec2 delete-placement-group --group-name my-cluster
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePlacementGroup](#)中的。

## delete-queued-reserved-instances

以下代码示例演示了如何使用 delete-queued-reserved-instances。

### AWS CLI

#### 删除已排队的购买

以下delete-queued-reserved-instances示例删除了排队等候购买的指定预留实例。

```
aws ec2 delete-queued-reserved-instances \  
  --reserved-instances-ids af9f760e-6f91-4559-85f7-4980eexample
```

输出:

```
{  
  "SuccessfulQueuedPurchaseDeletions": [  
    {  
      "ReservedInstancesId": "af9f760e-6f91-4559-85f7-4980eexample"  
    }  
  ],  
  "FailedQueuedPurchaseDeletions": []  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteQueuedReservedInstances](#)中的。

## delete-route-table

以下代码示例演示了如何使用 delete-route-table。

### AWS CLI

#### 删除路由表

此示例删除了指定的路由表。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-route-table --route-table-id rtb-22574640
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRouteTable](#)中的。

## delete-route

以下代码示例演示了如何使用 delete-route。

### AWS CLI

#### 删除路线

此示例从指定路由表中删除指定路由。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-route --route-table-id rtb-22574640 --destination-cidr-block  
0.0.0.0/0
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRoute](#)中的。

## delete-security-group

以下代码示例演示了如何使用 delete-security-group。

### AWS CLI

#### [EC2-Classical] 删除安全组

本示例将删除名为 MySecurityGroup 的安全组。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-security-group --group-name MySecurityGroup
```

### [EC2-VPC] 删除安全组

本示例将删除 ID 为 `sg-903004f8` 的安全组。请注意，您不能通过名称引用 EC2-VPC 的安全组。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-security-group --group-id sg-903004f8
```

有关更多信息，请参阅《AWS 命令行界面用户指南》中的“使用安全组”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSecurityGroup](#)中的。

## delete-snapshot

以下代码示例演示了如何使用 `delete-snapshot`。

### AWS CLI

#### 删除快照

本示例命令将删除快照 ID 为 `snap-1234567890abcdef0` 的快照。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-snapshot --snapshot-id snap-1234567890abcdef0
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSnapshot](#)中的。

## delete-spot-datafeed-subscription

以下代码示例演示了如何使用 `delete-spot-datafeed-subscription`。

### AWS CLI

#### 取消竞价型实例数据源订阅

此示例命令删除该账户的竞价数据源订阅。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-spot-datafeed-subscription
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSpotDatafeedSubscription](#)中的。

## delete-subnet-cidr-reservation

以下代码示例演示了如何使用 delete-subnet-cidr-reservation。

AWS CLI

删除子网 CIDR 预留

以下delete-subnet-cidr-reservation示例删除了指定的子网 CIDR 预留。

```
aws ec2 delete-subnet-cidr-reservation \  
  --subnet-cidr-reservation-id scr-044f977c4eEXAMPLE
```

输出:

```
{  
  "DeletedSubnetCidrReservation": {  
    "SubnetCidrReservationId": "scr-044f977c4eEXAMPLE",  
    "SubnetId": "subnet-03c51e2e6cEXAMPLE",  
    "Cidr": "10.1.0.16/28",  
    "ReservationType": "prefix",  
    "OwnerId": "123456789012"  
  }  
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的[子网 CIDR 预留](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSubnetCidrReservation](#)中的。

## delete-subnet

以下代码示例演示了如何使用 delete-subnet。

## AWS CLI

### 删除子网

此示例删除了指定的子网。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-subnet --subnet-id subnet-9d4a7b6c
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSubnet](#)中的。

### delete-tags

以下代码示例演示了如何使用 delete-tags。

## AWS CLI

### 示例 1：从资源中删除标签

以下delete-tags示例Stack=Test从指定图像中删除标签。当您同时指定值和密钥名称时，只有当标签的值与指定值匹配时，才会删除该标签。

```
aws ec2 delete-tags \  
  --resources ami-1234567890abcdef0 \  
  --tags Key=Stack,Value=Test
```

为标签指定值是可选的。以下delete-tags示例将purpose从指定实例中删除带有密钥名称的标签，而不管该标签的标签值如何。

```
aws ec2 delete-tags \  
  --resources i-1234567890abcdef0 \  
  --tags Key=purpose
```

如果将空字符串指定为标签值，则仅当标签的值为空字符串时，才会删除标记。以下delete-tags示例将空字符串指定为要删除的标签的标签值。

```
aws ec2 delete-tags \  
  --resources i-1234567890abcdef0 \  
  --tags Key=
```

```
--tags Key=Name,Value=
```

## 示例 2：从多个资源中删除标签

以下delete-tags示例从实例和 AMI 中删除标签“purpose=test”。如前面的示例所示，您可以省略命令中的标签值。

```
aws ec2 delete-tags \  
  --resources i-1234567890abcdef0 ami-1234567890abcdef0 \  
  --tags Key=Purpose
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTags](#)中的。

## delete-traffic-mirror-filter-rule

以下代码示例演示了如何使用 delete-traffic-mirror-filter-rule。

### AWS CLI

#### 删除流量镜像过滤器规则

以下delete-traffic-mirror-filter-rule示例删除了指定的流量镜像过滤规则。

```
aws ec2 delete-traffic-mirror-filter-rule \  
  --traffic-mirror-filter-rule-id tmfr-081f71283bEXAMPLE
```

输出：

```
{  
  "TrafficMirrorFilterRuleId": "tmfr-081f71283bEXAMPLE"  
}
```

有关更多信息，请参阅《[流量镜像指南](#)》中的[修改您的AWS 流量镜像过滤器规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTrafficMirrorFilterRule](#)中的。

## delete-traffic-mirror-filter

以下代码示例演示了如何使用 delete-traffic-mirror-filter。

## AWS CLI

### 删除流量镜像过滤器

以下delete-traffic-mirror-filter示例删除了指定的流量镜像过滤器。

```
aws ec2 delete-traffic-mirror-filter \  
  --traffic-mirror-filter-id tmf-0be0b25fcdEXAMPLE
```

输出：

```
{  
  "TrafficMirrorFilterId": "tmf-0be0b25fcdEXAMPLE"  
}
```

有关更多信息，请参阅《[流量镜像指南](#)》中的[删除AWS 流量镜像过滤器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTrafficMirrorFilter](#)中的。

## delete-traffic-mirror-session

以下代码示例演示了如何使用 delete-traffic-mirror-session。

## AWS CLI

### 删除流量镜像会话

以下delete-traffic-mirror-session示例删除了指定的流量镜像会话。

```
aws ec2 delete-traffic-mirror-session \  
  --traffic-mirror-session-id tms-0af3141ce5EXAMPLE
```

输出：

```
{  
  "TrafficMirrorSessionId": "tms-0af3141ce5EXAMPLE"  
}
```

有关更多信息，请参阅《[流量镜像指南](#)》中的[删除AWS 流量镜像会话](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTrafficMirrorSession](#)中的。



## delete-traffic-mirror-target

以下代码示例演示了如何使用 delete-traffic-mirror-target。

### AWS CLI

#### 删除流量镜像目标

以下delete-traffic-mirror-target示例删除了指定的流量镜像目标。

```
aws ec2 delete-traffic-mirror-target \  
  --traffic-mirror-target-id tmt-060f48ce9EXAMPLE
```

输出：

```
{  
  "TrafficMirrorTargetId": "tmt-060f48ce9EXAMPLE"  
}
```

有关更多信息，请参阅 [《流量镜像指南》中的删除AWS 流量镜像目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTrafficMirrorTarget](#)中的。

## delete-transit-gateway-connect-peer

以下代码示例演示了如何使用 delete-transit-gateway-connect-peer。

### AWS CLI

#### 删除 Transit Gateway Connect 对等体

以下delete-transit-gateway-connect-peer示例删除指定的 Connect 对等体。

```
aws ec2 delete-transit-gateway-connect-peer \  
  --transit-gateway-connect-peer-id tgw-connect-peer-0666adbac4EXAMPLE
```

输出：

```
{  
  "TransitGatewayConnectPeer": {  
    "TransitGatewayAttachmentId": "tgw-attach-0f0927767cEXAMPLE",  
    "TransitGatewayConnectPeerId": "tgw-connect-peer-0666adbac4EXAMPLE",
```

```

    "State": "deleting",
    "CreationTime": "2021-10-13T03:35:17.000Z",
    "ConnectPeerConfiguration": {
      "TransitGatewayAddress": "10.0.0.234",
      "PeerAddress": "172.31.1.11",
      "InsideCidrBlocks": [
        "169.254.6.0/29"
      ],
      "Protocol": "gre",
      "BgpConfigurations": [
        {
          "TransitGatewayAsn": 64512,
          "PeerAsn": 64512,
          "TransitGatewayAddress": "169.254.6.2",
          "PeerAddress": "169.254.6.1",
          "BgpStatus": "down"
        },
        {
          "TransitGatewayAsn": 64512,
          "PeerAsn": 64512,
          "TransitGatewayAddress": "169.254.6.3",
          "PeerAddress": "169.254.6.1",
          "BgpStatus": "down"
        }
      ]
    }
  }
}

```

有关更多信息，请参阅《公交网关指南》中的[公交网关 Connect 附件和 Transit Gateway Connect 对等体](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTransitGatewayConnectPeer](#)中的。

## delete-transit-gateway-connect

以下代码示例演示了如何使用 delete-transit-gateway-connect。

### AWS CLI

要删除公交网关 Connect 附件

以下delete-transit-gateway-connect示例删除指定的 Connect 附件。

```
aws ec2 delete-transit-gateway-connect \  
  --transit-gateway-attachment-id tgw-attach-037012e5dcEXAMPLE
```

输出：

```
{  
  "TransitGatewayConnect": {  
    "TransitGatewayAttachmentId": "tgw-attach-037012e5dcEXAMPLE",  
    "TransportTransitGatewayAttachmentId": "tgw-attach-0a89069f57EXAMPLE",  
    "TransitGatewayId": "tgw-02f776b1a7EXAMPLE",  
    "State": "deleting",  
    "CreationTime": "2021-03-09T19:59:17+00:00",  
    "Options": {  
      "Protocol": "gre"  
    }  
  }  
}
```

有关更多信息，请参阅《公交网关指南》中的[公交网关 Connect 附件和 Transit Gateway Connect 对等体](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTransitGatewayConnect](#)中的。

## delete-transit-gateway-multicast-domain

以下代码示例演示了如何使用 delete-transit-gateway-multicast-domain。

AWS CLI

删除传输网关组播域

以下delete-transit-gateway-multicast-domain示例删除了指定的多播域。

```
aws ec2 delete-transit-gateway-multicast-domain \  
  --transit-gateway-multicast-domain-id tgw-mcast-domain-0c4905cef7EXAMPLE
```

输出：

```
{  
  "TransitGatewayMulticastDomain": {  
    "TransitGatewayMulticastDomainId": "tgw-mcast-domain-02bb79002bEXAMPLE",
```

```
    "TransitGatewayId": "tgw-0d88d2d0d5EXAMPLE",
    "State": "deleting",
    "CreationTime": "2019-11-20T22:02:03.000Z"
  }
}
```

有关更多信息，请参阅《传输网关指南》中的[管理多播域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTransitGatewayMulticastDomain](#)中的。

## delete-transit-gateway-peering-attachment

以下代码示例演示了如何使用 delete-transit-gateway-peering-attachment。

### AWS CLI

#### 删除传输网关对等连接

以下delete-transit-gateway-peering-attachment示例删除了指定的公交网关对等连接。

```
aws ec2 delete-transit-gateway-peering-attachment \
  --transit-gateway-attachment-id tgw-attach-4455667788aabbccd
```

输出：

```
{
  "TransitGatewayPeeringAttachment": {
    "TransitGatewayAttachmentId": "tgw-attach-4455667788aabbccd",
    "RequesterTgwInfo": {
      "TransitGatewayId": "tgw-123abc05e04123abc",
      "OwnerId": "123456789012",
      "Region": "us-west-2"
    },
    "AcceptorTgwInfo": {
      "TransitGatewayId": "tgw-11223344aabbcc112",
      "OwnerId": "123456789012",
      "Region": "us-east-2"
    },
    "State": "deleting",
    "CreationTime": "2019-12-09T11:38:31.000Z"
  }
}
```

```
}
```

有关更多信息，请参阅《[公交网关指南](#)》中的 [Transit Gateway 对等连接附件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTransitGatewayPeeringAttachment](#)中的。

## delete-transit-gateway-policy-table

以下代码示例演示了如何使用 delete-transit-gateway-policy-table。

### AWS CLI

删除传输网关策略表

以下delete-transit-gateway-policy-table示例删除了指定的传输网关策略表。

```
aws ec2 delete-transit-gateway-policy-table \
  --transit-gateway-policy-table-id tgw-ptb-0a16f134b78668a81
```

输出：

```
{
  "TransitGatewayPolicyTables": [
    {
      "TransitGatewayPolicyTableId": "tgw-ptb-0a16f134b78668a81",
      "TransitGatewayId": "tgw-067f8505c18f0bd6e",
      "State": "deleting",
      "CreationTime": "2023-11-28T16:36:43+00:00",
      "Tags": []
    }
  ]
}
```

有关更多信息，请参阅 [Transit Gateway 用户指南中的公交网关策略表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTransitGatewayPolicyTable](#)中的。

## delete-transit-gateway-prefix-list-reference

以下代码示例演示了如何使用 delete-transit-gateway-prefix-list-reference。

## AWS CLI

### 删除前缀列表引用

以下delete-transit-gateway-prefix-list-reference示例删除了指定的前缀列表引用。

```
aws ec2 delete-transit-gateway-prefix-list-reference \  
  --transit-gateway-route-table-id tgw-rtb-0123456789abcd123 \  
  --prefix-list-id pl-11111122222222333
```

输出：

```
{  
  "TransitGatewayPrefixListReference": {  
    "TransitGatewayRouteTableId": "tgw-rtb-0123456789abcd123",  
    "PrefixListId": "pl-11111122222222333",  
    "PrefixListOwnerId": "123456789012",  
    "State": "deleting",  
    "Blackhole": false,  
    "TransitGatewayAttachment": {  
      "TransitGatewayAttachmentId": "tgw-attach-aabbccddaabbccaab",  
      "ResourceType": "vpc",  
      "ResourceId": "vpc-112233445566aabbcc"  
    }  
  }  
}
```

有关更多信息，请参阅 *T ransit Gateways 指南* 中的[前缀列表参考](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTransitGatewayPrefixListReference](#)中的。

## delete-transit-gateway-route-table

以下代码示例演示了如何使用 delete-transit-gateway-route-table。

### AWS CLI

#### 删除公网网关路由表

以下delete-transit-gateway-route-table示例删除了指定的公网网关路由表。

```
aws ec2 delete-transit-gateway-route-table \  
  --transit-gateway-route-table-id tgw-rtb-0123456789abcd123
```

```
--transit-gateway-route-table-id tgw-rtb-0b6f6aaa01EXAMPLE
```

输出：

```
{
  "TransitGatewayRouteTable": {
    "TransitGatewayRouteTableId": "tgw-rtb-0b6f6aaa01EXAMPLE",
    "TransitGatewayId": "tgw-02f776b1a7EXAMPLE",
    "State": "deleting",
    "DefaultAssociationRouteTable": false,
    "DefaultPropagationRouteTable": false,
    "CreationTime": "2019-07-17T20:27:26.000Z"
  }
}
```

有关更多信息，请参阅 [Transit Gateways 指南中的删除中转网关路由表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [Delete Transit Gateway Route Table](#) 中的。

## delete-transit-gateway-route

以下代码示例演示了如何使用 delete-transit-gateway-route。

### AWS CLI

从路由表中删除 CIDR 块

以下 delete-transit-gateway-route 示例从指定的中转网关路由表中删除 CIDR 块。

```
aws ec2 delete-transit-gateway-route \
  --transit-gateway-route-table-id tgw-rtb-0b6f6aaa01EXAMPLE \
  --destination-cidr-block 10.0.2.0/24
```

输出：

```
{
  "Route": {
    "DestinationCidrBlock": "10.0.2.0/24",
    "TransitGatewayAttachments": [
      {
        "ResourceId": "vpc-0065acced4EXAMPLE",
```

```
        "TransitGatewayAttachmentId": "tgw-attach-0b5968d3b6EXAMPLE",
        "ResourceType": "vpc"
    }
],
    "Type": "static",
    "State": "deleted"
}
}
```

有关更多信息，请参阅 [Transit Gateways 指南中的删除静态路由](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [Delete Transit Gateway Route](#) 中的。

## delete-transit-gateway-vpc-attachment

以下代码示例演示了如何使用 delete-transit-gateway-vpc-attachment。

### AWS CLI

#### 删除传输网关 VPC 附件

以下 delete-transit-gateway-vpc-attachment 示例删除指定的 VPC 附件。

```
aws ec2 delete-transit-gateway-vpc-attachment \
    --transit-gateway-attachment-id tgw-attach-0d2c54bdbEXAMPLE
```

输出：

```
{
  "TransitGatewayVpcAttachment": {
    "TransitGatewayAttachmentId": "tgw-attach-0d2c54bdb3EXAMPLE",
    "TransitGatewayId": "tgw-02f776b1a7EXAMPLE",
    "VpcId": "vpc-0065acced4f61c651",
    "VpcOwnerId": "111122223333",
    "State": "deleting",
    "CreationTime": "2019-07-17T16:04:27.000Z"
  }
}
```

有关更多信息，请参阅《传输网关指南》中的 [删除 VPC 附件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [Delete Transit Gateway Vpc Attachment](#) 中的。



## delete-transit-gateway

以下代码示例演示了如何使用 delete-transit-gateway。

### AWS CLI

#### 删除传输网关

以下delete-transit-gateway示例删除了指定的传输网关。

```
aws ec2 delete-transit-gateway \  
  --transit-gateway-id tgw-01f04542b2EXAMPLE
```

输出：

```
{  
  "TransitGateway": {  
    "TransitGatewayId": "tgw-01f04542b2EXAMPLE",  
    "State": "deleting",  
    "OwnerId": "123456789012",  
    "Description": "Example Transit Gateway",  
    "CreationTime": "2019-08-27T15:04:35.000Z",  
    "Options": {  
      "AmazonSideAsn": 64515,  
      "AutoAcceptSharedAttachments": "disable",  
      "DefaultRouteTableAssociation": "enable",  
      "AssociationDefaultRouteTableId": "tgw-rtb-0ce7a6948fEXAMPLE",  
      "DefaultRouteTablePropagation": "enable",  
      "PropagationDefaultRouteTableId": "tgw-rtb-0ce7a6948fEXAMPLE",  
      "VpnEcmpSupport": "enable",  
      "DnsSupport": "enable"  
    }  
  }  
}
```

有关更多信息，请参阅《[中转网关指南](#)》中的[删除](#)中转网关。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTransitGateway](#)中的。

## delete-verified-access-endpoint

以下代码示例演示了如何使用 delete-verified-access-endpoint。

## AWS CLI

### 删除已验证访问终端节点

以下delete-verified-access-endpoint示例删除了指定的已验证访问终端节点。

```
aws ec2 delete-verified-access-endpoint \  
  --verified-access-endpoint-id vae-066fac616d4d546f2
```

输出：

```
{  
  "VerifiedAccessEndpoint": {  
    "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",  
    "VerifiedAccessGroupId": "vagr-0dbe967baf14b7235",  
    "VerifiedAccessEndpointId": "vae-066fac616d4d546f2",  
    "ApplicationDomain": "example.com",  
    "EndpointType": "network-interface",  
    "AttachmentType": "vpc",  
    "DomainCertificateArn": "arn:aws:acm:us-east-2:123456789012:certificate/  
eb065ea0-26f9-4e75-a6ce-0a1a7EXAMPLE",  
    "EndpointDomain": "my-ava-  
app.edge-00c3372d53b1540bb.vai-0ce000c0b7643abea.prod.verified-access.us-  
east-2.amazonaws.com",  
    "SecurityGroupIds": [  
      "sg-004915970c4c8f13a"  
    ],  
    "NetworkInterfaceOptions": {  
      "NetworkInterfaceId": "eni-0aec70418c8d87a0f",  
      "Protocol": "https",  
      "Port": 443  
    },  
    "Status": {  
      "Code": "deleting"  
    },  
    "Description": "Testing Verified Access",  
    "CreationTime": "2023-08-25T20:54:43",  
    "LastUpdatedTime": "2023-08-25T22:46:32"  
  }  
}
```

有关更多信息，请参阅 [《已验证访问用户指南》](#) 中的AWS 已验证访问终端节点。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVerifiedAccessEndpoint](#)中的。

## delete-verified-access-group

以下代码示例演示了如何使用 delete-verified-access-group。

### AWS CLI

#### 删除已验证访问权限组

以下delete-verified-access-group示例删除了指定的已验证访问权限组。

```
aws ec2 delete-verified-access-group \  
  --verified-access-group-id vagr-0dbe967baf14b7235
```

输出：

```
{  
  "VerifiedAccessGroup": {  
    "VerifiedAccessGroupId": "vagr-0dbe967baf14b7235",  
    "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",  
    "Description": "Testing Verified Access",  
    "Owner": "123456789012",  
    "VerifiedAccessGroupArn": "arn:aws:ec2:us-east-2:123456789012:verified-  
access-group/vagr-0dbe967baf14b7235",  
    "CreationTime": "2023-08-25T19:55:19",  
    "LastUpdatedTime": "2023-08-25T22:49:03",  
    "DeletionTime": "2023-08-26T00:58:31"  
  }  
}
```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVerifiedAccessGroup](#)中的。

## delete-verified-access-instance

以下代码示例演示了如何使用 delete-verified-access-instance。

### AWS CLI

#### 删除已验证访问权限实例

以下delete-verified-access-instance示例删除指定的“已验证访问权限”实例。

```
aws ec2 delete-verified-access-instance \  
  --verified-access-instance-id vai-0ce000c0b7643abea
```

输出：

```
{  
  "VerifiedAccessInstance": {  
    "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",  
    "Description": "Testing Verified Access",  
    "VerifiedAccessTrustProviders": [],  
    "CreationTime": "2023-08-25T18:27:56",  
    "LastUpdatedTime": "2023-08-26T01:00:18"  
  }  
}
```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVerifiedAccessInstance](#)中的。

## delete-verified-access-trust-provider

以下代码示例演示了如何使用 delete-verified-access-trust-provider。

### AWS CLI

删除已验证访问信任提供商

以下delete-verified-access-trust-provider示例删除了指定的已验证访问信任提供商。

```
aws ec2 delete-verified-access-trust-provider \  
  --verified-access-trust-provider-id vatp-0bb32de759a3e19e7
```

输出：

```
{  
  "VerifiedAccessTrustProvider": {  
    "VerifiedAccessTrustProviderId": "vatp-0bb32de759a3e19e7",  
    "Description": "Testing Verified Access",  
    "TrustProviderType": "user",  
    "UserTrustProviderType": "iam-identity-center",  
  }  
}
```

```
    "PolicyReferenceName": "idc",
    "CreationTime": "2023-08-25T18:40:36",
    "LastUpdatedTime": "2023-08-25T18:40:36"
  }
}
```

有关更多信息，请参阅《已验证访问用户指南》中的“AWS 验证访问权限的[信任提供商](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVerifiedAccessTrustProvider](#)中的。

## delete-volume

以下代码示例演示了如何使用 delete-volume。

### AWS CLI

#### 删除卷

此示例命令删除卷 ID 为的可用卷vol-049df61146c4d7901。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-volume --volume-id vol-049df61146c4d7901
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVolume](#)中的。

## delete-vpc-endpoint-connection-notifications

以下代码示例演示了如何使用 delete-vpc-endpoint-connection-notifications。

### AWS CLI

#### 删除端点连接通知

此示例删除了指定的端点连接通知。

命令:

```
aws ec2 delete-vpc-endpoint-connection-notifications --connection-notification-ids
vpce-nfn-008776de7e03f5abc
```

输出：

```
{
  "Unsuccessful": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVpcEndpointConnectionNotifications](#)中的。

## delete-vpc-endpoint-service-configurations

以下代码示例演示了如何使用 delete-vpc-endpoint-service-configurations。

AWS CLI

删除终端节点服务配置

此示例删除了指定的终端节点服务配置。

命令：

```
aws ec2 delete-vpc-endpoint-service-configurations --service-ids vpce-
svc-03d5ebb7d9579a2b3
```

输出：

```
{
  "Unsuccessful": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVpcEndpointServiceConfigurations](#)中的。

## delete-vpc-endpoints

以下代码示例演示了如何使用 delete-vpc-endpoints。

AWS CLI

删除终端节点

此示例删除了端点 `vpce-aa22bb33` 和 `vpce-1a2b3c4d`。如果命令部分成功或不成功，则返回失败项目的列表。如果命令成功，则返回的列表为空。

命令:

```
aws ec2 delete-vpc-endpoints --vpc-endpoint-ids vpce-aa22bb33 vpce-1a2b3c4d
```

输出:

```
{
  "Unsuccessful": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVpcEndpoints](#)中的。

## delete-vpc-peering-connection

以下代码示例演示了如何使用 `delete-vpc-peering-connection`。

AWS CLI

删除 VPC 对等连接

此示例删除指定的 VPC 对等连接。

命令:

```
aws ec2 delete-vpc-peering-connection --vpc-peering-connection-id pcx-1a2b3c4d
```

输出:

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVpcPeeringConnection](#)中的。

## delete-vpc

以下代码示例演示了如何使用 `delete-vpc`。

## AWS CLI

### 删除 VPC

此示例删除指定的 VPC。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-vpc --vpc-id vpc-a01106c2
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVpc](#)中的。

## delete-vpn-connection-route

以下代码示例演示了如何使用 delete-vpn-connection-route。

### AWS CLI

#### 从 VPN 连接中删除静态路由

此示例从指定的 VPN 连接中删除指定的静态路由。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-vpn-connection-route --vpn-connection-id vpn-40f41529 --destination-cidr-block 11.12.0.0/16
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVpnConnectionRoute](#)中的。

## delete-vpn-connection

以下代码示例演示了如何使用 delete-vpn-connection。

### AWS CLI

#### 删除 VPN 连接

此示例删除指定的 VPN 连接。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-vpn-connection --vpn-connection-id vpn-40f41529
```



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVpnConnection](#)中的。

## delete-vpn-gateway

以下代码示例演示了如何使用 delete-vpn-gateway。

### AWS CLI

删除虚拟专用网关

此示例删除了指定的虚拟专用网关。如果命令成功，则不返回任何输出。

命令：

```
aws ec2 delete-vpn-gateway --vpn-gateway-id vgw-9a4cacf3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVpnGateway](#)中的。

## deprovision-byoip-cidr

以下代码示例演示了如何使用 deprovision-byoip-cidr。

### AWS CLI

要移除 IP 地址范围，请使用

以下示例删除了与一起使用的指定地址范围 AWS。

```
aws ec2 deprovision-byoip-cidr \  
  --cidr 203.0.113.25/24
```

输出：

```
{  
  "ByoipCidr": {  
    "Cidr": "203.0.113.25/24",  
    "State": "pending-deprovision"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeprovisionByoipCidr](#)中的。

## deprovision-ipam-pool-cidr

以下代码示例演示了如何使用 `deprovision-ipam-pool-cidr`。

### AWS CLI

#### 取消配置 IPAM 池 CIDR

以下 `deprovision-ipam-pool-cidr` 示例取消配置到 IPAM 池的 CIDR。

(Linux) :

```
aws ec2 deprovision-ipam-pool-cidr \  
  --ipam-pool-id ipam-pool-02ec043a19bbe5d08 \  
  --cidr 11.0.0.0/16
```

( 视窗 ) :

```
aws ec2 deprovision-ipam-pool-cidr ^\  
  --ipam-pool-id ipam-pool-02ec043a19bbe5d08 ^\  
  --cidr 11.0.0.0/16
```

输出 :

```
{  
  "IpamPoolCidr": {  
    "Cidr": "11.0.0.0/16",  
    "State": "pending-deprovision"  
  }  
}
```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的取消配置池 CIDR [R](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeprovisionIpamPoolCidr](#) 中的。

## deregister-image

以下代码示例演示了如何使用 `deregister-image`。

### AWS CLI

#### 取消注册 AMI

此示例取消注册指定的 AMI。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 deregister-image --image-id ami-4fa54026
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterImage](#)中的。

## deregister-instance-event-notification-attributes

以下代码示例演示了如何使用 `deregister-instance-event-notification-attributes`。

AWS CLI

示例 1：从事件通知中移除所有标签

以下 `deregister-instance-event-notification-attributes` 示例移除了 `IncludeAllTagsOfInstance=true`，其效果是 `IncludeAllTagsOfInstance` 将其设置为 `false`。

```
aws ec2 deregister-instance-event-notification-attributes \  
  --instance-tag-attribute IncludeAllTagsOfInstance=true
```

输出：

```
{  
  "InstanceTagAttribute": {  
    "InstanceTagKeys": [],  
    "IncludeAllTagsOfInstance": true  
  }  
}
```

有关更多信息，请参阅适用于 [Linux 实例的 Amazon 弹性计算云用户指南中的实例计划事件](#)。

示例 2：从事件通知中移除特定标签

以下 `deregister-instance-event-notification-attributes` 示例从事件通知中包含的标签中删除指定的标签。要描述事件通知中包含的其余标签，请使用 `describe-instance-event-notification-attributes`。

```
aws ec2 deregister-instance-event-notification-attributes \  
  --instance-tag-attribute InstanceTagKeys="tag-key2"
```

输出：

```
{  
  "InstanceTagAttribute": {  
    "InstanceTagKeys": [  
      "tag-key2"  
    ],  
    "IncludeAllTagsOfInstance": false  
  }  
}
```

有关更多信息，请参阅适用于 [Linux 实例的 Amazon 弹性计算云用户指南中的实例计划事件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeregisterInstanceEventNotificationAttributes](#) 中的。

## deregister-transit-gateway-multicast-group-members

以下代码示例演示了如何使用 `deregister-transit-gateway-multicast-group-members`。

### AWS CLI

从组播组中取消注册群组成员

此示例将指定的网络接口组成员从传输网关组播组中注销。

```
aws ec2 deregister-transit-gateway-multicast-group-members \  
  --transit-gateway-multicast-domain-id tgw-mcast-domain-0c4905cef7EXAMPLE \  
  --group-ip-address 224.0.1.0 \  
  --network-interface-ids eni-0e246d3269EXAMPLE
```

输出：

```
{  
  "DeregisteredMulticastGroupMembers": {  
    "TransitGatewayMulticastDomainId": "tgw-mcast-domain-0c4905cef7EXAMPLE",  
    "RegisteredNetworkInterfaceIds": [  

```

```
        "eni-0e246d3269EXAMPLE"
      ],
      "GroupIpAddress": "224.0.1.0"
    }
  }
}
```

有关更多信息，请参阅 [T AWS Transit Gateways 用户指南中的从组播组注销成员](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeregisterTransitGatewayMulticastGroupMembers](#) 中的。

## deregister-transit-gateway-multicast-group-source

以下代码示例演示了如何使用 `deregister-transit-gateway-multicast-group-source`。

### AWS CLI

从传输网关组播组中取消注册源

此示例从组播组中取消注册指定的网络接口组源。

```
aws ec2 deregister-transit-gateway-multicast-group-sources \
  --transit-gateway-multicast-domain-id tgw-mcast-domain-0c4905cef79d6e597 \
  --group-ip-address 224.0.1.0 \
  --network-interface-ids eni-07f290fc3c090cbae
```

输出：

```
{
  "DeregisteredMulticastGroupSources": {
    "TransitGatewayMulticastDomainId": "tgw-mcast-domain-0c4905cef79d6e597",
    "DeregisteredNetworkInterfaceIds": [
      "eni-07f290fc3c090cbae"
    ],
    "GroupIpAddress": "224.0.1.0"
  }
}
```

有关更多信息，请参阅 [T AWS Transit Gateways 用户指南中的从组播组取消注册源](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeregisterTransitGatewayMulticastGroupSource](#) 中的。

## describe-account-attributes

以下代码示例演示了如何使用 describe-account-attributes。

### AWS CLI

描述您 AWS 账户的所有属性

此示例描述了您的 AWS 账户的属性。

命令:

```
aws ec2 describe-account-attributes
```

输出:

```
{
  "AccountAttributes": [
    {
      "AttributeName": "vpc-max-security-groups-per-interface",
      "AttributeValues": [
        {
          "AttributeValue": "5"
        }
      ]
    },
    {
      "AttributeName": "max-instances",
      "AttributeValues": [
        {
          "AttributeValue": "20"
        }
      ]
    },
    {
      "AttributeName": "supported-platforms",
      "AttributeValues": [
        {
          "AttributeValue": "EC2"
        },
        {
          "AttributeValue": "VPC"
        }
      ]
    }
  ]
}
```

```
    ]
  },
  {
    "AttributeName": "default-vpc",
    "AttributeValues": [
      {
        "AttributeValue": "none"
      }
    ]
  },
  {
    "AttributeName": "max-elastic-ips",
    "AttributeValues": [
      {
        "AttributeValue": "5"
      }
    ]
  },
  {
    "AttributeName": "vpc-max-elastic-ips",
    "AttributeValues": [
      {
        "AttributeValue": "5"
      }
    ]
  }
]
}
```

描述您 AWS 账户的单一属性

此示例描述了您的 AWS 账户的supported-platforms属性。

命令:

```
aws ec2 describe-account-attributes --attribute-names supported-platforms
```

输出:

```
{
  "AccountAttributes": [
    {
```

```

    "AttributeName": "supported-platforms",
    "AttributeValues": [
      {
        "AttributeValue": "EC2"
      },
      {
        "AttributeValue": "VPC"
      }
    ]
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAccountAttributes](#)中的。

## describe-address-transfers

以下代码示例演示了如何使用 describe-address-transfers。

### AWS CLI

#### 描述弹性 IP 地址传输

以下describe-address-transfers示例描述了指定弹性 IP 地址的弹性 IP 地址传输。

```

aws ec2 describe-address-transfers \
  --allocation-ids eipalloc-09ad461b0d03f6aaf

```

输出：

```

{
  "AddressTransfers": [
    {
      "PublicIp": "100.21.184.216",
      "AllocationId": "eipalloc-09ad461b0d03f6aaf",
      "TransferAccountId": "123456789012",
      "TransferOfferExpirationTimestamp": "2023-02-22T22:51:01.000Z",
      "AddressTransferStatus": "pending"
    }
  ]
}

```



有关更多信息，请参阅 Amazon VPC 用户指南中的[传输弹性 IP 地址](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeAddressTransfers](#)中的。

## describe-addresses-attribute

以下代码示例演示了如何使用 describe-addresses-attribute。

### AWS CLI

查看与弹性 IP 地址关联的域名的属性

以下 describe-addresses-attribute 示例返回与弹性 IP 地址关联的域名的属性。

Linux :

```
aws ec2 describe-addresses-attribute \  
  --allocation-ids eipalloc-abcdef01234567890 \  
  --attribute domain-name
```

Windows:

```
aws ec2 describe-addresses-attribute ^  
  --allocation-ids eipalloc-abcdef01234567890 ^  
  --attribute domain-name
```

输出 :

```
{  
  "Addresses": [  
    {  
      "PublicIp": "192.0.2.0",  
      "AllocationId": "eipalloc-abcdef01234567890",  
      "PtrRecord": "example.com."  
    }  
  ]  
}
```

要查看弹性 IP 地址的属性，必须先将域名与弹性 IP 地址相关联。有关更多信息，请参阅 Amazon EC2 用户指南或 AWS CLI 命令参考[modify-address-attribute](#)中的[对电子邮件应用程序使用反向 DNS](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAddressesAttribute](#)中的。

## describe-addresses

以下代码示例演示了如何使用 describe-addresses。

### AWS CLI

示例 1：检索所有弹性 IP 地址的详细信息

以下 describe addresses 示例显示有关弹性 IP 地址的详细信息。

```
aws ec2 describe-addresses
```

输出：

```
{
  "Addresses": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "PublicIp": "198.51.100.0",
      "PublicIpv4Pool": "amazon",
      "Domain": "standard"
    },
    {
      "Domain": "vpc",
      "PublicIpv4Pool": "amazon",
      "InstanceId": "i-1234567890abcdef0",
      "NetworkInterfaceId": "eni-12345678",
      "AssociationId": "eipassoc-12345678",
      "NetworkInterfaceOwnerId": "123456789012",
      "PublicIp": "203.0.113.0",
      "AllocationId": "eipalloc-12345678",
      "PrivateIpAddress": "10.0.1.241"
    }
  ]
}
```

示例 2：检索 EC2-VPC 弹性 IP 地址的详细信息

以下 describe-addresses 示例显示，与 VPC 中的实例配合使用的弹性 IP 地址的详细信息。

```
aws ec2 describe-addresses \  
  --filters "Name=domain,Values=vpc"
```

输出：

```
{  
  "Addresses": [  
    {  
      "Domain": "vpc",  
      "PublicIpv4Pool": "amazon",  
      "InstanceId": "i-1234567890abcdef0",  
      "NetworkInterfaceId": "eni-12345678",  
      "AssociationId": "eipassoc-12345678",  
      "NetworkInterfaceOwnerId": "123456789012",  
      "PublicIp": "203.0.113.0",  
      "AllocationId": "eipalloc-12345678",  
      "PrivateIpAddress": "10.0.1.241"  
    }  
  ]  
}
```

示例 3：检索由分配 ID 指定的弹性 IP 地址的详细信息

以下 describe-addresses 示例显示有关具有指定分配 ID 的弹性 IP 地址的详细信息，该地址与 EC2-VPC 中的实例相关联。

```
aws ec2 describe-addresses \  
  --allocation-ids eipalloc-282d9641
```

输出：

```
{  
  "Addresses": [  
    {  
      "Domain": "vpc",  
      "PublicIpv4Pool": "amazon",  
      "InstanceId": "i-1234567890abcdef0",  
      "NetworkInterfaceId": "eni-1a2b3c4d",  
      "AssociationId": "eipassoc-123abc12",  
      "NetworkInterfaceOwnerId": "1234567891012",  
      "PublicIp": "203.0.113.25",  
    }  
  ]  
}
```

```
        "AllocationId": "eipalloc-282d9641",
        "PrivateIpAddress": "10.251.50.12"
      }
    ]
  }
```

示例 4：检索由 VPC 私有 IP 地址指定的弹性 IP 地址的详细信息

以下 describe-addresses 示例显示，与 EC2-VPC 中特定私有 IP 地址关联的弹性 IP 地址的详细信息。

```
aws ec2 describe-addresses \
  --filters "Name=private-ip-address,Values=10.251.50.12"
```

示例 5：检索 EC2-Classic 中有关弹性 IP 地址的详细信息

以下 describe-addresses 示例显示有关在 EC2-Classic 中使用的弹性 IP 地址的详细信息。

```
aws ec2 describe-addresses \
  --filters "Name=domain,Values=standard"
```

输出：

```
{
  "Addresses": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "PublicIp": "203.0.110.25",
      "PublicIpv4Pool": "amazon",
      "Domain": "standard"
    }
  ]
}
```

示例 6：检索由公有 IP 地址指定的弹性 IP 地址详细信息

以下 describe-addresses 示例显示有关具有值 203.0.110.25 的弹性 IP 地址的详细信息，该地址与 EC2-Classic 中的实例相关联。

```
aws ec2 describe-addresses \
  --public-ips 203.0.110.25
```

输出：

```
{
  "Addresses": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "PublicIp": "203.0.110.25",
      "PublicIpv4Pool": "amazon",
      "Domain": "standard"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAddresses](#)中的。

## describe-aggregate-id-format

以下代码示例演示了如何使用 describe-aggregate-id-format。

AWS CLI

描述区域中所有资源类型的加长 ID 格式设置

以下describe-aggregate-id-format示例描述了当前区域的整体 long ID 格式状态。该Deadline值表示这些资源从短 ID 格式永久切换到长 ID 格式的截止日期已过期。该UseLongIdsAggregated值表示所有 IAM 用户和 IAM 角色均配置为对所有资源类型使用长 ID 格式。

```
aws ec2 describe-aggregate-id-format
```

输出：

```
{
  "UseLongIdsAggregated": true,
  "Statuses": [
    {
      "Deadline": "2018-08-13T02:00:00.000Z",
      "Resource": "network-interface-attachment",
      "UseLongIds": true
    },
    {
```

```

        "Deadline": "2016-12-13T02:00:00.000Z",
        "Resource": "instance",
        "UseLongIds": true
    },
    {
        "Deadline": "2018-08-13T02:00:00.000Z",
        "Resource": "elastic-ip-association",
        "UseLongIds": true
    },
    ...
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAggregateIdFormat](#)中的。

## describe-availability-zones

以下代码示例演示了如何使用 describe-availability-zones。

### AWS CLI

#### 描述可用区

以下示例 describe-availability-zones 显示了可供您使用的可用区详细信息。响应仅包括当前区域的可用区。在本示例中，将使用配置文件默认的 us-west-2（俄勒冈州）区域。

```
aws ec2 describe-availability-zones
```

输出：

```

{
  "AvailabilityZones": [
    {
      "State": "available",
      "OptInStatus": "opt-in-not-required",
      "Messages": [],
      "RegionName": "us-west-2",
      "ZoneName": "us-west-2a",
      "ZoneId": "usw2-az1",
      "GroupName": "us-west-2",
      "NetworkBorderGroup": "us-west-2"
    },

```

```
{
  "State": "available",
  "OptInStatus": "opt-in-not-required",
  "Messages": [],
  "RegionName": "us-west-2",
  "ZoneName": "us-west-2b",
  "ZoneId": "usw2-az2",
  "GroupName": "us-west-2",
  "NetworkBorderGroup": "us-west-2"
},
{
  "State": "available",
  "OptInStatus": "opt-in-not-required",
  "Messages": [],
  "RegionName": "us-west-2",
  "ZoneName": "us-west-2c",
  "ZoneId": "usw2-az3",
  "GroupName": "us-west-2",
  "NetworkBorderGroup": "us-west-2"
},
{
  "State": "available",
  "OptInStatus": "opt-in-not-required",
  "Messages": [],
  "RegionName": "us-west-2",
  "ZoneName": "us-west-2d",
  "ZoneId": "usw2-az4",
  "GroupName": "us-west-2",
  "NetworkBorderGroup": "us-west-2"
},
{
  "State": "available",
  "OptInStatus": "opted-in",
  "Messages": [],
  "RegionName": "us-west-2",
  "ZoneName": "us-west-2-lax-1a",
  "ZoneId": "usw2-lax1-az1",
  "GroupName": "us-west-2-lax-1",
  "NetworkBorderGroup": "us-west-2-lax-1"
}
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAvailabilityZones](#)中的。

## describe-aws-network-performance-metric-subscription

以下代码示例演示了如何使用 `describe-aws-network-performance-metric-subscription`。

### AWS CLI

描述您的指标订阅

以下 `describe-aws-network-performance-metric-subscriptions` 示例描述了您的指标订阅。

```
aws ec2 describe-aws-network-performance-metric-subscriptions
```

输出：

```
{
  "Subscriptions": [
    {
      "Source": "us-east-1",
      "Destination": "eu-west-1",
      "Metric": "aggregate-latency",
      "Statistic": "p50",
      "Period": "five-minutes"
    }
  ]
}
```

有关更多信息，请参阅 [《基础架构性能用户指南》中的管理订阅](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAwsNetworkPerformanceMetricSubscription](#) 中的。

## describe-aws-network-performance-metric-subscriptions

以下代码示例演示了如何使用 `describe-aws-network-performance-metric-subscriptions`。

### AWS CLI

描述您的指标订阅



以下describe-aws-network-performance-metric-subscriptions示例描述了您的指标订阅。

```
aws ec2 describe-aws-network-performance-metric-subscriptions
```

输出：

```
{
  "Subscriptions": [
    {
      "Source": "us-east-1",
      "Destination": "eu-west-1",
      "Metric": "aggregate-latency",
      "Statistic": "p50",
      "Period": "five-minutes"
    }
  ]
}
```

有关更多信息，请参阅[《基础架构性能用户指南》](#)中的[管理订阅](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAwsNetworkPerformanceMetricSubscriptions](#)中的。

## describe-bundle-tasks

以下代码示例演示了如何使用 describe-bundle-tasks。

AWS CLI

描述您的捆绑包任务

此示例描述了您的所有捆绑包任务。

命令：

```
aws ec2 describe-bundle-tasks
```

输出：

```
{
```

```
"BundleTasks": [
  {
    "UpdateTime": "2015-09-15T13:26:54.000Z",
    "InstanceId": "i-1234567890abcdef0",
    "Storage": {
      "S3": {
        "Prefix": "winami",
        "Bucket": "bundletasks"
      }
    },
    "State": "bundling",
    "StartTime": "2015-09-15T13:24:35.000Z",
    "Progress": "3%",
    "BundleId": "bun-2a4e041c"
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeBundleTasks](#)中的。

## describe-byoip-cidrs

以下代码示例演示了如何使用 describe-byoip-cidrs。

### AWS CLI

描述您的预配置地址范围

以下describe-byoip-cidrs示例显示了有关您预配置供使用的公有 IPv4 地址范围的详细信息。AWS

```
aws ec2 describe-byoip-cidrs
```

输出：

```
{
  "ByoipCidrs": [
    {
      "Cidr": "203.0.113.25/24",
      "StatusMessage": "ipv4pool-ec2-1234567890abcdef0",
      "State": "provisioned"
    }
  ]
}
```

```
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeByoipCidrs](#)中的。

## describe-capacity-reservation-fleets

以下代码示例演示了如何使用 describe-capacity-reservation-fleets。

### AWS CLI

#### 查看容量预留队列

以下describe-capacity-reservation-fleets示例列出了指定容量预留队列的配置和容量信息。它还列出了有关舰队内部各个容量预留的详细信息。：

```
aws ec2 describe-capacity-reservation-fleets \
  --capacity-reservation-fleet-ids crf-abcdef01234567890
```

输出：

```
{
  "CapacityReservationFleets": [
    {
      "Status": "active",
      "EndDate": "2022-12-31T23:59:59.000Z",
      "InstanceMatchCriteria": "open",
      "Tags": [],
      "CapacityReservationFleetId": "crf-abcdef01234567890",
      "Tenancy": "default",
      "InstanceTypeSpecifications": [
        {
          "CapacityReservationId": "cr-1234567890abcdef0",
          "AvailabilityZone": "us-east-1a",
          "FulfilledCapacity": 5.0,
          "Weight": 1.0,
          "CreateDate": "2022-07-02T08:34:33.398Z",
          "InstancePlatform": "Linux/UNIX",
          "TotalInstanceCount": 5,
          "Priority": 1,
          "EbsOptimized": true,
          "InstanceType": "m5.xlarge"
        }
      ]
    }
  ]
}
```

```

        }
      ],
      "TotalTargetCapacity": 5,
      "TotalFulfilledCapacity": 5.0,
      "CreateTime": "2022-07-02T08:34:33.397Z",
      "AllocationStrategy": "prioritized"
    }
  ]
}

```

有关容量预留队列的更多信息，请参阅 Amazon EC2 用户 [指南中的容量预留队列](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeCapacityReservationFleets](#) 中的。

## describe-capacity-reservations

以下代码示例演示了如何使用 describe-capacity-reservations。

### AWS CLI

示例 1：描述您的一个或多个容量预留

以下 describe-capacity-reservations 示例显示了有关您在当前 AWS 地区的所有容量预留的详细信息。

```
aws ec2 describe-capacity-reservations
```

输出：

```

{
  "CapacityReservations": [
    {
      "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
      "EndDateType": "unlimited",
      "AvailabilityZone": "eu-west-1a",
      "InstanceMatchCriteria": "open",
      "Tags": [],
      "EphemeralStorage": false,
      "CreateDate": "2019-08-16T09:03:18.000Z",
      "AvailableInstanceCount": 1,
      "InstancePlatform": "Linux/UNIX",
      "TotalInstanceCount": 1,
    }
  ]
}

```

```

        "State": "active",
        "Tenancy": "default",
        "EbsOptimized": true,
        "InstanceType": "a1.medium"
    },
    {
        "CapacityReservationId": "cr-abcdEXAMPLE9876ef ",
        "EndDateType": "unlimited",
        "AvailabilityZone": "eu-west-1a",
        "InstanceMatchCriteria": "open",
        "Tags": [],
        "EphemeralStorage": false,
        "CreateDate": "2019-08-07T11:34:19.000Z",
        "AvailableInstanceCount": 3,
        "InstancePlatform": "Linux/UNIX",
        "TotalInstanceCount": 3,
        "State": "cancelled",
        "Tenancy": "default",
        "EbsOptimized": true,
        "InstanceType": "m5.large"
    }
]
}

```

## 示例 2：描述您的一个或多个容量预留

以下describe-capacity-reservations示例显示了有关指定容量预留的详细信息。

```
aws ec2 describe-capacity-reservations \
  --capacity-reservation-ids cr-1234abcd56EXAMPLE
```

输出：

```

{
  "CapacityReservations": [
    {
      "CapacityReservationId": "cr-1234abcd56EXAMPLE",
      "EndDateType": "unlimited",
      "AvailabilityZone": "eu-west-1a",
      "InstanceMatchCriteria": "open",
      "Tags": [],
      "EphemeralStorage": false,
      "CreateDate": "2019-08-16T09:03:18.000Z",

```

```
    "AvailableInstanceCount": 1,
    "InstancePlatform": "Linux/UNIX",
    "TotalInstanceCount": 1,
    "State": "active",
    "Tenancy": "default",
    "EbsOptimized": true,
    "InstanceType": "a1.medium"
  }
]
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[查看容量预留](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCapacityReservations](#)中的。

## describe-carrier-gateways

以下代码示例演示了如何使用 describe-carrier-gateways。

### AWS CLI

描述所有运营商网关

以下describe-carrier-gateways示例列出了您的所有运营商网关。

```
aws ec2 describe-carrier-gateways
```

输出：

```
{
  "CarrierGateways": [
    {
      "CarrierGatewayId": "cagw-0465cdEXAMPLE1111",
      "VpcId": "vpc-0c529aEXAMPLE",
      "State": "available",
      "OwnerId": "123456789012",
      "Tags": [
        {
          "Key": "example",
          "Value": "tag"
        }
      ]
    }
  ]
}
```

```
    ]
  }
]
}
```

有关更多信息，请参阅 < [https://docs.aws.amazon.com/vpc/latest/userguide/Carrier\\_Gateway.html](https://docs.aws.amazon.com/vpc/latest/userguide/Carrier_Gateway.html) > Amazon Virtual Private Cloud 用户指南中的运营商网关。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeCarrierGateways](#) 中的。

## describe-classic-link-instances

以下代码示例演示了如何使用 `describe-classic-link-instances`。

### AWS CLI

描述关联的 EC2-Classical 实例

此示例列出了所有关联的 EC2-Classical 实例。

命令：

```
aws ec2 describe-classic-link-instances
```

输出：

```
{
  "Instances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "VpcId": "vpc-88888888",
      "Groups": [
        {
          "GroupId": "sg-11122233"
        }
      ],
      "Tags": [
        {
          "Value": "ClassicInstance",
          "Key": "Name"
        }
      ]
    }
  ]
}
```

```

    },
    {
      "InstanceId": "i-0598c7d356eba48d7",
      "VpcId": "vpc-12312312",
      "Groups": [
        {
          "GroupId": "sg-aabbccdd"
        }
      ],
      "Tags": [
        {
          "Value": "ClassicInstance2",
          "Key": "Name"
        }
      ]
    }
  ]
}

```

此示例列出了您所有关联的 EC2-Classic 实例，并筛选响应以仅包括链接到 VPC vpc-88888888 的实例。

命令:

```
aws ec2 describe-classic-link-instances --filter "Name=vpc-id,Values=vpc-88888888"
```

输出:

```

{
  "Instances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "VpcId": "vpc-88888888",
      "Groups": [
        {
          "GroupId": "sg-11122233"
        }
      ],
      "Tags": [
        {
          "Value": "ClassicInstance",
          "Key": "Name"
        }
      ]
    }
  ]
}

```



```
}
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeClassicLinkInstances](#)中的。

## describe-client-vpn-authorization-rules

以下代码示例演示了如何使用 `describe-client-vpn-authorization-rules`。

### AWS CLI

描述 Client VPN 端点的授权规则

以下`describe-client-vpn-authorization-rules`示例显示了有关指定 Client VPN 端点的授权规则的详细信息。

```
aws ec2 describe-client-vpn-authorization-rules \
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde
```

输出：

```
{
  "AuthorizationRules": [
    {
      "ClientVpnEndpointId": "cvpn-endpoint-123456789123abcde",
      "GroupId": "",
      "AccessAll": true,
      "DestinationCidr": "0.0.0.0/0",
      "Status": {
        "Code": "active"
      }
    }
  ]
}
```

有关更多信息，请参阅《AWS Client VPN 管理员指南》中的[授权规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeClientVpnAuthorizationRules](#)中的。

## describe-client-vpn-connections

以下代码示例演示了如何使用 describe-client-vpn-connections。

### AWS CLI

描述与 Client VPN 端点的连接

以下 describe-client-vpn-connections 示例显示了与指定 Client VPN 端点的客户端连接的详细信息。

```
aws ec2 describe-client-vpn-connections \
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde
```

输出：

```
{
  "Connections": [
    {
      "ClientVpnEndpointId": "cvpn-endpoint-123456789123abcde",
      "Timestamp": "2019-08-12 07:58:34",
      "ConnectionId": "cvpn-connection-0e03eb24267165acd",
      "ConnectionEstablishedTime": "2019-08-12 07:57:14",
      "IngressBytes": "32302",
      "EgressBytes": "5696",
      "IngressPackets": "332",
      "EgressPackets": "67",
      "ClientIp": "172.31.0.225",
      "CommonName": "client1.domain.tld",
      "Status": {
        "Code": "terminated"
      },
      "ConnectionEndTime": "2019-08-12 07:58:34"
    },
    {
      "ClientVpnEndpointId": "cvpn-endpoint-123456789123abcde",
      "Timestamp": "2019-08-12 08:02:54",
      "ConnectionId": "cvpn-connection-00668867a40f18253",
      "ConnectionEstablishedTime": "2019-08-12 08:02:53",
      "IngressBytes": "2951",
      "EgressBytes": "2611",
      "IngressPackets": "9",
      "EgressPackets": "6",
```

```
    "ClientIp": "172.31.0.226",
    "CommonName": "client1.domain.tld",
    "Status": {
      "Code": "active"
    },
    "ConnectionEndTime": "-"
  }
]
```

有关更多信息，请参阅《Client VPN 管理员指南》中的“AWS 客户端[连接](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeClientVpnConnections](#) 中的。

## describe-client-vpn-endpoints

以下代码示例演示了如何使用 describe-client-vpn-endpoints。

### AWS CLI

描述您的客户端 VPN 端点

以下 describe-client-vpn-endpoints 示例显示有关您的所有 Client VPN 终端节点的详细信息。

```
aws ec2 describe-client-vpn-endpoints
```

输出：

```
{
  "ClientVpnEndpoints": [
    {
      "ClientVpnEndpointId": "cvpn-endpoint-123456789123abcde",
      "Description": "Endpoint for Admin access",
      "Status": {
        "Code": "available"
      },
      "CreationTime": "2020-11-13T11:37:27",
      "DnsName": "*.cvpn-endpoint-123456789123abcde.prod.clientvpn.ap-
south-1.amazonaws.com",
      "ClientCidrBlock": "172.31.0.0/16",
      "DnsServers": [
        "8.8.8.8"
      ]
    }
  ]
}
```

```

    ],
    "SplitTunnel": false,
    "VpnProtocol": "openvpn",
    "TransportProtocol": "udp",
    "VpnPort": 443,
    "ServerCertificateArn": "arn:aws:acm:ap-
south-1:123456789012:certificate/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "AuthenticationOptions": [
      {
        "Type": "certificate-authentication",
        "MutualAuthentication": {
          "ClientRootCertificateChain": "arn:aws:acm:ap-
south-1:123456789012:certificate/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE"
        }
      }
    ],
    "ConnectionLogOptions": {
      "Enabled": true,
      "CloudwatchLogGroup": "Client-vpn-connection-logs",
      "CloudwatchLogStream": "cvpn-endpoint-123456789123abcde-ap-
south-1-2020/11/13-FCD8HEMVAcCw"
    },
    "Tags": [
      {
        "Key": "Name",
        "Value": "Client VPN"
      }
    ],
    "SecurityGroupIds": [
      "sg-aabbcc11223344567"
    ],
    "VpcId": "vpc-a87f92c1",
    "SelfServicePortalUrl": "https://self-service.clientvpn.amazonaws.com/
endpoints/cvpn-endpoint-123456789123abcde",
    "ClientConnectOptions": {
      "Enabled": false
    }
  }
]
}

```

有关更多信息，请参阅 [《客户端 VPN 管理员指南》](#) 中的 [AWS Client VPN 端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeClientVpnEndpoints](#) 中的。

## describe-client-vpn-routes

以下代码示例演示了如何使用 `describe-client-vpn-routes`。

### AWS CLI

描述 Client VPN 终端节点的路由

以下 `describe-client-vpn-routes` 示例显示了有关指定 Client VPN 终端节点的路由的详细信息。

```
aws ec2 describe-client-vpn-routes \  
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde
```

输出：

```
{  
  "Routes": [  
    {  
      "ClientVpnEndpointId": "cvpn-endpoint-123456789123abcde",  
      "DestinationCidr": "10.0.0.0/16",  
      "TargetSubnet": "subnet-0123456789abcabca",  
      "Type": "Nat",  
      "Origin": "associate",  
      "Status": {  
        "Code": "active"  
      },  
      "Description": "Default Route"  
    },  
    {  
      "ClientVpnEndpointId": "cvpn-endpoint-123456789123abcde",  
      "DestinationCidr": "0.0.0.0/0",  
      "TargetSubnet": "subnet-0123456789abcabca",  
      "Type": "Nat",  
      "Origin": "add-route",  
      "Status": {  
        "Code": "active"  
      }  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS Client VPN 管理员指南》中的[路由](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeClientVpnRoutes](#)中的。

## describe-client-vpn-target-networks

以下代码示例演示了如何使用 `describe-client-vpn-target-networks`。

### AWS CLI

描述 Client VPN 终端节点的目标网络

以下`describe-client-vpn-target-networks`示例显示了有关指定 Client VPN 终端节点的目标网络的详细信息。

```
aws ec2 describe-client-vpn-target-networks \
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde
```

输出：

```
{
  "ClientVpnTargetNetworks": [
    {
      "AssociationId": "cvpn-assoc-012e837060753dc3d",
      "VpcId": "vpc-1111122222233333",
      "TargetNetworkId": "subnet-0123456789abcabca",
      "ClientVpnEndpointId": "cvpn-endpoint-123456789123abcde",
      "Status": {
        "Code": "associating"
      },
      "SecurityGroups": [
        "sg-012345678910abcab"
      ]
    }
  ]
}
```

有关更多信息，请参阅《AWS Client VPN 管理员指南》中的“[目标网络](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeClientVpnTargetNetworks](#)中的。

## describe-coip-pools

以下代码示例演示了如何使用 `describe-coip-pools`。

## AWS CLI

描述客户拥有的 IP 地址池

以下describe-coip-pools示例描述了您 AWS 账户中客户拥有的 IP 地址池。

```
aws ec2 describe-coip-pools
```

输出：

```
{
  "CoipPools": [
    {
      "PoolId": "ipv4pool-coip-123a45678bEXAMPLE",
      "PoolCidrs": [
        "0.0.0.0/0"
      ],
      "LocalGatewayRouteTableId": "lgw-rtb-059615ef7dEXAMPLE",
      "PoolArn": "arn:aws:ec2:us-west-2:123456789012:coip-pool/ipv4pool-coip-123a45678bEXAMPLE"
    }
  ]
}
```

有关更多信息，请参阅 [《AWS Outposts 用户指南》](#) 中的客户拥有的 IP 地址。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCoipPools](#)中的。

## describe-conversion-tasks

以下代码示例演示了如何使用 describe-conversion-tasks。

### AWS CLI

查看转换任务的状态

此示例返回 ID 为 import-i-ffvko 9js 的转换任务的状态。

命令：

```
aws ec2 describe-conversion-tasks --conversion-task-ids import-i-ffvko9js
```

输出：

```
{
  "ConversionTasks": [
    {
      "ConversionTaskId": "import-i-ffvko9js",
      "ImportInstance": {
        "InstanceId": "i-1234567890abcdef0",
        "Volumes": [
          {
            "Volume": {
              "Id": "vol-049df61146c4d7901",
              "Size": 16
            },
            "Status": "completed",
            "Image": {
              "Size": 1300687360,
              "ImportManifestUrl": "https://s3.amazonaws.com/myimportbucket/411443cd-d620-4f1c-9d66-13144EXAMPLE/RHEL5.vmdkmanifest.xml?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Expires=140EXAMPLE&Signature=XYNhznHNgcqsjDxL9wRL%2FJvEXAMPLE",
              "Format": "VMDK"
            },
            "BytesConverted": 1300682960,
            "AvailabilityZone": "us-east-1d"
          }
        ]
      },
      "ExpirationTime": "2014-05-14T22:06:23Z",
      "State": "completed"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeConversionTasks](#)中的。

## describe-customer-gateways

以下代码示例演示了如何使用 describe-customer-gateways。

### AWS CLI

描述您的客户网关



此示例描述了您的客户网关。

命令:

```
aws ec2 describe-customer-gateways
```

输出:

```
{
  "CustomerGateways": [
    {
      "CustomerGatewayId": "cgw-b4dc3961",
      "IpAddress": "203.0.113.12",
      "State": "available",
      "Type": "ipsec.1",
      "BgpAsn": "65000"
    },
    {
      "CustomerGatewayId": "cgw-0e11f167",
      "IpAddress": "12.1.2.3",
      "State": "available",
      "Type": "ipsec.1",
      "BgpAsn": "65534"
    }
  ]
}
```

描述特定的客户网关

此示例描述了指定的客户网关。

命令:

```
aws ec2 describe-customer-gateways --customer-gateway-ids cgw-0e11f167
```

输出:

```
{
  "CustomerGateways": [
    {
      "CustomerGatewayId": "cgw-0e11f167",
      "IpAddress": "12.1.2.3",
```

```
        "State": "available",
        "Type": "ipsec.1",
        "BgpAsn": "65534"
    }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCustomerGateways](#)中的。

## describe-dhcp-options

以下代码示例演示了如何使用 describe-dhcp-options。

### AWS CLI

#### 示例 1：描述您的 DHCP 选项

以下describe-dhcp-options示例检索有关您的 DHCP 选项的详细信息。

```
aws ec2 describe-dhcp-options
```

输出：

```
{
  "DhcpOptions": [
    {
      "DhcpConfigurations": [
        {
          "Key": "domain-name",
          "Values": [
            {
              "Value": "us-east-2.compute.internal"
            }
          ]
        },
        {
          "Key": "domain-name-servers",
          "Values": [
            {
              "Value": "AmazonProvidedDNS"
            }
          ]
        }
      ]
    }
  ]
}
```

```

    }
  ],
  "DhcpOptionsId": "dopt-19edf471",
  "OwnerId": "111122223333"
},
{
  "DhcpConfigurations": [
    {
      "Key": "domain-name",
      "Values": [
        {
          "Value": "us-east-2.compute.internal"
        }
      ]
    },
    {
      "Key": "domain-name-servers",
      "Values": [
        {
          "Value": "AmazonProvidedDNS"
        }
      ]
    }
  ]
},
  "DhcpOptionsId": "dopt-fEXAMPLE",
  "OwnerId": "111122223333"
}
]
}

```

有关更多信息，请参阅《AWS VPC 用户指南》中的[“使用 DHCP 选项集”](#)。

示例 2：描述您的 DHCP 选项并筛选输出

以下describe-dhcp-options示例描述了您的 DHCP 选项，并使用筛选器仅返回example.com适用于域名服务器的 DHCP 选项。该示例使用--query参数在输出中仅显示配置信息和 ID。

```

aws ec2 describe-dhcp-options \
  --filters Name=key,Values=domain-name-servers Name=value,Values=example.com \
  --query "DhcpOptions[*].[DhcpConfigurations,DhcpOptionsId]"

```

输出：

```
[
  [
    [
      {
        "Key": "domain-name",
        "Values": [
          {
            "Value": "example.com"
          }
        ]
      },
      {
        "Key": "domain-name-servers",
        "Values": [
          {
            "Value": "172.16.16.16"
          }
        ]
      }
    ],
    "dopt-001122334455667ab"
  ]
]
```

有关更多信息，请参阅《AWS VPC 用户指南》中的[“使用 DHCP 选项集”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeDhcpOptions](#) 中的。

## describe-egress-only-internet-gateways

以下代码示例演示了如何使用 `describe-egress-only-internet-gateways`。

### AWS CLI

描述您的仅限出口的互联网网关

此示例描述了您的仅限出口的互联网网关。

命令:

```
aws ec2 describe-egress-only-internet-gateways
```

输出:

```
{
  "EgressOnlyInternetGateways": [
    {
      "EgressOnlyInternetGatewayId": "eigw-015e0e244e24dfe8a",
      "Attachments": [
        {
          "State": "attached",
          "VpcId": "vpc-0c62a468"
        }
      ]
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEgressOnlyInternetGateways](#)中的。

## describe-elastic-gpus

以下代码示例演示了如何使用 describe-elastic-gpus。

### AWS CLI

描述弹性 GPU

命令:

```
aws ec2 describe-elastic-gpus --elastic-gpu-ids
egpu-12345678901234567890abcdefghijkl
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeElasticGpus](#)中的。

## describe-export-image-tasks

以下代码示例演示了如何使用 describe-export-image-tasks。

### AWS CLI

监视导出图像任务

以下describe-export-image-tasks示例检查指定导出图像任务的状态。在 Amazon S3 中生成的图像文件是my-export-bucket/exports/export-ami-1234567890abcdef0.vmdk。

```
aws ec2 describe-export-image-tasks \  
  --export-image-task-ids export-ami-1234567890abcdef0
```

正在执行的导出图像任务的输出。

```
{  
  "ExportImageTasks": [  
    {  
      "ExportImageTaskId": "export-ami-1234567890abcdef0"  
      "Progress": "21",  
      "S3ExportLocation": {  
        "S3Bucket": "my-export-bucket",  
        "S3Prefix": "exports/"  
      },  
      "Status": "active",  
      "StatusMessage": "updating"  
    }  
  ]  
}
```

已完成的导出图像任务的输出。

```
{  
  "ExportImageTasks": [  
    {  
      "ExportImageTaskId": "export-ami-1234567890abcdef0"  
      "S3ExportLocation": {  
        "S3Bucket": "my-export-bucket",  
        "S3Prefix": "exports/"  
      },  
      "Status": "completed"  
    }  
  ]  
}
```

有关更多信息，请参阅《[虚拟机导入/导出用户指南](#)》中的“从 AMI 导出虚拟机”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeExportImageTasks](#) 中的。

## describe-export-tasks

以下代码示例演示了如何使用 describe-export-tasks。

## AWS CLI

列出有关实例导出任务的详细信息

此示例描述了 ID 为 `export-i-fh8sjjsq` 的导出任务。

命令:

```
aws ec2 describe-export-tasks --export-task-ids export-i-fh8sjjsq
```

输出:

```
{
  "ExportTasks": [
    {
      "State": "active",
      "InstanceExportDetails": {
        "InstanceId": "i-1234567890abcdef0",
        "TargetEnvironment": "vmware"
      },
      "ExportToS3Task": {
        "S3Bucket": "myexportbucket",
        "S3Key": "RHEL5export-i-fh8sjjsq.ova",
        "DiskImageFormat": "vmdk",
        "ContainerFormat": "ova"
      },
      "Description": "RHEL5 instance",
      "ExportTaskId": "export-i-fh8sjjsq"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeExportTasks](#)中的。

## **describe-fast-launch-images**

以下代码示例演示了如何使用 `describe-fast-launch-images`。

## AWS CLI

描述为加快启动速度而配置的 Windows AMI 的详细信息

以下describe-fast-launch-images示例描述了您的账户中为更快启动而配置的每个 AMI 的详细信息，包括资源类型、快照配置、启动模板详情、最大并行启动次数、AMI 所有者 ID、快速启动配置的状态、状态更改的原因以及状态更改发生的时间。

```
aws ec2 describe-fast-launch-images
```

输出：

```
{
  "FastLaunchImages": [
    {
      "ImageId": "ami-01234567890abcdef",
      "ResourceType": "snapshot",
      "SnapshotConfiguration": {},
      "LaunchTemplate": {
        "LaunchTemplateId": "lt-01234567890abcdef",
        "LaunchTemplateName": "EC2FastLaunchDefaultResourceCreation-
a8c6215d-94e6-441b-9272-dbd1f87b07e2",
        "Version": "1"
      },
      "MaxParallelLaunches": 6,
      "OwnerId": "0123456789123",
      "State": "enabled",
      "StateTransitionReason": "Client.UserInitiated",
      "StateTransitionTime": "2022-01-27T22:20:06.552000+00:00"
    }
  ]
}
```

有关配置 Windows AMI 以加快启动速度的更多信息，请参阅 Amazon EC2 用户指南中的[配置 AMI 以加快启动速度](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFastLaunchImages](#)中的。

## describe-fast-snapshot-restores

以下代码示例演示了如何使用 describe-fast-snapshot-restores。

AWS CLI

描述快速快照恢复



以下describe-fast-snapshot-restores示例显示了状态为的所有快速快照恢复的disabled详细信息。

```
aws ec2 describe-fast-snapshot-restores \
  --filters Name=state,Values=disabled
```

输出：

```
{
  "FastSnapshotRestores": [
    {
      "SnapshotId": "snap-1234567890abcdef0",
      "AvailabilityZone": "us-west-2c",
      "State": "disabled",
      "StateTransitionReason": "Client.UserInitiated - Lifecycle state
transition",
      "OwnerId": "123456789012",
      "EnablingTime": "2020-01-25T23:57:49.596Z",
      "OptimizingTime": "2020-01-25T23:58:25.573Z",
      "EnabledTime": "2020-01-25T23:59:29.852Z",
      "DisablingTime": "2020-01-26T00:40:56.069Z",
      "DisabledTime": "2020-01-26T00:41:27.390Z"
    }
  ]
}
```

以下describe-fast-snapshot-restores示例描述了所有快速快照恢复。

```
aws ec2 describe-fast-snapshot-restores
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFastSnapshotRestores](#)中的。

## describe-fleet-history

以下代码示例演示了如何使用 describe-fleet-history。

### AWS CLI

描述 EC2 队列的历史

以下describe-fleet-history示例返回从指定时间开始的指定 EC2 队列的历史记录。输出适用于具有两个正在运行的实例的 EC2 队列。

```
aws ec2 describe-fleet-history \  
  --fleet-id fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE \  
  --start-time 2020-09-01T00:00:00Z
```

输出：

```
{  
  "HistoryRecords": [  
    {  
      "EventInformation": {  
        "EventSubType": "submitted"  
      },  
      "EventType": "fleetRequestChange",  
      "Timestamp": "2020-09-01T18:26:05.000Z"  
    },  
    {  
      "EventInformation": {  
        "EventSubType": "active"  
      },  
      "EventType": "fleetRequestChange",  
      "Timestamp": "2020-09-01T18:26:15.000Z"  
    },  
    {  
      "EventInformation": {  
        "EventDescription": "t2.small, ami-07c8bc5c1ce9598c3, ...",  
        "EventSubType": "progress"  
      },  
      "EventType": "fleetRequestChange",  
      "Timestamp": "2020-09-01T18:26:17.000Z"  
    },  
    {  
      "EventInformation": {  
        "EventDescription": "{\"instanceType\":\"t2.small\", ...}",  
        "EventSubType": "launched",  
        "InstanceId": "i-083a1c446e66085d2"  
      },  
      "EventType": "instanceChange",  
      "Timestamp": "2020-09-01T18:26:17.000Z"  
    },  
    {  
      "EventInformation": {  
        "EventDescription": "{\"instanceType\":\"t2.small\", ...}",  
        "EventSubType": "launched",  
      }  
    }  
  ]  
}
```

```
        "InstanceId": "i-090db02406cc3c2d6"
      },
      "EventType": "instanceChange",
      "Timestamp": "2020-09-01T18:26:17.000Z"
    }
  ],
  "LastEvaluatedTime": "2020-09-01T19:10:19.000Z",
  "FleetId": "fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE",
  "StartTime": "2020-08-31T23:53:20.000Z"
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[管理 EC2 队列](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFleetHistory](#)中的。

## describe-fleet-instances

以下代码示例演示了如何使用 describe-fleet-instances。

### AWS CLI

描述 EC2 队列的运行实例

以下describe-fleet-instances示例描述了指定 EC2 队列的运行实例。

```
aws ec2 describe-fleet-instances \
  --fleet-id 12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE
```

输出：

```
{
  "ActiveInstances": [
    {
      "InstanceId": "i-090db02406cc3c2d6",
      "InstanceType": "t2.small",
      "SpotInstanceRequestId": "sir-a43gtpfk",
      "InstanceHealth": "healthy"
    },
    {
      "InstanceId": "i-083a1c446e66085d2",
      "InstanceType": "t2.small",
      "SpotInstanceRequestId": "sir-iwcit2nj",

```

```
        "InstanceHealth": "healthy"
      }
    ],
    "FleetId": "fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE"
  }
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[管理 EC2 队列](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFleetInstances](#)中的。

## describe-fleets

以下代码示例演示了如何使用 describe-fleets。

### AWS CLI

#### 描述 EC2 队列

以下describe-fleets示例描述了指定的 EC2 队列。

```
aws ec2 describe-fleets \
  --fleet-ids fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE
```

输出：

```
{
  "Fleets": [
    {
      "ActivityStatus": "pending_fulfillment",
      "CreateTime": "2020-09-01T18:26:05.000Z",
      "FleetId": "fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE",
      "FleetState": "active",
      "ExcessCapacityTerminationPolicy": "termination",
      "FulfilledCapacity": 0.0,
      "FulfilledOnDemandCapacity": 0.0,
      "LaunchTemplateConfigs": [
        {
          "LaunchTemplateSpecification": {
            "LaunchTemplateId": "lt-0e632f2855a979cd5",
            "Version": "1"
          }
        }
      ]
    }
  ]
}
```

```
    ],
    "TargetCapacitySpecification": {
      "TotalTargetCapacity": 2,
      "OnDemandTargetCapacity": 0,
      "SpotTargetCapacity": 2,
      "DefaultTargetCapacityType": "spot"
    },
    "TerminateInstancesWithExpiration": false,
    "Type": "maintain",
    "ReplaceUnhealthyInstances": false,
    "SpotOptions": {
      "AllocationStrategy": "lowestPrice",
      "InstanceInterruptionBehavior": "terminate",
      "InstancePoolsToUseCount": 1
    },
    "OnDemandOptions": {
      "AllocationStrategy": "lowestPrice"
    }
  }
]
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[管理 EC2 队列](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFleets](#)中的。

## describe-flow-logs

以下代码示例演示了如何使用 describe-flow-logs。

### AWS CLI

示例 1：描述您的所有流日志

以下describe-flow-logs示例显示了所有流日志的详细信息。

```
aws ec2 describe-flow-logs
```

输出：

```
{
  "FlowLogs": [
```

```

    {
      "CreationTime": "2018-02-21T13:22:12.644Z",
      "DeliverLogsPermissionArn": "arn:aws:iam::123456789012:role/flow-logs-
role",
      "DeliverLogsStatus": "SUCCESS",
      "FlowLogId": "fl-aabbccdd112233445",
      "MaxAggregationInterval": 600,
      "FlowLogStatus": "ACTIVE",
      "LogGroupName": "FlowLogGroup",
      "ResourceId": "subnet-12345678901234567",
      "TrafficType": "ALL",
      "LogDestinationType": "cloud-watch-logs",
      "LogFormat": "${version} ${account-id} ${interface-id} ${srcaddr}
${dstaddr} ${srcport} ${dstport} ${protocol} ${packets} ${bytes} ${start} ${end}
${action} ${log-status}"
    },
    {
      "CreationTime": "2020-02-04T15:22:29.986Z",
      "DeliverLogsStatus": "SUCCESS",
      "FlowLogId": "fl-01234567890123456",
      "MaxAggregationInterval": 60,
      "FlowLogStatus": "ACTIVE",
      "ResourceId": "vpc-00112233445566778",
      "TrafficType": "ACCEPT",
      "LogDestinationType": "s3",
      "LogDestination": "arn:aws:s3:::my-flow-log-bucket/custom",
      "LogFormat": "${version} ${vpc-id} ${subnet-id} ${instance-id}
${interface-id} ${account-id} ${type} ${srcaddr} ${dstaddr} ${srcport} ${dstport}
${pkt-srcaddr} ${pkt-dstaddr} ${protocol} ${bytes} ${packets} ${start} ${end}
${action} ${tcp-flags} ${log-status}"
    }
  ]
}

```

## 示例 2：描述您的流日志的子集

以下describe-flow-logs示例使用筛选器仅显示 Amazon CloudWatch Logs 中指定日志组中的流日志的详细信息。

```

aws ec2 describe-flow-logs \
  --filter "Name=log-group-name,Values=MyFlowLogs"

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFlowLogs](#)中的。

## describe-fpga-image-attribute

以下代码示例演示了如何使用 `describe-fpga-image-attribute`。

### AWS CLI

描述 Amazon FPGA 镜像的属性

此示例描述了指定 AFI 的加载权限。

命令:

```
aws ec2 describe-fpga-image-attribute --fpga-image-id afi-0d123e123bfc85abc --
attribute loadPermission
```

输出:

```
{
  "FpgaImageAttribute": {
    "FpgaImageId": "afi-0d123e123bfc85abc",
    "LoadPermissions": [
      {
        "UserId": "123456789012"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFpgaImageAttribute](#)中的。

## describe-fpga-images

以下代码示例演示了如何使用 `describe-fpga-images`。

### AWS CLI

描述亚马逊 FPGA 镜像

此示例描述了账户123456789012拥有的 AFI。

命令:

```
aws ec2 describe-fpga-images --filters Name=owner-id,Values=123456789012
```

输出：

```
{
  "FpgaImages": [
    {
      "UpdateTime": "2017-12-22T12:09:14.000Z",
      "Name": "my-afi",
      "PciId": {
        "SubsystemVendorId": "0xfedd",
        "VendorId": "0x1d0f",
        "DeviceId": "0xf000",
        "SubsystemId": "0x1d51"
      },
      "FpgaImageGlobalId": "agfi-123cb27b5e84a0abc",
      "Public": false,
      "State": {
        "Code": "available"
      },
      "ShellVersion": "0x071417d3",
      "OwnerId": "123456789012",
      "FpgaImageId": "afi-0d123e123bfc85abc",
      "CreateTime": "2017-12-22T11:43:33.000Z",
      "Description": "my-afi"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFpgaImages](#)中的。

## describe-host-reservation-offerings

以下代码示例演示了如何使用 describe-host-reservation-offerings。

### AWS CLI

#### 描述专用主机预留服务

此示例描述了可供购买的 M4 实例系列的专用主机预留。

命令：

```
aws ec2 describe-host-reservation-offerings --filter Name=instance-family,Values=m4
```



输出：

```
{
  "OfferingSet": [
    {
      "HourlyPrice": "1.499",
      "OfferingId": "hro-03f707bf363b6b324",
      "InstanceFamily": "m4",
      "PaymentOption": "NoUpfront",
      "UpfrontPrice": "0.000",
      "Duration": 31536000
    },
    {
      "HourlyPrice": "1.045",
      "OfferingId": "hro-0ef9181cabdef7a02",
      "InstanceFamily": "m4",
      "PaymentOption": "NoUpfront",
      "UpfrontPrice": "0.000",
      "Duration": 94608000
    },
    {
      "HourlyPrice": "0.714",
      "OfferingId": "hro-04567a15500b92a51",
      "InstanceFamily": "m4",
      "PaymentOption": "PartialUpfront",
      "UpfrontPrice": "6254.000",
      "Duration": 31536000
    },
    {
      "HourlyPrice": "0.484",
      "OfferingId": "hro-0d5d7a9d23ed7fbfe",
      "InstanceFamily": "m4",
      "PaymentOption": "PartialUpfront",
      "UpfrontPrice": "12720.000",
      "Duration": 94608000
    },
    {
      "HourlyPrice": "0.000",
      "OfferingId": "hro-05da4108ca998c2e5",
      "InstanceFamily": "m4",
      "PaymentOption": "AllUpfront",
      "UpfrontPrice": "23913.000",
      "Duration": 94608000
    }
  ]
}
```

```
{
  "HourlyPrice": "0.000",
  "OfferingId": "hro-0a9f9be3b95a3dc8f",
  "InstanceFamily": "m4",
  "PaymentOption": "AllUpfront",
  "UpfrontPrice": "12257.000",
  "Duration": 31536000
}
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeHostReservationOfferings](#)中的。

## describe-host-reservations

以下代码示例演示了如何使用 describe-host-reservations。

### AWS CLI

描述您账户中的专用主机预留

此示例描述了您账户中的专用主机预留。

命令:

```
aws ec2 describe-host-reservations
```

输出:

```
{
  "HostReservationSet": [
    {
      "Count": 1,
      "End": "2019-01-10T12:14:09Z",
      "HourlyPrice": "1.499",
      "InstanceFamily": "m4",
      "OfferingId": "hro-03f707bf363b6b324",
      "PaymentOption": "NoUpfront",
      "State": "active",
      "HostIdSet": [
        "h-013abcd2a00cbd123"
      ],
    }
  ],
}
```

```
    "Start": "2018-01-10T12:14:09Z",
    "HostReservationId": "hr-0d418a3a4ffc669ae",
    "UpfrontPrice": "0.000",
    "Duration": 31536000
  }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeHostReservations](#)中的。

## describe-hosts

以下代码示例演示了如何使用 describe-hosts。

### AWS CLI

查看有关专用主机的详细信息

以下describe-hosts示例显示了您 AWS 账户中available专用主机的详细信息。

```
aws ec2 describe-hosts --filter "Name=state,Values=available"
```

输出：

```
{
  "Hosts": [
    {
      "HostId": "h-07879acf49EXAMPLE",
      "Tags": [
        {
          "Value": "production",
          "Key": "purpose"
        }
      ],
      "HostProperties": {
        "Cores": 48,
        "TotalVCpus": 96,
        "InstanceType": "m5.large",
        "Sockets": 2
      },
      "Instances": [],
      "State": "available",
    }
  ]
}
```

```
    "AvailabilityZone": "eu-west-1a",
    "AvailableCapacity": {
      "AvailableInstanceCapacity": [
        {
          "AvailableCapacity": 48,
          "InstanceType": "m5.large",
          "TotalCapacity": 48
        }
      ],
      "AvailableVCpus": 96
    },
    "HostRecovery": "on",
    "AllocationTime": "2019-08-19T08:57:44.000Z",
    "AutoPlacement": "off"
  }
]
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[查看专用主机](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeHosts](#) 中的。

## describe-iam-instance-profile-associations

以下代码示例演示了如何使用 describe-iam-instance-profile-associations。

### AWS CLI

描述 IAM 实例配置文件关联

本示例描述了所有 IAM 实例配置文件关联。

命令:

```
aws ec2 describe-iam-instance-profile-associations
```

输出:

```
{
  "IamInstanceProfileAssociations": [
    {
      "InstanceId": "i-09eb09efa73ec1dee",
      "State": "associated",
```

```

    "AssociationId": "iip-assoc-0db249b1f25fa24b8",
    "IamInstanceProfile": {
      "Id": "AIPAJVQN4F5WVLGCJDRGM",
      "Arn": "arn:aws:iam::123456789012:instance-profile/admin-role"
    }
  },
  {
    "InstanceId": "i-0402909a2f4dff14",
    "State": "associating",
    "AssociationId": "iip-assoc-0d1ec06278d29f44a",
    "IamInstanceProfile": {
      "Id": "AGJAJVQN4F5WVLGCJABCM",
      "Arn": "arn:aws:iam::123456789012:instance-profile/user1-role"
    }
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeIamInstanceProfileAssociations](#)中的。

## describe-id-format

以下代码示例演示了如何使用 describe-id-format。

### AWS CLI

示例 1：描述资源的 ID 格式

以下describe-id-format示例描述了安全组的 ID 格式。

```

aws ec2 describe-id-format \
  --resource security-group

```

在以下示例输出中，该Deadline值表示该资源类型从短 ID 格式永久切换到长 ID 格式的最后期限已于 2018 年 8 月 15 日 00:00 UTC 到期。

```

{
  "Statuses": [
    {
      "Deadline": "2018-08-15T00:00:00.000Z",
      "Resource": "security-group",
      "UseLongIds": true
    }
  ]
}

```

```
    }  
  ]  
}
```

## 示例 2：描述所有资源的 ID 格式

以下 `describe-id-format` 示例描述了所有资源类型的 ID 格式。所有支持短 ID 格式的资源类型均已切换为使用长 ID 格式。

```
aws ec2 describe-id-format
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeIdFormat](#) 中的。

## describe-identity-id-format

以下代码示例演示了如何使用 `describe-identity-id-format`。

### AWS CLI

#### 描述 IAM 角色的 ID 格式

以下 `describe-identity-id-format` 示例描述了您的 AWS 账户中的 IAM 角色 `EC2Role` 创建的实例所接收的 ID 格式。

```
aws ec2 describe-identity-id-format \  
  --principal-arn arn:aws:iam::123456789012:role/my-iam-role \  
  --resource instance
```

以下输出表明由此角色创建的实例会收到长 ID 格式的 ID。

```
{  
  "Statuses": [  
    {  
      "Deadline": "2016-12-15T00:00:00Z",  
      "Resource": "instance",  
      "UseLongIds": true  
    }  
  ]  
}
```

#### 描述 IAM 用户的 ID 格式

以下describe-identity-id-format示例描述了 IAM 用户AdminUser在您的 AWS 账户中创建的快照所接收的 ID 格式。

```
aws ec2 describe-identity-id-format \  
  --principal-arn arn:aws:iam::123456789012:user/AdminUser \  
  --resource snapshot
```

输出表明该用户创建的快照会收到长 ID 格式的 ID。

```
{  
  "Statuses": [  
    {  
      "Deadline": "2016-12-15T00:00:00Z",  
      "Resource": "snapshot",  
      "UseLongIds": true  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeIdentityIdFormat](#)中的。

## describe-image-attribute

以下代码示例演示了如何使用 describe-image-attribute。

### AWS CLI

描述 AMI 的启动权限

此示例描述了指定 AMI 的启动权限。

命令:

```
aws ec2 describe-image-attribute --image-id ami-5731123e --attribute  
  launchPermission
```

输出:

```
{  
  "LaunchPermissions": [  
    {
```

```
        "UserId": "123456789012"
      }
    ],
    "ImageId": "ami-5731123e",
  }
}
```

### 描述 AMI 的产品代码

此示例描述了指定 AMI 的产品代码。请注意，此 AMI 没有产品代码。

命令:

```
aws ec2 describe-image-attribute --image-id ami-5731123e --attribute productCodes
```

输出:

```
{
  "ProductCodes": [],
  "ImageId": "ami-5731123e",
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeImageAttribute](#)中的。

## describe-images

以下代码示例演示了如何使用 describe-images。

### AWS CLI

示例 1：描述 AMI

以下 describe-images 示例描述了指定区域中的指定 AMI。

```
aws ec2 describe-images \
  --region us-east-1 \
  --image-ids ami-1234567890EXAMPLE
```

输出:

```
{
  "Images": [
    {
```



```

    "VirtualizationType": "hvm",
    "Description": "Provided by Red Hat, Inc.",
    "PlatformDetails": "Red Hat Enterprise Linux",
    "EnaSupport": true,
    "Hypervisor": "xen",
    "State": "available",
    "SriovNetSupport": "simple",
    "ImageId": "ami-1234567890EXAMPLE",
    "UsageOperation": "RunInstances:0010",
    "BlockDeviceMappings": [
      {
        "DeviceName": "/dev/sda1",
        "Ebs": {
          "SnapshotId": "snap-111222333444aaabb",
          "DeleteOnTermination": true,
          "VolumeType": "gp2",
          "VolumeSize": 10,
          "Encrypted": false
        }
      }
    ],
    "Architecture": "x86_64",
    "ImageLocation": "123456789012/RHEL-8.0.0_HVM-20190618-x86_64-1-Hourly2-
GP2",
    "RootDeviceType": "ebs",
    "OwnerId": "123456789012",
    "RootDeviceName": "/dev/sda1",
    "CreationDate": "2019-05-10T13:17:12.000Z",
    "Public": true,
    "ImageType": "machine",
    "Name": "RHEL-8.0.0_HVM-20190618-x86_64-1-Hourly2-GP2"
  }
]
}

```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[亚马逊机器映像 \(AMI\)](#)。

#### 示例 2：根据筛选器描述 AMI

以下 describe-images 示例描述了由 Amazon 提供，并受 Amazon EBS 支持的 Windows AMI。

```
aws ec2 describe-images \
```

```
--owners amazon \  
--filters "Name=platform,Values=windows" "Name=root-device-type,Values=efs"
```

有关 `describe-images` 的输出示例，请参阅示例 1。

有关使用筛选器的其他示例，请参阅《Amazon EC2 用户指南》中的[列出和筛选资源](#)。

### 示例 3：根据标签描述 AMI

以下 `describe-images` 示例描述了带有标签 `Type=Custom` 的所有 AMI。示例使用 `--query` 参数仅显示 AMI ID。

```
aws ec2 describe-images \  
  --filters "Name=tag:Type,Values=Custom" \  
  --query 'Images[*].[ImageId]' \  
  --output text
```

输出：

```
ami-1234567890EXAMPLE  
ami-0abcdef1234567890
```

有关使用标签筛选器的更多示例，请参阅《Amazon EC2 用户指南》中的[使用标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeImages](#) 中的。

## describe-import-image-tasks

以下代码示例演示了如何使用 `describe-import-image-tasks`。

### AWS CLI

#### 监视导入图像任务

以下 `describe-import-image-tasks` 示例检查指定导入映像任务的状态。

```
aws ec2 describe-import-image-tasks \  
  --import-task-ids import-ami-1234567890abcdef0
```

正在进行的导入图像任务的输出。

```
{
  "ImportImageTasks": [
    {
      "ImportTaskId": "import-ami-1234567890abcdef0",
      "Progress": "28",
      "SnapshotDetails": [
        {
          "DiskImageSize": 705638400.0,
          "Format": "ova",
          "Status": "completed",
          "UserBucket": {
            "S3Bucket": "my-import-bucket",
            "S3Key": "vms/my-server-vm.ova"
          }
        }
      ],
      "Status": "active",
      "StatusMessage": "converting"
    }
  ]
}
```

已完成的导入图像任务的输出。生成的 AMI 的 ID 由提供 ImageId。

```
{
  "ImportImageTasks": [
    {
      "ImportTaskId": "import-ami-1234567890abcdef0",
      "ImageId": "ami-1234567890abcdef0",
      "SnapshotDetails": [
        {
          "DiskImageSize": 705638400.0,
          "Format": "ova",
          "SnapshotId": "snap-1234567890abcdef0"
          "Status": "completed",
          "UserBucket": {
            "S3Bucket": "my-import-bucket",
            "S3Key": "vms/my-server-vm.ova"
          }
        }
      ],
      "Status": "completed"
    }
  ]
}
```

```
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeImportImageTasks](#)中的。

## describe-import-snapshot-tasks

以下代码示例演示了如何使用 `describe-import-snapshot-tasks`。

### AWS CLI

#### 监视导入快照任务

以下`describe-import-snapshot-tasks`示例检查指定导入快照任务的状态。

```
aws ec2 describe-import-snapshot-tasks \
  --import-task-ids import-snap-1234567890abcdef0
```

正在进行的导入快照任务的输出：

```
{
  "ImportSnapshotTasks": [
    {
      "Description": "My server VMDK",
      "ImportTaskId": "import-snap-1234567890abcdef0",
      "SnapshotTaskDetail": {
        "Description": "My server VMDK",
        "DiskImageSize": "705638400.0",
        "Format": "VMDK",
        "Progress": "42",
        "Status": "active",
        "StatusMessage": "downloading/converting",
        "UserBucket": {
          "S3Bucket": "my-import-bucket",
          "S3Key": "vms/my-server-vm.vmdk"
        }
      }
    }
  ]
}
```

已完成的导入快照任务的输出。生成的快照的 ID 由提供`SnapshotId`。

```
{
  "ImportSnapshotTasks": [
    {
      "Description": "My server VMDK",
      "ImportTaskId": "import-snap-1234567890abcdef0",
      "SnapshotTaskDetail": {
        "Description": "My server VMDK",
        "DiskImageSize": "705638400.0",
        "Format": "VMDK",
        "SnapshotId": "snap-1234567890abcdef0"
        "Status": "completed",
        "UserBucket": {
          "S3Bucket": "my-import-bucket",
          "S3Key": "vms/my-server-vm.vmdk"
        }
      }
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeImportSnapshotTasks](#)中的。

## describe-instance-attribute

以下代码示例演示了如何使用 describe-instance-attribute。

### AWS CLI

#### 描述实例类型

此示例描述了指定实例的实例类型。

命令:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --attribute
instanceType
```

输出:

```
{
  "InstanceId": "i-1234567890abcdef0"
  "InstanceType": {
```

```
    "Value": "t1.micro"  
  }  
}
```

描述该 `disableApiTermination` 属性

此示例描述了指定实例的 `disableApiTermination` 属性。

命令:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --attribute  
disableApiTermination
```

输出 :

```
{  
  "InstanceId": "i-1234567890abcdef0"  
  "DisableApiTermination": {  
    "Value": "false"  
  }  
}
```

描述实例的块储存设备映射

此示例描述了指定实例的 `blockDeviceMapping` 属性。

命令:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --attribute  
blockDeviceMapping
```

输出 :

```
{  
  "InstanceId": "i-1234567890abcdef0"  
  "BlockDeviceMappings": [  
    {  
      "DeviceName": "/dev/sda1",  
      "Ebs": {  
        "Status": "attached",  
        "DeleteOnTermination": true,  
        "VolumeSize": 8,  
        "VolumeType": "gp2"  
      }  
    }  
  ]  
}
```

```

        "VolumeId": "vol-049df61146c4d7901",
        "AttachTime": "2013-05-17T22:42:34.000Z"
    }
},
{
    "DeviceName": "/dev/sdf",
    "Ebs": {
        "Status": "attached",
        "DeleteOnTermination": false,
        "VolumeId": "vol-049df61146c4d7901",
        "AttachTime": "2013-09-10T23:07:00.000Z"
    }
}
],
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInstanceAttribute](#)中的。

## describe-instance-connect-endpoints

以下代码示例演示了如何使用 describe-instance-connect-endpoints。

### AWS CLI

描述 EC2 实例 Connect 终端节点

以下describe-instance-connect-endpoints示例描述了指定的 EC2 Instance Connect 终端节点。

```

aws ec2 describe-instance-connect-endpoints \
  --region us-east-1 \
  --instance-connect-endpoint-ids eice-0123456789example

```

输出：

```

{
  "InstanceConnectEndpoints": [
    {
      "OwnerId": "111111111111",
      "InstanceConnectEndpointId": "eice-0123456789example",
      "InstanceConnectEndpointArn": "arn:aws:ec2:us-
east-1:111111111111:instance-connect-endpoint/eice-0123456789example",

```

```

        "State": "create-complete",
        "StateMessage": "",
        "DnsName": "eice-0123456789example.b67b86ba.ec2-instance-connect-
endpoint.us-east-1.amazonaws.com",
        "NetworkInterfaceIds": [
            "eni-0123456789example"
        ],
        "VpcId": "vpc-0123abcd",
        "AvailabilityZone": "us-east-1d",
        "CreatedAt": "2023-02-07T12:05:37+00:00",
        "SubnetId": "subnet-0123abcd",
        "Tags": []
    }
]
}

```

有关更多信息，请参阅 Amazon EC2 用户指南中的创建 EC2 Instance [Connect 终端节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeInstanceConnectEndpoints](#) 中的。

## describe-instance-credit-specifications

以下代码示例演示了如何使用 describe-instance-credit-specifications。

### AWS CLI

描述一个或多个实例的 CPU 使用率积分选项

以下 describe-instance-credit-specifications 示例描述了指定实例的 CPU 积分选项。

```
aws ec2 describe-instance-credit-specifications \
  --instance-ids i-1234567890abcdef0
```

输出：

```
{
  "InstanceCreditSpecifications": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "CpuCredits": "unlimited"
    }
  ]
}
```



```
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的使用可[突发性能实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInstanceCreditSpecifications](#)中的。

## describe-instance-event-notification-attributes

以下代码示例演示了如何使用 describe-instance-event-notification-attributes。

### AWS CLI

描述计划事件通知的标签

以下describe-instance-event-notification-attributes示例描述了要在计划的事件通知中显示的标签。

```
aws ec2 describe-instance-event-notification-attributes
```

输出：

```
{
  "InstanceTagAttribute": {
    "InstanceTagKeys": [],
    "IncludeAllTagsOfInstance": true
  }
}
```

有关更多信息，请参阅适用于[Linux 实例的 Amazon 弹性计算云用户指南中的实例计划事件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInstanceEventNotificationAttributes](#)中的。

## describe-instance-event-windows

以下代码示例演示了如何使用 describe-instance-event-windows。

### AWS CLI

示例 1：描述所有事件窗口

以下describe-instance-event-windows示例描述了指定区域中的所有事件窗口。

```
aws ec2 describe-instance-event-windows \  
  --region us-east-1
```

输出：

```
{  
  "InstanceEventWindows": [  
    {  
      "InstanceEventWindowId": "iew-0abcdef1234567890",  
      "Name": "myEventWindowName",  
      "CronExpression": "* 21-23 * * 2,3",  
      "AssociationTarget": {  
        "InstanceIds": [  
          "i-1234567890abcdef0",  
          "i-0598c7d356eba48d7"  
        ],  
        "Tags": [],  
        "DedicatedHostIds": []  
      },  
      "State": "active",  
      "Tags": []  
    },  
    ...  
  ],  
  "NextToken": "9d624e0c-388b-4862-a31e-a85c64fc1d4a"  
}
```

示例 2：描述特定的事件窗口

以下describe-instance-event-windows示例通过使用instance-event-window参数描述特定事件窗口来描述特定事件。

```
aws ec2 describe-instance-event-windows \  
  --region us-east-1 \  
  --instance-event-window-ids iew-0abcdef1234567890
```

输出：

```
{
```

```

"InstanceEventWindows": [
  {
    "InstanceEventWindowId": "iew-0abcdef1234567890",
    "Name": "myEventWindowName",
    "CronExpression": "* 21-23 * * 2,3",
    "AssociationTarget": {
      "InstanceIds": [
        "i-1234567890abcdef0",
        "i-0598c7d356eba48d7"
      ],
      "Tags": [],
      "DedicatedHostIds": []
    },
    "State": "active",
    "Tags": []
  }
]
}

```

### 示例 3：描述与一个或多个过滤器匹配的事件窗口

以下 `describe-instance-event-windows` 示例使用 `filter` 参数描述了与一个或多个过滤器匹配的事件窗口。 `instance-id` 过滤器用于描述与指定实例关联的所有事件窗口。使用筛选器时，它会进行直接匹配。但是， `instance-id` 筛选器不同。如果与实例 ID 没有直接匹配，则它会退回到与事件窗口的间接关联，例如实例的标签或专用主机 ID（如果实例是专用主机）。

```

aws ec2 describe-instance-event-windows \
  --region us-east-1 \
  --filters Name=instance-id,Values=i-1234567890abcdef0 \
  --max-results 100 \
  --next-token <next-token-value>

```

输出：

```

{
  "InstanceEventWindows": [
    {
      "InstanceEventWindowId": "iew-0dbc0adb66f235982",
      "TimeRanges": [
        {
          "StartWeekDay": "sunday",
          "StartHour": 2,
          "EndWeekDay": "sunday",

```

```

        "EndHour": 8
      }
    ],
    "Name": "myEventWindowName",
    "AssociationTarget": {
      "InstanceIds": [],
      "Tags": [],
      "DedicatedHostIds": [
        "h-0140d9a7ecbd102dd"
      ]
    },
    "State": "active",
    "Tags": []
  }
]
}

```

在示例输出中，实例位于与事件窗口关联的专用主机上。

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中的[注意事项](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeInstanceEventWindows](#)中的。

## describe-instance-status

以下代码示例演示了如何使用 describe-instance-status。

### AWS CLI

描述实例的状态

以下 describe-instance-status 示例描述了指定实例的当前状态。

```
aws ec2 describe-instance-status \
  --instance-ids i-1234567890abcdef0
```

输出：

```
{
  "InstanceStatuses": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "InstanceState": {
```

```
        "Code": 16,
        "Name": "running"
    },
    "AvailabilityZone": "us-east-1d",
    "SystemStatus": {
        "Status": "ok",
        "Details": [
            {
                "Status": "passed",
                "Name": "reachability"
            }
        ]
    },
    "InstanceState": {
        "Status": "ok",
        "Details": [
            {
                "Status": "passed",
                "Name": "reachability"
            }
        ]
    }
}
]
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[监控实例状态](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInstanceStatus](#)中的。

## describe-instance-topology

以下代码示例演示了如何使用 describe-instance-topology。

### AWS CLI

描述所有实例的实例拓扑

以下describe-instance-topology示例描述了与该命令支持的实例类型相匹配的所有实例的拓扑。

```
aws ec2 describe-instance-topology \
  --region us-west-2
```

输出：

```
{
  "Instances": [
    {
      "InstanceId": "i-1111111111example",
      "InstanceType": "p4d.24xlarge",
      "GroupName": "my-m1-cpg",
      "NetworkNodes": [
        "nn-1111111111example",
        "nn-2222222222example",
        "nn-3333333333example"
      ],
      "ZoneId": "usw2-az2",
      "AvailabilityZone": "us-west-2a"
    },
    {
      "InstanceId": "i-2222222222example",
      "InstanceType": "p4d.24xlarge",
      "NetworkNodes": [
        "nn-1111111111example",
        "nn-2222222222example",
        "nn-3333333333example"
      ],
      "ZoneId": "usw2-az2",
      "AvailabilityZone": "us-west-2a"
    },
    {
      "InstanceId": "i-3333333333example",
      "InstanceType": "trn1.32xlarge",
      "NetworkNodes": [
        "nn-1212121212example",
        "nn-1211122211example",
        "nn-1311133311example"
      ],
      "ZoneId": "usw2-az4",
      "AvailabilityZone": "us-west-2d"
    },
    {
      "InstanceId": "i-4444444444example",
      "InstanceType": "trn1.2xlarge",
      "NetworkNodes": [
        "nn-1111111111example",
        "nn-5434334334example",

```

```
        "nn-1235301234example"
      ],
      "ZoneId": "usw2-az2",
      "AvailabilityZone": "us-west-2a"
    }
  ],
  "NextToken": "SomeEncryptedToken"
}
```

有关更多信息，包括更多示例，请参阅 [Amazon EC2 用户指南中的 Amazon EC2 实例拓扑](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeInstanceTopology](#) 中的。

## describe-instance-type-offerings

以下代码示例演示了如何使用 `describe-instance-type-offerings`。

### AWS CLI

示例 1：列出某个地区提供的实例类型

以下 `describe-instance-type-offerings` 示例列出了配置为 AWS CLI 默认区域的区域中提供的实例类型。

```
aws ec2 describe-instance-type-offerings
```

要列出不同区域提供的实例类型，请使用 `--region` 参数指定区域。

```
aws ec2 describe-instance-type-offerings \
  --region us-east-2
```

输出：

```
{
  "InstanceTypeOfferings": [
    {
      "InstanceType": "m5.2xlarge",
      "LocationType": "region",
      "Location": "us-east-2"
    },
    {
      "InstanceType": "t3.micro",
```

```
        "LocationType": "region",
        "Location": "us-east-2"
    },
    ...
]
}
```

### 示例 2：列出可用区中提供的实例类型

以下describe-instance-type-offerings示例列出了指定可用区中提供的实例类型。可用区必须位于指定的区域。

```
aws ec2 describe-instance-type-offerings \
  --location-type availability-zone \
  --filters Name=location,Values=us-east-2a \
  --region us-east-2
```

### 示例 3：检查是否支持某个实例类型

以下describe-instance-type-offerings命令指示指定区域是否支持该c5.xlarge实例类型。

```
aws ec2 describe-instance-type-offerings \
  --filters Name=instance-type,Values=c5.xlarge \
  --region us-east-2
```

以下describe-instance-type-offerings示例列出了指定区域支持的所有 C5 实例类型。

```
aws ec2 describe-instance-type-offerings \
  --filters Name=instance-type,Values=c5* \
  --query "InstanceTypeOfferings[].InstanceType" \
  --region us-east-2
```

输出：

```
[
  "c5d.12xlarge",
  "c5d.9xlarge",
  "c5n.xlarge",
  "c5.xlarge",
  "c5d.metal",
  "c5n.metal",
```



```
"c5.large",  
"c5d.2xlarge",  
"c5n.4xlarge",  
"c5.2xlarge",  
"c5n.large",  
"c5n.9xlarge",  
"c5d.large",  
"c5.18xlarge",  
"c5d.18xlarge",  
"c5.12xlarge",  
"c5n.18xlarge",  
"c5.metal",  
"c5d.4xlarge",  
"c5.24xlarge",  
"c5d.xlarge",  
"c5n.2xlarge",  
"c5d.24xlarge",  
"c5.9xlarge",  
"c5.4xlarge"
```

```
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInstanceTypeOfferings](#)中的。

## describe-instance-types

以下代码示例演示了如何使用 describe-instance-types。

### AWS CLI

#### 示例 1：描述实例类型

以下 describe-instance-types 示例显示指定实例类型的详细信息。

```
aws ec2 describe-instance-types \  
  --instance-types t2.micro
```

输出：

```
{  
  "InstanceTypes": [  
    {  
      "InstanceType": "t2.micro",  
      "CurrentGeneration": true,
```

```
"FreeTierEligible": true,
"SupportedUsageClasses": [
  "on-demand",
  "spot"
],
"SupportedRootDeviceTypes": [
  "ebs"
],
"BareMetal": false,
"Hypervisor": "xen",
"ProcessorInfo": {
  "SupportedArchitectures": [
    "i386",
    "x86_64"
  ],
  "SustainedClockSpeedInGhz": 2.5
},
"VCpuInfo": {
  "DefaultVCpus": 1,
  "DefaultCores": 1,
  "DefaultThreadsPerCore": 1,
  "ValidCores": [
    1
  ],
  "ValidThreadsPerCore": [
    1
  ]
},
"MemoryInfo": {
  "SizeInMiB": 1024
},
"InstanceStorageSupported": false,
"EbsInfo": {
  "EbsOptimizedSupport": "unsupported",
  "EncryptionSupport": "supported"
},
"NetworkInfo": {
  "NetworkPerformance": "Low to Moderate",
  "MaximumNetworkInterfaces": 2,
  "Ipv4AddressesPerInterface": 2,
  "Ipv6AddressesPerInterface": 2,
  "Ipv6Supported": true,
  "EnaSupport": "unsupported"
},
```

```
    "PlacementGroupInfo": {
      "SupportedStrategies": [
        "partition",
        "spread"
      ]
    },
    "HibernationSupported": false,
    "BurstablePerformanceSupported": true,
    "DedicatedHostsSupported": false,
    "AutoRecoverySupported": true
  }
]
```

有关更多信息，请参阅适用于 Linux 实例的 Amazon 弹性计算云用户指南中的实例[类型](#)。

### 示例 2：筛选可用的实例类型

可以指定筛选器，将结果范围限定为具有特定特征的实例类型。以下 `describe-instance-types` 示例列出支持休眠的实例类型。

```
aws ec2 describe-instance-types \
  --filters Name=hibernation-supported,Values=true --query
  'InstanceTypes[*].InstanceType'
```

输出：

```
[
  "m5.8xlarge",
  "r3.large",
  "c3.8xlarge",
  "r5.large",
  "m4.4xlarge",
  "c4.large",
  "m5.xlarge",
  "m4.xlarge",
  "c3.large",
  "c4.8xlarge",
  "c4.4xlarge",
  "c5.xlarge",
  "c5.12xlarge",
  "r5.4xlarge",
  "c5.4xlarge"
```

```
]
```

有关更多信息，请参阅适用于 Linux 实例的 Amazon 弹性计算云用户指南中的实例[类型](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeInstanceTypes](#) 中的。

## describe-instances

以下代码示例演示了如何使用 describe-instances。

### AWS CLI

#### 示例 1：描述实例

以下 describe-instances 示例描述了指定的实例。

```
aws ec2 describe-instances \  
  --instance-ids i-1234567890abcdef0
```

输出：

```
{  
  "Reservations": [  
    {  
      "Groups": [],  
      "Instances": [  
        {  
          "AmiLaunchIndex": 0,  
          "ImageId": "ami-0abcdef1234567890",  
          "InstanceId": "i-1234567890abcdef0",  
          "InstanceType": "t3.nano",  
          "KeyName": "my-key-pair",  
          "LaunchTime": "2022-11-15T10:48:59+00:00",  
          "Monitoring": {  
            "State": "disabled"  
          },  
          "Placement": {  
            "AvailabilityZone": "us-east-2a",  
            "GroupName": "",  
            "Tenancy": "default"  
          },  
          "PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",  
          "PrivateIpAddress": "10-0-0-157",  
        }  
      ]  
    }  
  ]  
}
```

```
    "ProductCodes": [],
    "PublicDnsName": "ec2-34-253-223-13.us-
east-2.compute.amazonaws.com",
    "PublicIpAddress": "34.253.223.13",
    "State": {
      "Code": 16,
      "Name": "running"
    },
    "StateTransitionReason": "",
    "SubnetId": "subnet-04a636d18e83cfacb",
    "VpcId": "vpc-1234567890abcdef0",
    "Architecture": "x86_64",
    "BlockDeviceMappings": [
      {
        "DeviceName": "/dev/xvda",
        "Ebs": {
          "AttachTime": "2022-11-15T10:49:00+00:00",
          "DeleteOnTermination": true,
          "Status": "attached",
          "VolumeId": "vol-02e6ccdca7de29cf2"
        }
      }
    ],
    "ClientToken": "1234abcd-1234-abcd-1234-d46a8903e9bc",
    "EbsOptimized": true,
    "EnaSupport": true,
    "Hypervisor": "xen",
    "IamInstanceProfile": {
      "Arn": "arn:aws:iam::111111111111:instance-profile/
AmazonSSMRoleForInstancesQuickSetup",
      "Id": "11111111111111111111111111111111"
    },
    "NetworkInterfaces": [
      {
        "Association": {
          "IpOwnerId": "amazon",
          "PublicDnsName": "ec2-34-253-223-13.us-
east-2.compute.amazonaws.com",
          "PublicIp": "34.253.223.13"
        },
        "Attachment": {
          "AttachTime": "2022-11-15T10:48:59+00:00",
          "AttachmentId": "eni-attach-1234567890abcdefg",
          "DeleteOnTermination": true,
```

```

        "DeviceIndex": 0,
        "Status": "attached",
        "NetworkCardIndex": 0
    },
    "Description": "",
    "Groups": [
        {
            "GroupName": "launch-wizard-146",
            "GroupId": "sg-1234567890abcdefg"
        }
    ],
    "Ipv6Addresses": [],
    "MacAddress": "00:11:22:33:44:55",
    "NetworkInterfaceId": "eni-1234567890abcdefg",
    "OwnerId": "104024344472",
    "PrivateDnsName": "ip-10-0-0-157.us-
east-2.compute.internal",
    "PrivateIpAddress": "10-0-0-157",
    "PrivateIpAddresses": [
        {
            "Association": {
                "IpOwnerId": "amazon",
                "PublicDnsName": "ec2-34-253-223-13.us-
east-2.compute.amazonaws.com",
                "PublicIp": "34.253.223.13"
            },
            "Primary": true,
            "PrivateDnsName": "ip-10-0-0-157.us-
east-2.compute.internal",
            "PrivateIpAddress": "10-0-0-157"
        }
    ],
    "SourceDestCheck": true,
    "Status": "in-use",
    "SubnetId": "subnet-1234567890abcdefg",
    "VpcId": "vpc-1234567890abcdefg",
    "InterfaceType": "interface"
    }
],
"RootDeviceName": "/dev/xvda",
"RootDeviceType": "ebs",
"SecurityGroups": [
    {
        "GroupName": "launch-wizard-146",

```

```
        "GroupId": "sg-1234567890abcdefg"
      }
    ],
    "SourceDestCheck": true,
    "Tags": [
      {
        "Key": "Name",
        "Value": "my-instance"
      }
    ],
    "VirtualizationType": "hvm",
    "CpuOptions": {
      "CoreCount": 1,
      "ThreadsPerCore": 2
    },
    "CapacityReservationSpecification": {
      "CapacityReservationPreference": "open"
    },
    "HibernationOptions": {
      "Configured": false
    },
    "MetadataOptions": {
      "State": "applied",
      "HttpTokens": "optional",
      "HttpPutResponseHopLimit": 1,
      "HttpEndpoint": "enabled",
      "HttpProtocolIpv6": "disabled",
      "InstanceMetadataTags": "enabled"
    },
    "EnclaveOptions": {
      "Enabled": false
    },
    "PlatformDetails": "Linux/UNIX",
    "UsageOperation": "RunInstances",
    "UsageOperationUpdateTime": "2022-11-15T10:48:59+00:00",
    "PrivateDnsNameOptions": {
      "HostnameType": "ip-name",
      "EnableResourceNameDnsARecord": true,
      "EnableResourceNameDnsAAAARecord": false
    },
    "MaintenanceOptions": {
      "AutoRecovery": "default"
    }
  }
}
```

```
    ],  
    "OwnerId": "111111111111",  
    "ReservationId": "r-1234567890abcdefg"  
  }  
]  
}
```

### 示例 2：筛选具有指定类型的实例

以下 `describe-instances` 示例使用筛选器，将结果范围限定为指定类型的实例。

```
aws ec2 describe-instances \  
  --filters Name=instance-type,Values=m5.large
```

有关示例输出，请参阅示例 1。

有关更多信息，请参阅《Amazon EC2 用户指南》中的[使用 CLI 列出和筛选](#)。

### 示例 3：筛选具有指定类型和可用区的实例

以下 `describe-instances` 示例使用多个筛选器，将结果范围限定为同样位于指定可用区且具有指定类型的实例。

```
aws ec2 describe-instances \  
  --filters Name=instance-type,Values=t2.micro,t3.micro Name=availability-  
zone,Values=us-east-2c
```

有关示例输出，请参阅示例 1。

### 示例 4：使用 JSON 文件筛选具有指定类型和可用区的实例

以下 `describe-instances` 示例使用 JSON 输入文件执行与上一个示例相同的筛选。当筛选器变得更加复杂时，可以更容易地在 JSON 文件中加以指定。

```
aws ec2 describe-instances \  
  --filters file://filters.json
```

`filters.json` 的内容：

```
[
```



```
{
  "Name": "instance-type",
  "Values": ["t2.micro", "t3.micro"]
},
{
  "Name": "availability-zone",
  "Values": ["us-east-2c"]
}
]
```

有关示例输出，请参阅示例 1。

#### 示例 5：筛选具有指定 Owner 标签的实例

以下 `describe-instances` 示例使用标签筛选器，将结果范围限定为具有指定标签键（Owner）标签的实例，无论标签值如何。

```
aws ec2 describe-instances \
  --filters "Name=tag-key,Values=Owner"
```

有关示例输出，请参阅示例 1。

#### 示例 6：筛选具有指定 my-team 标签值的实例

以下 `describe-instances` 示例使用标签筛选器，将结果范围限定为具有指定标签值（my-team）标签的实例，无论标签键如何。

```
aws ec2 describe-instances \
  --filters "Name=tag-value,Values=my-team"
```

有关示例输出，请参阅示例 1。

#### 示例 7：筛选具有指定 Owner 标签和 my-team 值的实例

以下 `describe-instances` 示例使用标签筛选器，将结果范围限定为具有指定标签值（Owner=my-team）的实例。

```
aws ec2 describe-instances \
  --filters "Name=tag:Owner,Values=my-team"
```

有关示例输出，请参阅示例 1。

### 示例 8：仅显示所有实例的实例和子网 ID

以下 `describe-instances` 示例使用 `--query` 参数，以 JSON 格式仅显示所有实例的实例和子网 ID。

Linux 和 macOS：

```
aws ec2 describe-instances \  
  --query 'Reservations[*].Instances[*].{Instance:InstanceId,Subnet:SubnetId}' \  
  --output json
```

Windows:

```
aws ec2 describe-instances ^  
  --query "Reservations[*].Instances[*].{Instance:InstanceId,Subnet:SubnetId}" ^  
  --output json
```

输出：

```
[  
  {  
    "Instance": "i-057750d42936e468a",  
    "Subnet": "subnet-069beee9b12030077"  
  },  
  {  
    "Instance": "i-001efd250faaa6ffa",  
    "Subnet": "subnet-0b715c6b7db68927a"  
  },  
  {  
    "Instance": "i-027552a73f021f3bd",  
    "Subnet": "subnet-0250c25a1f4e15235"  
  }  
  ...  
]
```

### 示例 9：筛选指定类型的实例并仅显示其实例 ID

以下 `describe-instances` 示例使用筛选器，将结果范围限定为指定类型的实例，并使用 `--query` 参数仅显示实例 ID。

```
aws ec2 describe-instances \  
  --filters "Name=instance-type,Values=t2.micro" \  
  --query 'Instances[*].InstanceId' \  
  --output json
```

```
--filters "Name=instance-type,Values=t2.micro" \
--query "Reservations[*].Instances[*].[InstanceId]" \
--output text
```

输出：

```
i-031c0dc19de2fb70c
i-00d8bfff789a736b75
i-0b715c6b7db68927a
i-0626d4edd54f1286d
i-00b8ae04f9f99908e
i-0fc71c25d2374130c
```

示例 10：筛选指定类型的实例，并仅显示其实例 ID、可用区和指定的标签值

以下 describe-instances 示例以表格格式显示实例的实例 ID、可用区和 Name 标签值，这些实例有一个名为 tag-key 的标签。

Linux 和 macOS：

```
aws ec2 describe-instances \
  --filters Name=tag-key,Values=Name \
  --query 'Reservations[*].Instances[*].
{Instance:InstanceId,AZ:Placement.AvailabilityZone,Name:Tags[?Key==`Name`]|
[0].Value}' \
  --output table
```

Windows:

```
aws ec2 describe-instances ^
  --filters Name=tag-key,Values=Name ^
  --query "Reservations[*].Instances[*].
{Instance:InstanceId,AZ:Placement.AvailabilityZone,Name:Tags[?Key=='Name']|
[0].Value}" ^
  --output table
```

输出：

```
-----
|                               DescribeInstances                               |
```

```
+-----+-----+-----+
|      AZ      |      Instance      |      Name      |
+-----+-----+-----+
| us-east-2b  | i-057750d42936e468a | my-prod-server |
| us-east-2a  | i-001efd250faaa6ffa | test-server-1  |
| us-east-2a  | i-027552a73f021f3bd | test-server-2  |
+-----+-----+-----+
```

### 示例 11：描述分区置放群组中的实例

以下 `describe-instances` 示例描述了指定的实例。输出包括实例的置放信息，其中包含实例的置放群组名称和分区编号。

```
aws ec2 describe-instances \
  --instance-ids i-0123a456700123456 \
  --query "Reservations[*].Instances[*].Placement"
```

输出：

```
[
  [
    {
      "AvailabilityZone": "us-east-1c",
      "GroupName": "HDFS-Group-A",
      "PartitionNumber": 3,
      "Tenancy": "default"
    }
  ]
]
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[描述置放群组中的实例](#)。

### 示例 12：筛选具有指定置放群组和分区编号的实例

以下 `describe-instances` 示例将结果筛选为仅具有指定置放群组和分区编号的实例。

```
aws ec2 describe-instances \
  --filters "Name=placement-group-name,Values=HDFS-Group-A" "Name=placement-
partition-number,Values=7"
```

以下仅显示输出中的相关信息。

```
"Instances": [
  {
    "InstanceId": "i-0123a456700123456",
    "InstanceType": "r4.large",
    "Placement": {
      "AvailabilityZone": "us-east-1c",
      "GroupName": "HDFS-Group-A",
      "PartitionNumber": 7,
      "Tenancy": "default"
    }
  },
  {
    "InstanceId": "i-9876a543210987654",
    "InstanceType": "r4.large",
    "Placement": {
      "AvailabilityZone": "us-east-1c",
      "GroupName": "HDFS-Group-A",
      "PartitionNumber": 7,
      "Tenancy": "default"
    }
  }
],
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[描述置放群组中的实例](#)。

示例 13：筛选配置为允许从实例元数据访问标签的实例

以下 describe-instances 示例将结果筛选为，仅显示配置为允许从实例元数据访问实例标签的实例。

```
aws ec2 describe-instances \
  --filters "Name=metadata-options.instance-metadata-tags,Values=enabled" \
  --query "Reservations[*].Instances[*].InstanceId" \
  --output text
```

预期的输出如下所示。

```
i-1234567890abcdefg
i-abcdefg1234567890
i-11111111aaaaaaaaa
i-aaaaaaaaa111111111
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[使用实例元数据中的实例标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInstances](#)中的。

## describe-internet-gateways

以下代码示例演示了如何使用 describe-internet-gateways。

### AWS CLI

描述互联网网关

以下describe-internet-gateways示例描述了指定的互联网网关。

```
aws ec2 describe-internet-gateways \
  --internet-gateway-ids igw-0d0fb496b3EXAMPLE
```

输出：

```
{
  "InternetGateways": [
    {
      "Attachments": [
        {
          "State": "available",
          "VpcId": "vpc-0a60eb65b4EXAMPLE"
        }
      ],
      "InternetGatewayId": "igw-0d0fb496b3EXAMPLE",
      "OwnerId": "123456789012",
      "Tags": [
        {
          "Key": "Name",
          "Value": "my-igw"
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的[互联网网关](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInternetGateways](#)中的。

## describe-ipam-pools

以下代码示例演示了如何使用 describe-ipam-pools。

### AWS CLI

查看 IPAM 池的详细信息

以下describe-ipam-pools示例显示了池的详细信息。

(Linux) :

```
aws ec2 describe-ipam-pools \  
  --filters Name=owner-id,Values=123456789012 Name=ipam-scope-id,Values=ipam-  
  scope-02fc38cd4c48e7d38
```

( 视窗 ) :

```
aws ec2 describe-ipam-pools ^  
  --filters Name=owner-id,Values=123456789012 Name=ipam-scope-id,Values=ipam-  
  scope-02fc38cd4c48e7d38
```

输出 :

```
{  
  "IpamPools": [  
    {  
      "OwnerId": "123456789012",  
      "IpamPoolId": "ipam-pool-02ec043a19bbe5d08",  
      "IpamPoolArn": "arn:aws:ec2::123456789012:ipam-pool/ipam-  
pool-02ec043a19bbe5d08",  
      "IpamScopeArn": "arn:aws:ec2::123456789012:ipam-scope/ipam-  
scope-02fc38cd4c48e7d38",  
      "IpamScopeType": "private",  
      "IpamArn": "arn:aws:ec2::123456789012:ipam/ipam-08440e7a3acde3908",  
      "IpamRegion": "us-east-1",  
      "Locale": "None",  
      "PoolDepth": 1,  
      "State": "create-complete",  
      "AutoImport": true,  
      "AddressFamily": "ipv4",  
      "AllocationMinNetmaskLength": 16,  
    }  
  ]  
}
```

```
    "AllocationMaxNetmaskLength": 26,
    "AllocationDefaultNetmaskLength": 24,
    "AllocationResourceTags": [
      {
        "Key": "Environment",
        "Value": "Preprod"
      }
    ],
    "Tags": [
      {
        "Key": "Name",
        "Value": "Preprod pool"
      }
    ]
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeIpamPools](#)中的。

## describe-ipam-resource-discoveries

以下代码示例演示了如何使用 `describe-ipam-resource-discoveries`。

### AWS CLI

#### 示例 1：查看资源发现的完整详情

在此示例中，您是一名委派的 IPAM 管理员，他想要创建资源发现并与其他 AWS 组织的 IPAM 管理员共享，以便管理员可以管理和监控组织中资源的 IP 地址。

在以下情况下，此示例可能很有用：

您尝试创建资源发现，但出现错误，提示您已达到 1 的上限。您意识到自己可能已经创建了资源发现，并想在自己的账户中查看它。IPAM 未发现您所在区域的资源。您要查看为资源 -- `operating-regions` 定义的，并确保已将正确的区域添加为运营区域，以便可以发现那里的资源。

以下 `describe-ipam-resource-discoveries` 示例列出了您的 AWS 账户中资源发现的详细信息。您可以为每个 AWS 区域进行一次资源发现。

```
aws ec2 describe-ipam-resource-discoveries \
```



```
--region us-east-1
```

输出：

```
{
  "IpamResourceDiscoveries": [
    {
      "OwnerId": "149977607591",
      "IpamResourceDiscoveryId": "ipam-res-disco-0f8bdee9067137c0d",
      "IpamResourceDiscoveryArn": "arn:aws:ec2::149977607591:ipam-resource-
discovery/ipam-res-disco-0f8bdee9067137c0d",
      "IpamResourceDiscoveryRegion": "us-east-1",
      "OperatingRegions": [
        {
          "RegionName": "us-east-1"
        }
      ],
      "IsDefault": false,
      "State": "create-complete",
      "Tags": []
    }
  ]
}
```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的将 IPA [M 与组织以外的账户集成](#)。

示例 2：仅查看资源发现 ID

以下describe-ipam-resource-discoveries示例列出了您 AWS 账户中资源发现的 ID。您可以为每个 AWS 区域进行一次资源发现。

```
aws ec2 describe-ipam-resource-discoveries \
  --query "IpamResourceDiscoveries[*].IpamResourceDiscoveryId" \
  --output text
```

输出：

```
ipam-res-disco-0481e39b242860333
```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的将 IPA [M 与组织以外的账户集成](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeIpamResourceDiscoveries](#)中的。

## describe-ipam-resource-discovery-associations

以下代码示例演示了如何使用 `describe-ipam-resource-discovery-associations`。

### AWS CLI

#### 查看与 IPAM 的所有资源发现关联

在此示例中，您是 IPAM 委托管理员，他已将资源发现与您的 IPAM 关联起来，以便将其他账户与您的 IPAM 集成。您已经注意到，您的 IPAM 未按预期在资源发现的操作区域中发现资源。您需要检查资源发现的状态和状态，以确保创建资源的帐户仍处于活动状态，并且资源发现仍在共享中。

`--region`必须是您的 IPAM 的主区域。

以下`describe-ipam-resource-discovery-associations`示例列出了您 AWS 账户中的资源发现关联。

```
aws ec2 describe-ipam-resource-discovery-associations \
  --region us-east-1
```

输出：

```
{
  "IpamResourceDiscoveryAssociations": [
    {
      "OwnerId": "320805250157",
      "IpamResourceDiscoveryAssociationId": "ipam-res-disco-
assoc-05e6b45eca5bf5cf7",
      "IpamResourceDiscoveryAssociationArn": "arn:aws:ec2::320805250157:ipam-
resource-discovery-association/ipam-res-disco-assoc-05e6b45eca5bf5cf7",
      "IpamResourceDiscoveryId": "ipam-res-disco-0f4ef577a9f37a162",
      "IpamId": "ipam-005f921c17ebd5107",
      "IpamArn": "arn:aws:ec2::320805250157:ipam/ipam-005f921c17ebd5107",
      "IpamRegion": "us-east-1",
      "IsDefault": true,
      "ResourceDiscoveryStatus": "active",
      "State": "associate-complete",
      "Tags": []
    }
  ],
}
```

```

    {
      "OwnerId": "149977607591",
      "IpamResourceDiscoveryAssociationId": "ipam-res-disco-
assoc-0dfd21ae189ab5f62",
      "IpamResourceDiscoveryAssociationArn": "arn:aws:ec2::149977607591:ipam-
resource-discovery-association/ipam-res-disco-assoc-0dfd21ae189ab5f62",
      "IpamResourceDiscoveryId": "ipam-res-disco-0365d2977fc1672fe",
      "IpamId": "ipam-005f921c17ebd5107",
      "IpamArn": "arn:aws:ec2::149977607591:ipam/ipam-005f921c17ebd5107",
      "IpamRegion": "us-east-1",
      "IsDefault": false,
      "ResourceDiscoveryStatus": "active",
      "State": "create-complete",
      "Tags": []
    }
  ]
}

```

在此示例中，运行此命令后，您会注意到发现了一个非默认资源（"IsDefault": false） that is ``"ResourceDiscoveryStatus": "not-found"和"State": "create-complete"。资源发现所有者的账户已关闭。在另一种情况下，如果您注意到即"ResourceDiscoveryStatus": "not-found"和"State": "associate-complete"，则表示发生了以下情况之一：

资源发现已被资源发现所有者删除。资源发现所有者取消了资源发现的共享。

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的将 IPA [M 与组织以外的账户集成](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeIpamResourceDiscoveryAssociations](#)中的。

## describe-ipam-scopes

以下代码示例演示了如何使用 describe-ipam-scopes。

### AWS CLI

查看 IPAM 作用域的详细信息

以下describe-ipam-scopes示例显示了作用域的详细信息。

```
aws ec2 describe-ipam-scopes \
```

```
--filters Name=owner-id,Values=123456789012 Name=ipam-  
id,Values=ipam-08440e7a3acde3908
```

输出：

```
{  
  "IpamScopes": [  
    {  
      "OwnerId": "123456789012",  
      "IpamScopeId": "ipam-scope-02fc38cd4c48e7d38",  
      "IpamScopeArn": "arn:aws:ec2::123456789012:ipam-scope/ipam-  
scope-02fc38cd4c48e7d38",  
      "IpamArn": "arn:aws:ec2::123456789012:ipam/ipam-08440e7a3acde3908",  
      "IpamRegion": "us-east-1",  
      "IpamScopeType": "private",  
      "IsDefault": true,  
      "PoolCount": 2,  
      "State": "create-complete",  
      "Tags": []  
    },  
    {  
      "OwnerId": "123456789012",  
      "IpamScopeId": "ipam-scope-0b9eed026396dbc16",  
      "IpamScopeArn": "arn:aws:ec2::123456789012:ipam-scope/ipam-  
scope-0b9eed026396dbc16",  
      "IpamArn": "arn:aws:ec2::123456789012:ipam/ipam-08440e7a3acde3908",  
      "IpamRegion": "us-east-1",  
      "IpamScopeType": "public",  
      "IsDefault": true,  
      "PoolCount": 0,  
      "State": "create-complete",  
      "Tags": []  
    },  
    {  
      "OwnerId": "123456789012",  
      "IpamScopeId": "ipam-scope-0f1aff29486355c22",  
      "IpamScopeArn": "arn:aws:ec2::123456789012:ipam-scope/ipam-  
scope-0f1aff29486355c22",  
      "IpamArn": "arn:aws:ec2::123456789012:ipam/ipam-08440e7a3acde3908",  
      "IpamRegion": "us-east-1",  
      "IpamScopeType": "private",  
      "IsDefault": false,  
      "Description": "Example description",  
    }  
  ]  
}
```

```

    "PoolCount": 0,
    "State": "create-complete",
    "Tags": [
      {
        "Key": "Name",
        "Value": "Example name value"
      }
    ]
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeIpamScopes](#)中的。

## describe-ipams

以下代码示例演示了如何使用 describe-ipams。

### AWS CLI

查看 IPAM 的详细信息

以下describe-ipams示例显示了 IPAM 的详细信息。

```

aws ec2 describe-ipams \
  --filters Name=owner-id,Values=123456789012

```

输出：

```

{
  "Ipams": [
    {
      "OwnerId": "123456789012",
      "IpamId": "ipam-08440e7a3acde3908",
      "IpamArn": "arn:aws:ec2::123456789012:ipam/ipam-08440e7a3acde3908",
      "IpamRegion": "us-east-1",
      "PublicDefaultScopeId": "ipam-scope-0b9eed026396dbc16",
      "PrivateDefaultScopeId": "ipam-scope-02fc38cd4c48e7d38",
      "ScopeCount": 3,
      "OperatingRegions": [
        {
          "RegionName": "us-east-1"
        }
      ],
    }
  ],
}

```

```
    {
      "RegionName": "us-east-2"
    },
    {
      "RegionName": "us-west-1"
    }
  ],
  "State": "create-complete",
  "Tags": [
    {
      "Key": "Name",
      "Value": "ExampleIPAM"
    }
  ]
}
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeIpams](#)中的。

## describe-ipv6-pools

以下代码示例演示了如何使用 `describe-ipv6-pools`。

### AWS CLI

描述您的 IPv6 地址池

以下`describe-ipv6-pools`示例显示了您的所有 IPv6 地址池的详细信息。

```
aws ec2 describe-ipv6-pools
```

输出：

```
{
  "Ipv6Pools": [
    {
      "PoolId": "ipv6pool-ec2-012345abc12345abc",
      "PoolCidrBlocks": [
        {
          "Cidr": "2001:db8:123::/48"
        }
      ]
    }
  ],
```

```
    "Tags": [
      {
        "Key": "pool-1",
        "Value": "public"
      }
    ]
  }
]
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [DescribeIpv6Pools](#)。

## describe-key-pairs

以下代码示例演示了如何使用 describe-key-pairs。

### AWS CLI

显示密钥对

以下 describe-key-pairs 示例显示有关指定密钥对的信息。

```
aws ec2 describe-key-pairs \
  --key-names my-key-pair
```

输出：

```
{
  "KeyPairs": [
    {
      "KeyPairId": "key-0b94643da6EXAMPLE",
      "KeyFingerprint":
"1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f",
      "KeyName": "my-key-pair",
      "KeyType": "rsa",
      "Tags": [],
      "CreateTime": "2022-05-27T21:51:16.000Z"
    }
  ]
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的 [描述公有密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeKeyPairs](#)中的。

## describe-launch-template-versions

以下代码示例演示了如何使用 describe-launch-template-versions。

### AWS CLI

描述启动模板版本

此示例描述了指定启动模板的版本。

命令:

```
aws ec2 describe-launch-template-versions --launch-template-id lt-068f72b72934aff71
```

输出:

```
{
  "LaunchTemplateVersions": [
    {
      "LaunchTemplateId": "lt-068f72b72934aff71",
      "LaunchTemplateName": "Webservers",
      "VersionNumber": 3,
      "CreatedBy": "arn:aws:iam::123456789102:root",
      "LaunchTemplateData": {
        "KeyName": "kp-us-east",
        "ImageId": "ami-6057e21a",
        "InstanceType": "t2.small",
        "NetworkInterfaces": [
          {
            "SubnetId": "subnet-7b16de0c",
            "DeviceIndex": 0,
            "Groups": [
              "sg-7c227019"
            ]
          }
        ]
      },
      "DefaultVersion": false,
      "CreateTime": "2017-11-20T13:19:54.000Z"
    },
    {
```



```
"LaunchTemplateId": "lt-068f72b72934aff71",
"LaunchTemplateName": "Webservers",
"VersionNumber": 2,
"CreatedBy": "arn:aws:iam::123456789102:root",
"LaunchTemplateData": {
  "KeyName": "kp-us-east",
  "ImageId": "ami-6057e21a",
  "InstanceType": "t2.medium",
  "NetworkInterfaces": [
    {
      "SubnetId": "subnet-1a2b3c4d",
      "DeviceIndex": 0,
      "Groups": [
        "sg-7c227019"
      ]
    }
  ]
},
"DefaultVersion": false,
"CreateTime": "2017-11-20T13:12:32.000Z"
},
{
  "LaunchTemplateId": "lt-068f72b72934aff71",
  "LaunchTemplateName": "Webservers",
  "VersionNumber": 1,
  "CreatedBy": "arn:aws:iam::123456789102:root",
  "LaunchTemplateData": {
    "UserData": "",
    "KeyName": "kp-us-east",
    "ImageId": "ami-aabbcc11",
    "InstanceType": "t2.medium",
    "NetworkInterfaces": [
      {
        "SubnetId": "subnet-7b16de0c",
        "DeviceIndex": 0,
        "DeleteOnTermination": false,
        "Groups": [
          "sg-7c227019"
        ],
        "AssociatePublicIpAddress": true
      }
    ]
  ]
},
"DefaultVersion": true,
```

```
        "CreateTime": "2017-11-20T12:52:33.000Z"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLaunchTemplateVersions](#)中的。

## describe-launch-templates

以下代码示例演示了如何使用 describe-launch-templates。

### AWS CLI

#### 描述启动模板

此示例描述了您的启动模板。

命令:

```
aws ec2 describe-launch-templates
```

输出:

```
{
  "LaunchTemplates": [
    {
      "LatestVersionNumber": 2,
      "LaunchTemplateId": "lt-0e06d290751193123",
      "LaunchTemplateName": "TemplateForWebServer",
      "DefaultVersionNumber": 2,
      "CreatedBy": "arn:aws:iam::123456789012:root",
      "CreateTime": "2017-11-27T09:30:23.000Z"
    },
    {
      "LatestVersionNumber": 6,
      "LaunchTemplateId": "lt-0c45b5e061ec98456",
      "LaunchTemplateName": "DBServersTemplate",
      "DefaultVersionNumber": 1,
      "CreatedBy": "arn:aws:iam::123456789012:root",
      "CreateTime": "2017-11-20T09:25:22.000Z"
    },
    {

```

```

    "LatestVersionNumber": 1,
    "LaunchTemplateId": "lt-0d47d774e8e52dabc",
    "LaunchTemplateName": "MyLaunchTemplate2",
    "DefaultVersionNumber": 1,
    "CreatedBy": "arn:aws:iam::123456789012:root",
    "CreateTime": "2017-11-02T12:06:21.000Z"
  },
  {
    "LatestVersionNumber": 3,
    "LaunchTemplateId": "lt-01e5f948eb4f589d6",
    "LaunchTemplateName": "testingtemplate2",
    "DefaultVersionNumber": 1,
    "CreatedBy": "arn:aws:sts::123456789012:assumed-role/AdminRole/i-03ee35176e2e5aabc",
    "CreateTime": "2017-12-01T08:19:48.000Z"
  },
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLaunchTemplates](#)中的。

## describe-local-gateway-route-table-virtual-interface-group-associations

以下代码示例演示了如何使用 `describe-local-gateway-route-table-virtual-interface-group-associations`。

### AWS CLI

描述虚拟接口组和本地网关路由表之间的关联

以下`describe-local-gateway-route-table-virtual-interface-group-associations`示例描述了您的 AWS 账户中虚拟接口组和本地网关路由表之间的关联。

```
aws ec2 describe-local-gateway-route-table-virtual-interface-group-associations
```

输出：

```

{
  "LocalGatewayRouteTableVirtualInterfaceGroupAssociations": [
    {
      "LocalGatewayRouteTableVirtualInterfaceGroupAssociationId": "lgw-vif-grp-assoc-07145b276bEXAMPLE",

```

```

        "LocalGatewayVirtualInterfaceGroupId": "lgw-vif-grp-07145b276bEXAMPLE",
        "LocalGatewayId": "lgw-0ab1c23d4eEXAMPLE",
        "LocalGatewayRouteTableId": "lgw-rtb-059615ef7dEXAMPLE",
        "LocalGatewayRouteTableArn": "arn:aws:ec2:us-west-2:123456789012:local-
gateway-route-table/lgw-rtb-059615ef7dEXAMPLE",
        "OwnerId": "123456789012",
        "State": "associated",
        "Tags": []
    }
]
}

```

有关更多信息，请参阅 AWS Outposts 用户 [指南中的使用本地网关](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参  
考 [DescribeLocalGatewayRouteTableVirtualInterfaceGroupAssociations](#) 中的。

## describe-local-gateway-route-table-vpc-associations

以下代码示例演示了如何使用 describe-local-gateway-route-table-vpc-associations。

### AWS CLI

描述 VPC 和本地网关路由表之间的关联

以下 describe-local-gateway-route-table-vpc-associations 示例显示有关 VPC 和本地网关路由表之间指定关联的信息。

```

aws ec2 describe-local-gateway-route-table-vpc-associations \
  --local-gateway-route-table-vpc-association-ids lgw-vpc-assoc-0e0f27af15EXAMPLE

```

输出：

```

{
  "LocalGatewayRouteTableVpcAssociation": {
    "LocalGatewayRouteTableVpcAssociationId": "lgw-vpc-assoc-0e0f27af1EXAMPLE",
    "LocalGatewayRouteTableId": "lgw-rtb-059615ef7dEXAMPLE",
    "LocalGatewayId": "lgw-09b493aa7cEXAMPLE",
    "VpcId": "vpc-0efe9bde08EXAMPLE",
    "State": "associated"
  }
}

```

```
}
```

有关更多信息，请参阅《Outposts 用户指南》中的[本地网关路由表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLocalGatewayRouteTableVpcAssociations](#)中的。

## describe-local-gateway-route-tables

以下代码示例演示了如何使用 describe-local-gateway-route-tables。

### AWS CLI

描述您的本地网关路由表

以下describe-local-gateway-route-tables示例显示有关本地网关路由表的详细信息。

```
aws ec2 describe-local-gateway-route-tables
```

输出：

```
{
  "LocalGatewayRouteTables": [
    {
      "LocalGatewayRouteTableId": "lgw-rtb-059615ef7deEXAMPLE",
      "LocalGatewayId": "lgw-09b493aa7cEXAMPLE",
      "OutpostArn": "arn:aws:outposts:us-west-2:111122223333:outpost/
op-0dc11b66edEXAMPLE",
      "State": "available"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLocalGatewayRouteTables](#)中的。

## describe-local-gateway-virtual-interface-groups

以下代码示例演示了如何使用 describe-local-gateway-virtual-interface-groups。

### AWS CLI

描述本地网关虚拟接口组

以下describe-local-gateway-virtual-interface-groups示例描述了您 AWS 账户中的本地网关虚拟接口组。

```
aws ec2 describe-local-gateway-virtual-interface-groups
```

输出：

```
{
  "LocalGatewayVirtualInterfaceGroups": [
    {
      "LocalGatewayVirtualInterfaceGroupId": "lgw-vif-grp-07145b276bEXAMPLE",
      "LocalGatewayVirtualInterfaceIds": [
        "lgw-vif-01a23bc4d5EXAMPLE",
        "lgw-vif-543ab21012EXAMPLE"
      ],
      "LocalGatewayId": "lgw-0ab1c23d4eEXAMPLE",
      "OwnerId": "123456789012",
      "Tags": []
    }
  ]
}
```

有关更多信息，请参阅 AWS Outposts 用户[指南中的使用本地网关](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLocalGatewayVirtualInterfaceGroups](#)中的。

## describe-local-gateway-virtual-interfaces

以下代码示例演示了如何使用 describe-local-gateway-virtual-interfaces。

AWS CLI

描述本地网关虚拟接口

以下describe-local-gateway-virtual-interfaces示例描述了您 AWS 账户中的本地网关虚拟接口。

```
aws ec2 describe-local-gateway-virtual-interfaces
```

输出：

```
{
  "LocalGatewayVirtualInterfaces": [
    {
      "LocalGatewayVirtualInterfaceId": "lgw-vif-01a23bc4d5EXAMPLE",
      "LocalGatewayId": "lgw-0ab1c23d4eEXAMPLE",
      "Vlan": 2410,
      "LocalAddress": "0.0.0.0/0",
      "PeerAddress": "0.0.0.0/0",
      "LocalBgpAsn": 65010,
      "PeerBgpAsn": 65000,
      "OwnerId": "123456789012",
      "Tags": []
    },
    {
      "LocalGatewayVirtualInterfaceId": "lgw-vif-543ab21012EXAMPLE",
      "LocalGatewayId": "lgw-0ab1c23d4eEXAMPLE",
      "Vlan": 2410,
      "LocalAddress": "0.0.0.0/0",
      "PeerAddress": "0.0.0.0/0",
      "LocalBgpAsn": 65010,
      "PeerBgpAsn": 65000,
      "OwnerId": "123456789012",
      "Tags": []
    }
  ]
}
```

有关更多信息，请参阅 AWS Outposts 用户 [指南中的使用本地网关](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeLocalGatewayVirtualInterfaces](#) 中的。

## describe-local-gateways

以下代码示例演示了如何使用 describe-local-gateways。

### AWS CLI

描述您的本地网关

以下 describe-local-gateways 示例显示了可供您使用的本地网关的详细信息。

```
aws ec2 describe-local-gateways
```

输出：

```
{
  "LocalGateways": [
    {
      "LocalGatewayId": "lgw-09b493aa7cEXAMPLE",
      "OutpostArn": "arn:aws:outposts:us-west-2:123456789012:outpost/
op-0dc11b66ed59f995a",
      "OwnerId": "123456789012",
      "State": "available"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLocalGateways](#)中的。

## describe-managed-prefix-lists

以下代码示例演示了如何使用 describe-managed-prefix-lists。

AWS CLI

描述托管前缀列表

以下describe-managed-prefix-lists示例描述了 AWS 账户拥有的前缀列表123456789012。

```
aws ec2 describe-managed-prefix-lists \
  --filters Name=owner-id,Values=123456789012
```

输出：

```
{
  "PrefixLists": [
    {
      "PrefixListId": "pl-11223344556677aab",
      "AddressFamily": "IPv6",
      "State": "create-complete",
      "PrefixListArn": "arn:aws:ec2:us-west-2:123456789012:prefix-list/
pl-11223344556677aab",
      "PrefixListName": "vpc-ipv6-cidrs",
      "MaxEntries": 25,
    }
  ]
}
```



```
    "Version": 1,
    "Tags": [],
    "OwnerId": "123456789012"
  },
  {
    "PrefixListId": "pl-0123456abcabcabc1",
    "AddressFamily": "IPv4",
    "State": "active",
    "PrefixListArn": "arn:aws:ec2:us-west-2:123456789012:prefix-list/
pl-0123456abcabcabc1",
    "PrefixListName": "vpc-cidrs",
    "MaxEntries": 10,
    "Version": 1,
    "Tags": [],
    "OwnerId": "123456789012"
  }
]
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的[托管前缀列表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeManagedPrefixLists](#)中的。

## describe-moving-addresses

以下代码示例演示了如何使用 describe-moving-addresses。

### AWS CLI

描述您的搬家地址

此示例描述了您的所有移动弹性 IP 地址。

命令:

```
aws ec2 describe-moving-addresses
```

输出:

```
{
  "MovingAddressStatuses": [
    {
      "PublicIp": "198.51.100.0",
```

```
    "MoveStatus": "MovingToVpc"
  }
]
}
```

此示例描述了所有要迁移到 EC2-VPC 平台的地址。

命令:

```
aws ec2 describe-moving-addresses --filters Name=moving-status,Values=MovingToVpc
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeMovingAddresses](#)中的。

## describe-nat-gateways

以下代码示例演示了如何使用 describe-nat-gateways。

AWS CLI

示例 1：描述公有 NAT 网关

以下describe-nat-gateways示例描述了指定的公有 NAT 网关。

```
aws ec2 describe-nat-gateways \
  --nat-gateway-id nat-01234567890abcdef
```

输出：

```
{
  "NatGateways": [
    {
      "CreateTime": "2023-08-25T01:56:51.000Z",
      "NatGatewayAddresses": [
        {
          "AllocationId": "eipalloc-0790180cd2EXAMPLE",
          "NetworkInterfaceId": "eni-09cc4b2558794f7f9",
          "PrivateIp": "10.0.0.211",
          "PublicIp": "54.85.121.213",
          "AssociationId": "eipassoc-04d295cc9b8815b24",
          "IsPrimary": true,
          "Status": "succeeded"
        }
      ],
    }
  ],
}
```

```

        {
            "AllocationId": "eipalloc-0be6ecac95EXAMPLE",
            "NetworkInterfaceId": "eni-09cc4b2558794f7f9",
            "PrivateIp": "10.0.0.74",
            "PublicIp": "3.211.231.218",
            "AssociationId": "eipassoc-0f96bdca17EXAMPLE",
            "IsPrimary": false,
            "Status": "succeeded"
        }
    ],
    "NatGatewayId": "nat-01234567890abcdef",
    "State": "available",
    "SubnetId": "subnet-655eab5f08EXAMPLE",
    "VpcId": "vpc-098eb5ef58EXAMPLE",
    "Tags": [
        {
            "Key": "Name",
            "Value": "public-nat"
        }
    ],
    "ConnectivityType": "public"
}
]
}

```

## 示例 2：描述私有 NAT 网关

以下 `describe-nat-gateways` 示例描述了指定的私有 NAT 网关。

```
aws ec2 describe-nat-gateways \
  --nat-gateway-id nat-1234567890abcdef0
```

输出：

```

{
  "NatGateways": [
    {
      "CreateTime": "2023-08-25T00:50:05.000Z",
      "NatGatewayAddresses": [
        {
          "NetworkInterfaceId": "eni-0065a61b324d1897a",
          "PrivateIp": "10.0.20.240",
          "IsPrimary": true,

```

```
        "Status": "succeeded"
      },
      {
        "NetworkInterfaceId": "eni-0065a61b324d1897a",
        "PrivateIp": "10.0.20.33",
        "IsPrimary": false,
        "Status": "succeeded"
      },
      {
        "NetworkInterfaceId": "eni-0065a61b324d1897a",
        "PrivateIp": "10.0.20.197",
        "IsPrimary": false,
        "Status": "succeeded"
      }
    ],
    "NatGatewayId": "nat-1234567890abcdef0",
    "State": "available",
    "SubnetId": "subnet-08fc749671EXAMPLE",
    "VpcId": "vpc-098eb5ef58EXAMPLE",
    "Tags": [
      {
        "Key": "Name",
        "Value": "private-nat"
      }
    ],
    "ConnectivityType": "private"
  }
]
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 [NAT 网关](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeNatGateways](#) 中的。

## describe-network-acls

以下代码示例演示了如何使用 `describe-network-acls`。

### AWS CLI

描述您的网络 ACL

以下 `describe-network-acls` 示例检索有关您的网络 ACL 的详细信息。

```
aws ec2 describe-network-acls
```

输出：

```
{
  "NetworkAcls": [
    {
      "Associations": [
        {
          "NetworkAclAssociationId": "aclassoc-0c1679dc41EXAMPLE",
          "NetworkAclId": "acl-0ea1f54ca7EXAMPLE",
          "SubnetId": "subnet-0931fc2fa5EXAMPLE"
        }
      ],
      "Entries": [
        {
          "CidrBlock": "0.0.0.0/0",
          "Egress": true,
          "Protocol": "-1",
          "RuleAction": "allow",
          "RuleNumber": 100
        },
        {
          "CidrBlock": "0.0.0.0/0",
          "Egress": true,
          "Protocol": "-1",
          "RuleAction": "deny",
          "RuleNumber": 32767
        },
        {
          "CidrBlock": "0.0.0.0/0",
          "Egress": false,
          "Protocol": "-1",
          "RuleAction": "allow",
          "RuleNumber": 100
        },
        {
          "CidrBlock": "0.0.0.0/0",
          "Egress": false,
          "Protocol": "-1",
          "RuleAction": "deny",
          "RuleNumber": 32767
        }
      ]
    }
  ]
}
```

```
    ],
    "IsDefault": true,
    "NetworkAclId": "acl-0ea1f54ca7EXAMPLE",
    "Tags": [],
    "VpcId": "vpc-06e4ab6c6cEXAMPLE",
    "OwnerId": "111122223333"
  },
  {
    "Associations": [],
    "Entries": [
      {
        "CidrBlock": "0.0.0.0/0",
        "Egress": true,
        "Protocol": "-1",
        "RuleAction": "allow",
        "RuleNumber": 100
      },
      {
        "Egress": true,
        "Ipv6CidrBlock": ":::/0",
        "Protocol": "-1",
        "RuleAction": "allow",
        "RuleNumber": 101
      },
      {
        "CidrBlock": "0.0.0.0/0",
        "Egress": true,
        "Protocol": "-1",
        "RuleAction": "deny",
        "RuleNumber": 32767
      },
      {
        "Egress": true,
        "Ipv6CidrBlock": ":::/0",
        "Protocol": "-1",
        "RuleAction": "deny",
        "RuleNumber": 32768
      },
      {
        "CidrBlock": "0.0.0.0/0",
        "Egress": false,
        "Protocol": "-1",
        "RuleAction": "allow",
        "RuleNumber": 100
      }
    ]
  }
}
```

```
    },
    {
      "Egress": false,
      "Ipv6CidrBlock": "::/0",
      "Protocol": "-1",
      "RuleAction": "allow",
      "RuleNumber": 101
    },
    {
      "CidrBlock": "0.0.0.0/0",
      "Egress": false,
      "Protocol": "-1",
      "RuleAction": "deny",
      "RuleNumber": 32767
    },
    {
      "Egress": false,
      "Ipv6CidrBlock": "::/0",
      "Protocol": "-1",
      "RuleAction": "deny",
      "RuleNumber": 32768
    }
  ],
  "IsDefault": true,
  "NetworkAclId": "acl-0e2a78e4e2EXAMPLE",
  "Tags": [],
  "VpcId": "vpc-03914afb3eEXAMPLE",
  "OwnerId": "111122223333"
}
]
}
```

有关更多信息，请参阅《AWS VPC 用户指南》中的[网络 ACL](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeNetworkAcls](#)中的。

## describe-network-insights-access-scope-analyses

以下代码示例演示了如何使用 describe-network-insights-access-scope-analyses。

### AWS CLI

描述 Network Insights 访问范围分析

以下describe-network-insights-access-scope-analyses示例描述了您 AWS 账户中的访问范围分析。

```
aws ec2 describe-network-insights-access-scope-analyses \  
  --region us-east-1
```

输出：

```
{  
  "NetworkInsightsAccessScopeAnalyses": [  
    {  
      "NetworkInsightsAccessScopeAnalysisId": "nisa-123456789111",  
      "NetworkInsightsAccessScopeAnalysisArn": "arn:aws:ec2:us-  
east-1:123456789012:network-insights-access-scope-analysis/nisa-123456789111",  
      "NetworkInsightsAccessScopeId": "nis-123456789222",  
      "Status": "succeeded",  
      "StartDate": "2022-01-25T19:45:36.842000+00:00",  
      "FindingsFound": "true",  
      "Tags": []  
    }  
  ]  
}
```

有关更多信息，请参阅《[网络访问分析器指南](#)》中的“[使用 AWS CLI 开始使用网络访问分析器](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeNetworkInsightsAccessScopeAnalyses](#)中的。

## describe-network-insights-access-scopes

以下代码示例演示了如何使用 describe-network-insights-access-scopes。

AWS CLI

描述《网络见解》访问范围

以下describe-network-insights-access-scopes示例描述了您 AWS 账户中的访问范围分析。

```
aws ec2 describe-network-insights-access-scopes \  
  --region us-east-1
```



输出：

```
{
  "NetworkInsightsAccessScopes": [
    {
      "NetworkInsightsAccessScopeId": "nis-123456789111",
      "NetworkInsightsAccessScopeArn": "arn:aws:ec2:us-east-1:123456789012:network-insights-access-scope/nis-123456789111",
      "CreateDate": "2021-11-29T21:12:41.416000+00:00",
      "UpdateDate": "2021-11-29T21:12:41.416000+00:00",
      "Tags": []
    }
  ]
}
```

有关更多信息，请参阅《[网络访问分析器指南](#)》中的“[使用 AWS CLI 开始使用网络访问分析器](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeNetworkInsightsAccessScopes](#) 中的。

## describe-network-insights-analyses

以下代码示例演示了如何使用 describe-network-insights-analyses。

### AWS CLI

#### 查看路径分析的结果

以下 describe-network-insights-analyses 示例描述了指定的分析。在此示例中，源是互联网网关，目标是 EC2 实例，协议是 TCP。分析成功（Status 是 succeeded），路径无法到达（NetworkPathFound 是 false）。解释代码 ENI\_SG\_RULES\_MISMATCH 表明该实例的安全组不包含允许目标端口上的流量的规则。

```
aws ec2 describe-network-insights-analyses \
  --network-insights-analysis-ids nia-02207aa13eb480c7a
```

输出：

```
{
  "NetworkInsightsAnalyses": [
    {
      "NetworkInsightsAnalysisId": "nia-02207aa13eb480c7a",
```

```

    "NetworkInsightsAnalysisArn": "arn:aws:ec2:us-
east-1:123456789012:network-insights-analysis/nia-02207aa13eb480c7a",
    "NetworkInsightsPathId": "nip-0b26f224f1d131fa8",
    "StartDate": "2021-01-20T22:58:37.495Z",
    "Status": "succeeded",
    "NetworkPathFound": false,
    "Explanations": [
      {
        "Direction": "ingress",
        "ExplanationCode": "ENI_SG_RULES_MISMATCH",
        "NetworkInterface": {
          "Id": "eni-0a25edef15a6cc08c",
          "Arn": "arn:aws:ec2:us-east-1:123456789012:network-
interface/eni-0a25edef15a6cc08c"
        },
        "SecurityGroups": [
          {
            "Id": "sg-02f0d35a850ba727f",
            "Arn": "arn:aws:ec2:us-east-1:123456789012:security-
group/sg-02f0d35a850ba727f"
          }
        ],
        "Subnet": {
          "Id": "subnet-004ff41eccb4d1194",
          "Arn": "arn:aws:ec2:us-east-1:123456789012:subnet/
subnet-004ff41eccb4d1194"
        },
        "Vpc": {
          "Id": "vpc-f1663d98ad28331c7",
          "Arn": "arn:aws:ec2:us-east-1:123456789012:vpc/vpc-
f1663d98ad28331c7"
        }
      }
    ],
    "Tags": []
  }
]
}

```

有关更多信息，请参阅 [《Reachability Analyzer 指南》](#) 中的 [AWS CLI 使用入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeNetworkInsightsAnalyses](#) 中的。

## describe-network-insights-paths

以下代码示例演示了如何使用 describe-network-insights-paths。

### AWS CLI

#### 描述路径

以下describe-network-insights-paths示例描述了指定的路径。

```
aws ec2 describe-network-insights-paths \  
  --network-insights-path-ids nip-0b26f224f1d131fa8
```

#### 输出：

```
{  
  "NetworkInsightsPaths": [  
    {  
      "NetworkInsightsPathId": "nip-0b26f224f1d131fa8",  
      "NetworkInsightsPathArn": "arn:aws:ec2:us-east-1:123456789012:network-  
insights-path/nip-0b26f224f1d131fa8",  
      "CreateDate": "2021-01-20T22:43:46.933Z",  
      "Source": "igw-0797cccdc9d73b0e5",  
      "Destination": "i-0495d385ad28331c7",  
      "Protocol": "tcp"  
    }  
  ]  
}
```

有关更多信息，请参阅 [《Reachability Analyzer 指南》中的 AWS CLI 使用入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeNetworkInsightsPaths](#) 中的。

## describe-network-interface-attribute

以下代码示例演示了如何使用 describe-network-interface-attribute。

### AWS CLI

#### 描述网络接口的连接属性

此示例命令描述了指定网络接口的 attachment 属性。

**命令:**

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200 --attribute attachment
```

**输出:**

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Attachment": {
    "Status": "attached",
    "DeviceIndex": 0,
    "AttachTime": "2015-05-21T20:02:20.000Z",
    "InstanceId": "i-1234567890abcdef0",
    "DeleteOnTermination": true,
    "AttachmentId": "eni-attach-43348162",
    "InstanceOwnerId": "123456789012"
  }
}
```

**描述网络接口的描述属性**

此示例命令描述了指定网络接口的description属性。

**命令:**

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200 --attribute description
```

**输出:**

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Description": {
    "Value": "My description"
  }
}
```

**描述网络接口的 GroupSet 属性**

此示例命令描述了指定网络接口的groupSet属性。

**命令:**

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200 --attribute groupSet
```

**输出:**

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Groups": [
    {
      "GroupName": "my-security-group",
      "GroupId": "sg-903004f8"
    }
  ]
}
```

**描述网络接口的 sourceDestCheck 属性**

此示例命令描述了指定网络接口的sourceDestCheck属性。

**命令:**

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200 --attribute sourceDestCheck
```

**输出:**

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "SourceDestCheck": {
    "Value": true
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeNetworkInterfaceAttribute](#)中的。

**describe-network-interface-permissions**

以下代码示例演示了如何使用 describe-network-interface-permissions。

## AWS CLI

### 描述您的网络接口权限

此示例描述了您的所有网络接口权限。

命令:

```
aws ec2 describe-network-interface-permissions
```

输出:

```
{
  "NetworkInterfacePermissions": [
    {
      "PermissionState": {
        "State": "GRANTED"
      },
      "NetworkInterfacePermissionId": "eni-perm-06fd19020ede149ea",
      "NetworkInterfaceId": "eni-b909511a",
      "Permission": "INSTANCE-ATTACH",
      "AwsAccountId": "123456789012"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeNetworkInterfacePermissions](#)中的。

## describe-network-interfaces

以下代码示例演示了如何使用 describe-network-interfaces。

### AWS CLI

#### 描述您的网络接口

此示例描述了您的所有网络接口。

命令:

```
aws ec2 describe-network-interfaces
```

输出：

```
{
  "NetworkInterfaces": [
    {
      "Status": "in-use",
      "MacAddress": "02:2f:8f:b0:cf:75",
      "SourceDestCheck": true,
      "VpcId": "vpc-a01106c2",
      "Description": "my network interface",
      "Association": {
        "PublicIp": "203.0.113.12",
        "AssociationId": "eipassoc-0fbb766a",
        "PublicDnsName": "ec2-203-0-113-12.compute-1.amazonaws.com",
        "IpOwnerId": "123456789012"
      },
      "NetworkInterfaceId": "eni-e5aa89a3",
      "PrivateIpAddresses": [
        {
          "PrivateDnsName": "ip-10-0-1-17.ec2.internal",
          "Association": {
            "PublicIp": "203.0.113.12",
            "AssociationId": "eipassoc-0fbb766a",
            "PublicDnsName": "ec2-203-0-113-12.compute-1.amazonaws.com",
            "IpOwnerId": "123456789012"
          },
          "Primary": true,
          "PrivateIpAddress": "10.0.1.17"
        }
      ],
      "RequesterManaged": false,
      "Ipv6Addresses": [],
      "PrivateDnsName": "ip-10-0-1-17.ec2.internal",
      "AvailabilityZone": "us-east-1d",
      "Attachment": {
        "Status": "attached",
        "DeviceIndex": 1,
        "AttachTime": "2013-11-30T23:36:42.000Z",
        "InstanceId": "i-1234567890abcdef0",
        "DeleteOnTermination": false,
        "AttachmentId": "eni-attach-66c4350a",
        "InstanceOwnerId": "123456789012"
      },
      "Groups": [
```

```
    {
      "GroupName": "default",
      "GroupId": "sg-8637d3e3"
    }
  ],
  "SubnetId": "subnet-b61f49f0",
  "OwnerId": "123456789012",
  "TagSet": [],
  "PrivateIpAddress": "10.0.1.17"
},
{
  "Status": "in-use",
  "MacAddress": "02:58:f5:ef:4b:06",
  "SourceDestCheck": true,
  "VpcId": "vpc-a01106c2",
  "Description": "Primary network interface",
  "Association": {
    "PublicIp": "198.51.100.0",
    "IpOwnerId": "amazon"
  },
  "NetworkInterfaceId": "eni-f9ba99bf",
  "PrivateIpAddresses": [
    {
      "Association": {
        "PublicIp": "198.51.100.0",
        "IpOwnerId": "amazon"
      },
      "Primary": true,
      "PrivateIpAddress": "10.0.1.149"
    }
  ],
  "RequesterManaged": false,
  "Ipv6Addresses": [],
  "AvailabilityZone": "us-east-1d",
  "Attachment": {
    "Status": "attached",
    "DeviceIndex": 0,
    "AttachTime": "2013-11-30T23:35:33.000Z",
    "InstanceId": "i-0598c7d356eba48d7",
    "DeleteOnTermination": true,
    "AttachmentId": "eni-attach-1b9db777",
    "InstanceOwnerId": "123456789012"
  },
  "Groups": [
```



```
    {
      "GroupName": "default",
      "GroupId": "sg-8637d3e3"
    },
    "SubnetId": "subnet-b61f49f0",
    "OwnerId": "123456789012",
    "TagSet": [],
    "PrivateIpAddress": "10.0.1.149"
  ]
}
```

此示例描述了带有带有密钥Purpose和值的标签的网络接口Prod。

命令:

```
aws ec2 describe-network-interfaces --filters Name=tag:Purpose,Values=Prod
```

输出:

```
{
  "NetworkInterfaces": [
    {
      "Status": "available",
      "MacAddress": "12:2c:bd:f9:bf:17",
      "SourceDestCheck": true,
      "VpcId": "vpc-8941ebec",
      "Description": "ProdENI",
      "NetworkInterfaceId": "eni-b9a5ac93",
      "PrivateIpAddresses": [
        {
          "PrivateDnsName": "ip-10-0-1-55.ec2.internal",
          "Primary": true,
          "PrivateIpAddress": "10.0.1.55"
        },
        {
          "PrivateDnsName": "ip-10-0-1-117.ec2.internal",
          "Primary": false,
          "PrivateIpAddress": "10.0.1.117"
        }
      ],
      "RequesterManaged": false,
    }
  ]
}
```

```
    "PrivateDnsName": "ip-10-0-1-55.ec2.internal",
    "AvailabilityZone": "us-east-1d",
    "Ipv6Addresses": [],
    "Groups": [
      {
        "GroupName": "MySG",
        "GroupId": "sg-905002f5"
      }
    ],
    "SubnetId": "subnet-31d6c219",
    "OwnerId": "123456789012",
    "TagSet": [
      {
        "Value": "Prod",
        "Key": "Purpose"
      }
    ],
    "PrivateIpAddress": "10.0.1.55"
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeNetworkInterfaces](#)中的。

## describe-placement-groups

以下代码示例演示了如何使用 describe-placement-groups。

### AWS CLI

描述您的置放群组

此示例命令描述了您的所有置放群组。

命令:

```
aws ec2 describe-placement-groups
```

输出:

```
{
```

```
    "PlacementGroups": [  
      {  
        "GroupName": "my-cluster",  
        "State": "available",  
        "Strategy": "cluster"  
      },  
      ...  
    ]  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePlacementGroups](#)中的。

## describe-prefix-lists

以下代码示例演示了如何使用 describe-prefix-lists。

### AWS CLI

#### 描述前缀列表

此示例列出了该区域的所有可用前缀列表。

命令：

```
aws ec2 describe-prefix-lists
```

输出：

```
{  
  "PrefixLists": [  
    {  
      "PrefixListName": "com.amazonaws.us-east-1.s3",  
      "Cidrs": [  
        "54.231.0.0/17"  
      ],  
      "PrefixListId": "pl-63a5400a"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePrefixLists](#)中的。

## describe-principal-id-format

以下代码示例演示了如何使用 `describe-principal-id-format`。

### AWS CLI

描述启用长 ID 格式的 IAM 用户和角色的 ID 格式

以下 `describe-principal-id-format` 示例描述了根用户、所有 IAM 角色和启用长 ID 格式的所有 IAM 用户的 ID 格式。

```
aws ec2 describe-principal-id-format \  
  --resource instance
```

输出：

```
{  
  "Principals": [  
    {  
      "Arn": "arn:aws:iam::123456789012:root",  
      "Statuses": [  
        {  
          "Deadline": "2016-12-15T00:00:00.000Z",  
          "Resource": "reservation",  
          "UseLongIds": true  
        },  
        {  
          "Deadline": "2016-12-15T00:00:00.000Z",  
          "Resource": "instance",  
          "UseLongIds": true  
        },  
        {  
          "Deadline": "2016-12-15T00:00:00.000Z",  
          "Resource": "volume",  
          "UseLongIds": true  
        }  
      ]  
    },  
    ...  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribePrincipalIdFormat](#) 中的。

## describe-public-ipv4-pools

以下代码示例演示了如何使用 `describe-public-ipv4-pools`。

### AWS CLI

描述您的公有 IPv4 地址池

以下 `describe-public-ipv4-pools` 示例显示了有关您使用自带 IP 地址 (BYOIP) 配置公有 IPv4 地址范围时创建的地址池的详细信息。

```
aws ec2 describe-public-ipv4-pools
```

输出：

```
{
  "PublicIpv4Pools": [
    {
      "PoolId": "ipv4pool-ec2-1234567890abcdef0",
      "PoolAddressRanges": [
        {
          "FirstAddress": "203.0.113.0",
          "LastAddress": "203.0.113.255",
          "AddressCount": 256,
          "AvailableAddressCount": 256
        }
      ],
      "TotalAddressCount": 256,
      "TotalAvailableAddressCount": 256
    }
  ]
}
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [DescribePublicIpv4Pools](#)。

## describe-regions

以下代码示例演示了如何使用 `describe-regions`。

### AWS CLI

示例 1：描述所有已启用的区域

以下 `describe-regions` 示例描述了为您的账户启用的所有区域。

```
aws ec2 describe-regions
```

输出：

```
{
  "Regions": [
    {
      "Endpoint": "ec2.eu-north-1.amazonaws.com",
      "RegionName": "eu-north-1",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.ap-south-1.amazonaws.com",
      "RegionName": "ap-south-1",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.eu-west-3.amazonaws.com",
      "RegionName": "eu-west-3",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.eu-west-2.amazonaws.com",
      "RegionName": "eu-west-2",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.eu-west-1.amazonaws.com",
      "RegionName": "eu-west-1",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.ap-northeast-3.amazonaws.com",
      "RegionName": "ap-northeast-3",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.ap-northeast-2.amazonaws.com",
      "RegionName": "ap-northeast-2",
      "OptInStatus": "opt-in-not-required"
    }
  ],
}
```

```
{
  "Endpoint": "ec2.ap-northeast-1.amazonaws.com",
  "RegionName": "ap-northeast-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.sa-east-1.amazonaws.com",
  "RegionName": "sa-east-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.ca-central-1.amazonaws.com",
  "RegionName": "ca-central-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.ap-southeast-1.amazonaws.com",
  "RegionName": "ap-southeast-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.ap-southeast-2.amazonaws.com",
  "RegionName": "ap-southeast-2",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.eu-central-1.amazonaws.com",
  "RegionName": "eu-central-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.us-east-1.amazonaws.com",
  "RegionName": "us-east-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.us-east-2.amazonaws.com",
  "RegionName": "us-east-2",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.us-west-1.amazonaws.com",
  "RegionName": "us-west-1",
  "OptInStatus": "opt-in-not-required"
}
```

```
    },
    {
      "Endpoint": "ec2.us-west-2.amazonaws.com",
      "RegionName": "us-west-2",
      "OptInStatus": "opt-in-not-required"
    }
  ]
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[区域和区](#)。

示例 2：描述具有端点的已启用区域，其名称包含特定字符串

以下 describe-regions 示例描述了在端点中具有字符串“us”的所有已启用区域。

```
aws ec2 describe-regions \
  --filters "Name=endpoint,Values=*us*"
```

输出：

```
{
  "Regions": [
    {
      "Endpoint": "ec2.us-east-1.amazonaws.com",
      "RegionName": "us-east-1"
    },
    {
      "Endpoint": "ec2.us-east-2.amazonaws.com",
      "RegionName": "us-east-2"
    },
    {
      "Endpoint": "ec2.us-west-1.amazonaws.com",
      "RegionName": "us-west-1"
    },
    {
      "Endpoint": "ec2.us-west-2.amazonaws.com",
      "RegionName": "us-west-2"
    }
  ]
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[区域和区](#)。



### 示例 3：描述所有区域

以下 `describe-regions` 示例描述了所有可用区域，包括已禁用的区域。

```
aws ec2 describe-regions \  
  --all-regions
```

输出：

```
{  
  "Regions": [  
    {  
      "Endpoint": "ec2.eu-north-1.amazonaws.com",  
      "RegionName": "eu-north-1",  
      "OptInStatus": "opt-in-not-required"  
    },  
    {  
      "Endpoint": "ec2.ap-south-1.amazonaws.com",  
      "RegionName": "ap-south-1",  
      "OptInStatus": "opt-in-not-required"  
    },  
    {  
      "Endpoint": "ec2.eu-west-3.amazonaws.com",  
      "RegionName": "eu-west-3",  
      "OptInStatus": "opt-in-not-required"  
    },  
    {  
      "Endpoint": "ec2.eu-west-2.amazonaws.com",  
      "RegionName": "eu-west-2",  
      "OptInStatus": "opt-in-not-required"  
    },  
    {  
      "Endpoint": "ec2.eu-west-1.amazonaws.com",  
      "RegionName": "eu-west-1",  
      "OptInStatus": "opt-in-not-required"  
    },  
    {  
      "Endpoint": "ec2.ap-northeast-3.amazonaws.com",  
      "RegionName": "ap-northeast-3",  
      "OptInStatus": "opt-in-not-required"  
    },  
    {  
      "Endpoint": "ec2.me-south-1.amazonaws.com",
```

```
    "RegionName": "me-south-1",
    "OptInStatus": "not-opted-in"
  },
  {
    "Endpoint": "ec2.ap-northeast-2.amazonaws.com",
    "RegionName": "ap-northeast-2",
    "OptInStatus": "opt-in-not-required"
  },
  {
    "Endpoint": "ec2.ap-northeast-1.amazonaws.com",
    "RegionName": "ap-northeast-1",
    "OptInStatus": "opt-in-not-required"
  },
  {
    "Endpoint": "ec2.sa-east-1.amazonaws.com",
    "RegionName": "sa-east-1",
    "OptInStatus": "opt-in-not-required"
  },
  {
    "Endpoint": "ec2.ca-central-1.amazonaws.com",
    "RegionName": "ca-central-1",
    "OptInStatus": "opt-in-not-required"
  },
  {
    "Endpoint": "ec2.ap-east-1.amazonaws.com",
    "RegionName": "ap-east-1",
    "OptInStatus": "not-opted-in"
  },
  {
    "Endpoint": "ec2.ap-southeast-1.amazonaws.com",
    "RegionName": "ap-southeast-1",
    "OptInStatus": "opt-in-not-required"
  },
  {
    "Endpoint": "ec2.ap-southeast-2.amazonaws.com",
    "RegionName": "ap-southeast-2",
    "OptInStatus": "opt-in-not-required"
  },
  {
    "Endpoint": "ec2.eu-central-1.amazonaws.com",
    "RegionName": "eu-central-1",
    "OptInStatus": "opt-in-not-required"
  },
  {
```

```
    "Endpoint": "ec2.us-east-1.amazonaws.com",
    "RegionName": "us-east-1",
    "OptInStatus": "opt-in-not-required"
  },
  {
    "Endpoint": "ec2.us-east-2.amazonaws.com",
    "RegionName": "us-east-2",
    "OptInStatus": "opt-in-not-required"
  },
  {
    "Endpoint": "ec2.us-west-1.amazonaws.com",
    "RegionName": "us-west-1",
    "OptInStatus": "opt-in-not-required"
  },
  {
    "Endpoint": "ec2.us-west-2.amazonaws.com",
    "RegionName": "us-west-2",
    "OptInStatus": "opt-in-not-required"
  }
]
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[区域和区](#)。

#### 示例 4：仅列出区域名称

以下 describe-regions 示例使用 --query 参数筛选输出，并仅以文本形式返回区域的名称。

```
aws ec2 describe-regions \
  --all-regions \
  --query "Regions[].{Name:RegionName}" \
  --output text
```

输出：

```
eu-north-1
ap-south-1
eu-west-3
eu-west-2
eu-west-1
ap-northeast-3
ap-northeast-2
me-south-1
```

```
ap-northeast-1
sa-east-1
ca-central-1
ap-east-1
ap-southeast-1
ap-southeast-2
eu-central-1
us-east-1
us-east-2
us-west-1
us-west-2
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[区域和区](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeRegions](#)中的。

## describe-replace-root-volume-tasks

以下代码示例演示了如何使用 `describe-replace-root-volume-tasks`。

### AWS CLI

示例 1：查看有关特定根卷更换任务的信息

以下 `describe-replace-root-volume-tasks` 示例描述了根卷替换任务 `replacevol-0111122223333abcd`。

```
aws ec2 describe-replace-root-volume-tasks \
  --replace-root-volume-task-ids replacevol-0111122223333abcd
```

输出：

```
{
  "ReplaceRootVolumeTasks": [
    {
      "ReplaceRootVolumeTaskId": "replacevol-0111122223333abcd",
      "Tags": [],
      "InstanceId": "i-0123456789abcdefa",
      "TaskState": "succeeded",
      "StartTime": "2022-03-14T15:16:28Z",
      "CompleteTime": "2022-03-14T15:16:52Z"
    }
  ]
}
```

```
}
```

有关更多信息，请参阅 Amazon 弹性计算云用户指南中的[替换根卷](#)。

示例 2：查看有关特定实例的所有根卷更换任务的信息

以下describe-replace-root-volume-tasks示例描述了所有根卷更换任务，例如 i-0123456789abcdefa。

```
aws ec2 describe-replace-root-volume-tasks \  
  --filters Name=instance-id,Values=i-0123456789abcdefa
```

输出：

```
{  
  "ReplaceRootVolumeTasks": [  
    {  
      "ReplaceRootVolumeTaskId": "replacevol-0111122223333abcd",  
      "Tags": [],  
      "InstanceId": "i-0123456789abcdefa",  
      "TaskState": "succeeded",  
      "StartTime": "2022-03-14T15:06:38Z",  
      "CompleteTime": "2022-03-14T15:07:03Z"  
    },  
    {  
      "ReplaceRootVolumeTaskId": "replacevol-0444455555555abcd",  
      "Tags": [],  
      "InstanceId": "i-0123456789abcdefa",  
      "TaskState": "succeeded",  
      "StartTime": "2022-03-14T15:16:28Z",  
      "CompleteTime": "2022-03-14T15:16:52Z"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon 弹性计算云用户指南中的[替换根卷](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeReplaceRootVolumeTasks](#)中的。

## describe-reserved-instances-listings

以下代码示例演示了如何使用 describe-reserved-instances-listings。

## AWS CLI

### 描述预留实例清单

以下describe-reserved-instances-listings示例检索有关指定预留实例列表的信息。

```
aws ec2 describe-reserved-instances-listings \
  --reserved-instances-listing-id 5ec28771-05ff-4b9b-aa31-9e57dexample
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeReservedInstancesListings](#)中的。

## describe-reserved-instances-modifications

以下代码示例演示了如何使用 describe-reserved-instances-modifications。

## AWS CLI

### 描述预留实例的修改

此示例命令描述了为您的账户提交的所有预留实例修改请求。

命令:

```
aws ec2 describe-reserved-instances-modifications
```

输出:

```
{
  "ReservedInstancesModifications": [
    {
      "Status": "fulfilled",
      "ModificationResults": [
        {
          "ReservedInstancesId": "93bbbca2-62f1-4d9d-b225-16bada29e6c7",
          "TargetConfiguration": {
            "AvailabilityZone": "us-east-1b",
            "InstanceType": "m1.large",
            "InstanceCount": 3
          }
        }
      ],
    }
  ]
}
```

```

        "ReservedInstancesId": "1ba8e2e3-aabb-46c3-bcf5-3fe2fda922e6",
        "TargetConfiguration": {
            "AvailabilityZone": "us-east-1d",
            "InstanceType": "m1.xlarge",
            "InstanceCount": 1
        }
    ],
    "EffectiveDate": "2015-08-12T17:00:00.000Z",
    "CreateDate": "2015-08-12T17:52:52.630Z",
    "UpdateDate": "2015-08-12T18:08:06.698Z",
    "ClientToken": "c9adb218-3222-4889-8216-0cf0e52dc37e:
    "ReservedInstancesModificationId": "rimod-d3ed4335-b1d3-4de6-
ab31-0f13aaf46687",
    "ReservedInstancesIds": [
        {
            "ReservedInstancesId": "b847fa93-e282-4f55-b59a-1342f5bd7c02"
        }
    ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeReservedInstancesModifications](#)中的。

## describe-reserved-instances-offerings

以下代码示例演示了如何使用 describe-reserved-instances-offerings。

### AWS CLI

#### 描述预留实例产品

此示例命令描述了该地区所有可供购买的预留实例。

命令:

```
aws ec2 describe-reserved-instances-offerings
```

输出:

```
{
```

```
"ReservedInstancesOfferings": [  
  {  
    "OfferingType": "Partial Upfront",  
    "AvailabilityZone": "us-east-1b",  
    "InstanceTenancy": "default",  
    "PricingDetails": [],  
    "ProductDescription": "Red Hat Enterprise Linux",  
    "UsagePrice": 0.0,  
    "RecurringCharges": [  
      {  
        "Amount": 0.088,  
        "Frequency": "Hourly"  
      }  
    ],  
    "Marketplace": false,  
    "CurrencyCode": "USD",  
    "FixedPrice": 631.0,  
    "Duration": 94608000,  
    "ReservedInstancesOfferingId": "9a06095a-bdc6-47fe-a94a-2a382f016040",  
    "InstanceType": "c1.medium"  
  },  
  {  
    "OfferingType": "PartialUpfront",  
    "AvailabilityZone": "us-east-1b",  
    "InstanceTenancy": "default",  
    "PricingDetails": [],  
    "ProductDescription": "Linux/UNIX",  
    "UsagePrice": 0.0,  
    "RecurringCharges": [  
      {  
        "Amount": 0.028,  
        "Frequency": "Hourly"  
      }  
    ],  
    "Marketplace": false,  
    "CurrencyCode": "USD",  
    "FixedPrice": 631.0,  
    "Duration": 94608000,  
    "ReservedInstancesOfferingId": "bfbefc6c-0d10-418d-b144-7258578d329d",  
    "InstanceType": "c1.medium"  
  },  
  ...  
]
```



## 使用选项描述您的预留实例产品

此示例列出了按以下规格提供的 AWS 预留实例：t1.micro 实例类型、Windows ( 亚马逊 VPC ) 产品和高使用率产品。

命令:

```
aws ec2 describe-reserved-instances-offerings --no-include-marketplace --instance-type "t1.micro" --product-description "Windows (Amazon VPC)" --offering-type "no upfront"
```

输出 :

```
{
  "ReservedInstancesOfferings": [
    {
      "OfferingType": "No Upfront",
      "AvailabilityZone": "us-east-1b",
      "InstanceTenancy": "default",
      "PricingDetails": [],
      "ProductDescription": "Windows",
      "UsagePrice": 0.0,
      "RecurringCharges": [
        {
          "Amount": 0.015,
          "Frequency": "Hourly"
        }
      ],
      "Marketplace": false,
      "CurrencyCode": "USD",
      "FixedPrice": 0.0,
      "Duration": 31536000,
      "ReservedInstancesOfferingId": "c48ab04c-fe69-4f94-8e39-a23842292823",
      "InstanceType": "t1.micro"
    },
    ...
    {
      "OfferingType": "No Upfront",
      "AvailabilityZone": "us-east-1d",
      "InstanceTenancy": "default",
      "PricingDetails": [],
      "ProductDescription": "Windows (Amazon VPC)",
```

```
    "UsagePrice": 0.0,
    "RecurringCharges": [
      {
        "Amount": 0.015,
        "Frequency": "Hourly"
      }
    ],
    "Marketplace": false,
    "CurrencyCode": "USD",
    "FixedPrice": 0.0,
    "Duration": 31536000,
    "ReservedInstancesOfferingId": "3a98bf7d-2123-42d4-b4f5-8dbec4b06dc6",
    "InstanceType": "t1.micro"
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeReservedInstancesOfferings](#)中的。

## describe-reserved-instances

以下代码示例演示了如何使用 describe-reserved-instances。

### AWS CLI

描述您的预留实例

此示例命令描述了您拥有的预留实例。

命令:

```
aws ec2 describe-reserved-instances
```

输出:

```
{
  "ReservedInstances": [
    {
      "ReservedInstancesId": "b847fa93-e282-4f55-b59a-1342fexample",
      "OfferingType": "No Upfront",
      "AvailabilityZone": "us-west-1c",
      "End": "2016-08-14T21:34:34.000Z",
      "ProductDescription": "Linux/UNIX",
```

```

    "UsagePrice": 0.00,
    "RecurringCharges": [
      {
        "Amount": 0.104,
        "Frequency": "Hourly"
      }
    ],
    "Start": "2015-08-15T21:34:35.086Z",
    "State": "active",
    "FixedPrice": 0.0,
    "CurrencyCode": "USD",
    "Duration": 31536000,
    "InstanceTenancy": "default",
    "InstanceType": "m3.medium",
    "InstanceCount": 2
  },
  ...
]
}

```

### 使用筛选器描述您的预留实例

此示例筛选了响应，使其仅包含 us-west-1c 中为期三年的 t2.micro Linux/UNIX 预留实例。

#### 命令:

```

aws ec2 describe-reserved-instances --filters Name=duration,Values=94608000
Name=instance-type,Values=t2.micro Name=product-description,Values=Linux/UNIX
Name=availability-zone,Values=us-east-1e

```

#### 输出:

```

{
  "ReservedInstances": [
    {
      "ReservedInstancesId": "f127bd27-edb7-44c9-a0eb-0d7e09259af0",
      "OfferingType": "All Upfront",
      "AvailabilityZone": "us-east-1e",
      "End": "2018-03-26T21:34:34.000Z",
      "ProductDescription": "Linux/UNIX",
      "UsagePrice": 0.00,
      "RecurringCharges": [],
      "Start": "2015-03-27T21:34:35.848Z",

```

```
        "State": "active",
        "FixedPrice": 151.0,
        "CurrencyCode": "USD",
        "Duration": 94608000,
        "InstanceTenancy": "default",
        "InstanceType": "t2.micro",
        "InstanceCount": 1
    }
]
}
```

有关更多信息，请参阅《AWS 命令行界面用户指南》中的“使用 Amazon EC2 实例”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeReservedInstances](#)中的。

## describe-route-tables

以下代码示例演示了如何使用 describe-route-tables。

### AWS CLI

描述您的路由表

以下describe-route-tables示例检索有关您的路由表的详细信息

```
aws ec2 describe-route-tables
```

输出：

```
{
  "RouteTables": [
    {
      "Associations": [
        {
          "Main": true,
          "RouteTableAssociationId": "rtbassoc-0df3f54e06EXAMPLE",
          "RouteTableId": "rtb-09ba434c1bEXAMPLE"
        }
      ],
      "PropagatingVgws": [],
      "RouteTableId": "rtb-09ba434c1bEXAMPLE",
      "Routes": [
        {
```

```
        "DestinationCidrBlock": "10.0.0.0/16",
        "GatewayId": "local",
        "Origin": "CreateRouteTable",
        "State": "active"
    },
    {
        "DestinationCidrBlock": "0.0.0.0/0",
        "NatGatewayId": "nat-06c018cbd8EXAMPLE",
        "Origin": "CreateRoute",
        "State": "blackhole"
    }
],
"Tags": [],
"VpcId": "vpc-0065acced4EXAMPLE",
"OwnerId": "111122223333"
},
{
    "Associations": [
        {
            "Main": true,
            "RouteTableAssociationId": "rtbassoc-9EXAMPLE",
            "RouteTableId": "rtb-a1eec7de"
        }
    ],
    "PropagatingVgws": [],
    "RouteTableId": "rtb-a1eec7de",
    "Routes": [
        {
            "DestinationCidrBlock": "172.31.0.0/16",
            "GatewayId": "local",
            "Origin": "CreateRouteTable",
            "State": "active"
        },
        {
            "DestinationCidrBlock": "0.0.0.0/0",
            "GatewayId": "igw-fEXAMPLE",
            "Origin": "CreateRoute",
            "State": "active"
        }
    ],
    "Tags": [],
    "VpcId": "vpc-3EXAMPLE",
    "OwnerId": "111122223333"
},
```

```

    {
      "Associations": [
        {
          "Main": false,
          "RouteTableAssociationId": "rtbassoc-0b100c28b2EXAMPLE",
          "RouteTableId": "rtb-07a98f76e5EXAMPLE",
          "SubnetId": "subnet-0d3d002af8EXAMPLE"
        }
      ],
      "PropagatingVgws": [],
      "RouteTableId": "rtb-07a98f76e5EXAMPLE",
      "Routes": [
        {
          "DestinationCidrBlock": "10.0.0.0/16",
          "GatewayId": "local",
          "Origin": "CreateRouteTable",
          "State": "active"
        },
        {
          "DestinationCidrBlock": "0.0.0.0/0",
          "GatewayId": "igw-06cf664d80EXAMPLE",
          "Origin": "CreateRoute",
          "State": "active"
        }
      ],
      "Tags": [],
      "VpcId": "vpc-0065acced4EXAMPLE",
      "OwnerId": "111122223333"
    }
  ]
}

```

有关更多信息，请参阅《AWS VPC 用户指南》中的[使用路由表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeRouteTables](#)中的。

## describe-scheduled-instance-availability

以下代码示例演示了如何使用 describe-scheduled-instance-availability。

### AWS CLI

描述可用的日程安排

此示例描述了从指定日期开始每周星期日发生的日程安排。

命令:

```
aws ec2 describe-scheduled-instance-availability --recurrence
Frequency=Weekly,Interval=1,OccurrenceDays=[1] --first-slot-start-time-range
EarliestTime=2016-01-31T00:00:00Z,LatestTime=2016-01-31T04:00:00Z
```

输出:

```
{
  "ScheduledInstanceAvailabilitySet": [
    {
      "AvailabilityZone": "us-west-2b",
      "TotalScheduledInstanceHours": 1219,
      "PurchaseToken": "eyJ2IjoiMSIsInMiOiJEsImMiOi...",
      "MinTermDurationInDays": 366,
      "AvailableInstanceCount": 20,
      "Recurrence": {
        "OccurrenceDaySet": [
          1
        ],
        "Interval": 1,
        "Frequency": "Weekly",
        "OccurrenceRelativeToEnd": false
      },
      "Platform": "Linux/UNIX",
      "FirstSlotStartTime": "2016-01-31T00:00:00Z",
      "MaxTermDurationInDays": 366,
      "SlotDurationInHours": 23,
      "NetworkPlatform": "EC2-VPC",
      "InstanceType": "c4.large",
      "HourlyPrice": "0.095"
    },
    ...
  ]
}
```

要缩小结果范围，您可以添加过滤器来指定操作系统、网络 and 实例类型。

命令:

--filters name=Platform , values=linux/UNIX Name=Network-Platform , values=ec2-vpc 名称=实例类型 , values=c4.large

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeScheduledInstanceAvailability](#)中的。

## describe-scheduled-instances

以下代码示例演示了如何使用 describe-scheduled-instances。

### AWS CLI

描述您的计划实例

此示例描述了指定的计划实例。

命令:

```
aws ec2 describe-scheduled-instances --scheduled-instance-ids
sci-1234-1234-1234-1234-123456789012
```

输出:

```
{
  "ScheduledInstanceSet": [
    {
      "AvailabilityZone": "us-west-2b",
      "ScheduledInstanceId": "sci-1234-1234-1234-1234-123456789012",
      "HourlyPrice": "0.095",
      "CreateDate": "2016-01-25T21:43:38.612Z",
      "Recurrence": {
        "OccurrenceDaySet": [
          1
        ],
        "Interval": 1,
        "Frequency": "Weekly",
        "OccurrenceRelativeToEnd": false,
        "OccurrenceUnit": ""
      },
      "Platform": "Linux/UNIX",
      "TermEndDate": "2017-01-31T09:00:00Z",
      "InstanceCount": 1,
      "SlotDurationInHours": 32,
    }
  ]
}
```



```
        "TermStartDate": "2016-01-31T09:00:00Z",
        "NetworkPlatform": "EC2-VPC",
        "TotalScheduledInstanceHours": 1696,
        "NextSlotStartTime": "2016-01-31T09:00:00Z",
        "InstanceType": "c4.large"
    }
]
}
```

此示例描述了您的所有计划实例。

命令:

```
aws ec2 describe-scheduled-instances
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeScheduledInstances](#)中的。

## describe-security-group-references

以下代码示例演示了如何使用 describe-security-group-references。

AWS CLI

描述安全组引用

此示例描述了的安全组引用sg-bbbb2222。响应表明 VPC 中的安全组sg-bbbb2222正在引用安全组vpc-aaaaaaaa。

命令:

```
aws ec2 describe-security-group-references --group-id sg-bbbbb22222
```

输出:

```
{
  "SecurityGroupsReferenceSet": [
    {
      "ReferencingVpcId": "vpc-aaaaaaaa ",
      "GroupId": "sg-bbbbb22222",
      "VpcPeeringConnectionId": "pcx-b04deed9"
    }
  ]
}
```

```
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSecurityGroupReferences](#)中的。

## describe-security-group-rules

以下代码示例演示了如何使用 describe-security-group-rules。

### AWS CLI

#### 示例 1：描述安全组的安全组规则

以下describe-security-group-rules示例描述了指定安全组的安全组规则。使用filters选项将结果范围限定为特定的安全组。

```
aws ec2 describe-security-group-rules \
  --filters Name="group-id",Values="sg-1234567890abcdef0"
```

输出：

```
{
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-abcdef01234567890",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "111122223333",
      "IsEgress": false,
      "IpProtocol": "-1",
      "FromPort": -1,
      "ToPort": -1,
      "ReferencedGroupInfo": {
        "GroupId": "sg-1234567890abcdef0",
        "UserId": "111122223333"
      },
      "Tags": []
    },
    {
      "SecurityGroupRuleId": "sgr-bcdef01234567890a",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "111122223333",
      "IsEgress": true,
```

```

        "IpProtocol": "-1",
        "FromPort": -1,
        "ToPort": -1,
        "CidrIpv6": "::/0",
        "Tags": []
    },
    {
        "SecurityGroupRuleId": "sgr-cdef01234567890ab",
        "GroupId": "sg-1234567890abcdef0",
        "GroupOwnerId": "111122223333",
        "IsEgress": true,
        "IpProtocol": "-1",
        "FromPort": -1,
        "ToPort": -1,
        "CidrIpv4": "0.0.0.0/0",
        "Tags": []
    }
]
}

```

## 示例 2：描述安全组规则

以下describe-security-group-rules示例描述了指定的安全组规则。

```

aws ec2 describe-security-group-rules \
  --security-group-rule-ids sgr-cdef01234567890ab

```

输出：

```

{
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-cdef01234567890ab",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "111122223333",
      "IsEgress": true,
      "IpProtocol": "-1",
      "FromPort": -1,
      "ToPort": -1,
      "CidrIpv4": "0.0.0.0/0",
      "Tags": []
    }
  ]
}

```

```
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的[安全组规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeSecurityGroupRules](#)中的。

## describe-security-groups

以下代码示例演示了如何使用 describe-security-groups。

### AWS CLI

#### 示例 1：描述安全组

以下 describe-security-groups 示例描述了指定的安全组。

```
aws ec2 describe-security-groups \  
  --group-ids sg-903004f8
```

输出：

```
{  
  "SecurityGroups": [  
    {  
      "IpPermissionsEgress": [  
        {  
          "IpProtocol": "-1",  
          "IpRanges": [  
            {  
              "CidrIp": "0.0.0.0/0"  
            }  
          ],  
          "UserIdGroupPairs": [],  
          "PrefixListIds": []  
        }  
      ],  
      "Description": "My security group",  
      "Tags": [  
        {  
          "Value": "SG1",  
          "Key": "Name"  
        }  
      ],  
    }  
  ],  
}
```

```

    "IpPermissions": [
      {
        "IpProtocol": "-1",
        "IpRanges": [],
        "UserIdGroupPairs": [
          {
            "UserId": "123456789012",
            "GroupId": "sg-903004f8"
          }
        ],
        "PrefixListIds": []
      },
      {
        "PrefixListIds": [],
        "FromPort": 22,
        "IpRanges": [
          {
            "Description": "Access from NY office",
            "CidrIp": "203.0.113.0/24"
          }
        ],
        "ToPort": 22,
        "IpProtocol": "tcp",
        "UserIdGroupPairs": []
      }
    ],
    "GroupName": "MySecurityGroup",
    "VpcId": "vpc-1a2b3c4d",
    "OwnerId": "123456789012",
    "GroupId": "sg-903004f8",
  }
]
}

```

## 示例 2：描述具有特定规则的安全组

以下 `describe-security-groups` 示例使用过滤器将结果范围限定为具有允许 SSH 流量的规则（端口 22）和允许来自所有地址的流量的规则（`0.0.0.0/0`）的安全组。示例使用 `--query` 参数仅显示安全组的名称。安全组必须匹配要在结果中返回的所有筛选条件；但是，单个规则不必匹配所有筛选条件。例如，输出返回一个安全组，其中一个规则允许来自特定 IP 地址的 SSH 流量，另一个规则允许来自所有地址的 HTTP 流量。

```
aws ec2 describe-security-groups \
```

```
--filters Name=ip-permission.from-port,Values=22 Name=ip-permission.to-
port,Values=22 Name=ip-permission.cidr,Values='0.0.0.0/0' \
--query "SecurityGroups[*].[GroupName]" \
--output text
```

输出：

```
default
my-security-group
web-servers
launch-wizard-1
```

### 示例 3：根据标签描述安全组

以下 `describe-security-groups` 示例使用筛选器，将结果范围限定为安全组名称中包含 `test` 且带有标签 `Test=To-delete` 的安全组。示例使用 `--query` 参数仅显示安全组的名称和 ID。

```
aws ec2 describe-security-groups \
--filters Name=group-name,Values=*test* Name=tag:Test,Values=To-delete \
--query "SecurityGroups[*].{Name:GroupName, ID:GroupId}"
```

输出：

```
[
  {
    "Name": "testfornewinstance",
    "ID": "sg-33bb22aa"
  },
  {
    "Name": "newgroupptest",
    "ID": "sg-1a2b3c4d"
  }
]
```

有关使用标签筛选器的更多示例，请参阅《Amazon EC2 用户指南》中的[使用标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeSecurityGroups](#) 中的。

## describe-snapshot-attribute

以下代码示例演示了如何使用 `describe-snapshot-attribute`。

## AWS CLI

### 描述快照的快照属性

以下describe-snapshot-attribute示例列出了与之共享快照的账户。

```
aws ec2 describe-snapshot-attribute \  
  --snapshot-id snap-01234567890abcdef \  
  --attribute createVolumePermission
```

输出：

```
{  
  "SnapshotId": "snap-01234567890abcdef",  
  "CreateVolumePermissions": [  
    {  
      "UserId": "123456789012"  
    }  
  ]  
}
```

有关更多信息，请参阅[亚马逊弹性计算云用户指南中的共享 Amazon EBS 快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSnapshotAttribute](#)中的。

## describe-snapshot-tier-status

以下代码示例演示了如何使用 describe-snapshot-tier-status。

### AWS CLI

#### 查看有关已存档快照的存档信息

以下describe-snapshot-tier-status示例提供了有关已存档快照的存档信息。

```
aws ec2 describe-snapshot-tier-status \  
  --filters "Name=snapshot-id, Values=snap-01234567890abcdef"
```

输出：

```
{  
  "SnapshotTierStatuses": [  
    {
```

```
    "Status": "completed",
    "ArchivalCompleteTime": "2021-09-15T17:33:16.147Z",
    "LastTieringProgress": 100,
    "Tags": [],
    "VolumeId": "vol-01234567890abcdef",
    "LastTieringOperationState": "archival-completed",
    "StorageTier": "archive",
    "OwnerId": "123456789012",
    "SnapshotId": "snap-01234567890abcdef",
    "LastTieringStartTime": "2021-09-15T16:44:37.574Z"
  }
]
}
```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的[查看已存档快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSnapshotTierStatus](#)中的。

## describe-snapshots

以下代码示例演示了如何使用 describe-snapshots。

### AWS CLI

#### 示例 1：描述快照

以下 describe-snapshots 示例描述了指定的快照。

```
aws ec2 describe-snapshots \
  --snapshot-ids snap-1234567890abcdef0
```

输出：

```
{
  "Snapshots": [
    {
      "Description": "This is my snapshot",
      "Encrypted": false,
      "VolumeId": "vol-049df61146c4d7901",
      "State": "completed",
      "VolumeSize": 8,
      "StartTime": "2019-02-28T21:28:32.000Z",
      "Progress": "100%",
```



```
    "OwnerId": "012345678910",
    "SnapshotId": "snap-01234567890abcdef",
    "Tags": [
      {
        "Key": "Stack",
        "Value": "test"
      }
    ]
  }
]
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的 [Amazon EBS 快照](#)。

### 示例 2：描述基于筛选器的快照

以下 describe-snapshots 示例使用筛选条件将结果范围限定为您的 AWS 账户拥有的处于该 pending 状态的快照。示例使用 --query 参数仅显示快照 ID 和快照的启动时间。

```
aws ec2 describe-snapshots \
  --owner-ids self \
  --filters Name=status,Values=pending \
  --query "Snapshots[*].{ID:SnapshotId,Time:StartTime}"
```

输出：

```
[
  {
    "ID": "snap-1234567890abcdef0",
    "Time": "2019-08-04T12:48:18.000Z"
  },
  {
    "ID": "snap-066877671789bd71b",
    "Time": "2019-08-04T02:45:16.000Z"
  },
  ...
]
```

以下 describe-snapshots 示例使用筛选器，将结果范围限定为从指定卷创建的快照。示例使用 --query 参数仅显示快照 ID。

```
aws ec2 describe-snapshots \
```

```
--filters Name=volume-id,Values=049df61146c4d7901 \  
--query "Snapshots[*].[SnapshotId]" \  
--output text
```

输出：

```
snap-1234567890abcdef0  
snap-08637175a712c3fb9  
...
```

有关使用筛选器的其他示例，请参阅《Amazon EC2 用户指南》中的[列出和筛选资源](#)。

### 示例 3：根据标签描述快照

以下 `describe-snapshots` 示例使用标签筛选器，将结果范围限定为带有标签 `Stack=Prod` 的快照。

```
aws ec2 describe-snapshots \  
  --filters Name=tag:Stack,Values=prod
```

有关 `describe-snapshots` 的输出示例，请参阅示例 1。

有关使用标签筛选器的更多示例，请参阅《Amazon EC2 用户指南》中的[使用标签](#)。

### 示例 4：根据期限描述快照

以下 `describe-snapshots` 示例使用 JMESPath 表达式来描述您的 AWS 账户在指定日期之前创建的所有快照。仅显示快照 ID。

```
aws ec2 describe-snapshots \  
  --owner-ids 012345678910 \  
  --query "Snapshots[?(StartTime<='2020-03-31')].[SnapshotId]"
```

有关使用筛选器的其他示例，请参阅《Amazon EC2 用户指南》中的[列出和筛选资源](#)。

### 示例 5：仅查看存档的快照

以下 `describe-snapshots` 示例列出归档层中存储的快照。

```
aws ec2 describe-snapshots \  
  --filters "Name=storage-tier,Values=archive"
```

输出：

```
{
  "Snapshots": [
    {
      "Description": "Snap A",
      "Encrypted": false,
      "VolumeId": "vol-01234567890aaaaaa",
      "State": "completed",
      "VolumeSize": 8,
      "StartTime": "2021-09-07T21:00:00.000Z",
      "Progress": "100%",
      "OwnerId": "123456789012",
      "SnapshotId": "snap-01234567890aaaaaa",
      "StorageTier": "archive",
      "Tags": []
    },
  ]
}
```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的[查看已存档快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSnapshots](#)中的。

## describe-spot-datafeed-subscription

以下代码示例演示了如何使用 describe-spot-datafeed-subscription。

### AWS CLI

描述账户的竞价型实例数据源订阅

此示例命令描述了账户的数据馈送。

命令：

```
aws ec2 describe-spot-datafeed-subscription
```

输出：

```
{
  "SpotDatafeedSubscription": {
    "OwnerId": "123456789012",
```

```
    "Prefix": "spotdata",
    "Bucket": "my-s3-bucket",
    "State": "Active"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSpotDatafeedSubscription](#)中的。

## describe-spot-fleet-instances

以下代码示例演示了如何使用 describe-spot-fleet-instances。

### AWS CLI

描述与竞价型队列关联的竞价型实例

此示例命令列出了与指定竞价型队列关联的竞价型实例。

命令:

```
aws ec2 describe-spot-fleet-instances --spot-fleet-request-id sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

输出:

```
{
  "ActiveInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "InstanceType": "m3.medium",
      "SpotInstanceRequestId": "sir-08b93456"
    },
    ...
  ],
  "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSpotFleetInstances](#)中的。

## describe-spot-fleet-request-history

以下代码示例演示了如何使用 describe-spot-fleet-request-history。

## AWS CLI

### 描述 Spot 舰队的历史

此示例命令返回指定竞价队列从指定时间开始的历史记录。

命令:

```
aws ec2 describe-spot-fleet-request-history --spot-fleet-request-id sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE --start-time 2015-05-26T00:00:00Z
```

以下示例输出显示了竞价型队列的两个竞价型实例的成功启动。

输出:

```
{
  "HistoryRecords": [
    {
      "Timestamp": "2015-05-26T23:17:20.697Z",
      "EventInformation": {
        "EventSubType": "submitted"
      },
      "EventType": "fleetRequestChange"
    },
    {
      "Timestamp": "2015-05-26T23:17:20.873Z",
      "EventInformation": {
        "EventSubType": "active"
      },
      "EventType": "fleetRequestChange"
    },
    {
      "Timestamp": "2015-05-26T23:21:21.712Z",
      "EventInformation": {
        "InstanceId": "i-1234567890abcdef0",
        "EventSubType": "launched"
      },
      "EventType": "instanceChange"
    },
    {
      "Timestamp": "2015-05-26T23:21:21.816Z",
      "EventInformation": {
        "InstanceId": "i-1234567890abcdef1",
        "EventSubType": "launched"
      }
    }
  ]
}
```

```

    },
    "EventType": "instanceChange"
  }
],
"SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
"NextToken": "CpHNsscimcV5oH7bSbub03CI2Qms5+ypNpNm
+53MN1R0YcXAkp0xFlfKf91yVxSExmbtma3awYxMFzNA663ZskT0AHtJ6TCb2Z8bQC2EnZgyELbymtWPfpZ1ZbauVg
+P+TfG1WxWWB/Vr5dk5d4LfdgA/DRAHUrYgxzrEXAMPLE=",
"StartTime": "2015-05-26T00:00:00Z"
}

```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeSpotFleetRequestHistory](#) 中的。

## describe-spot-fleet-requests

以下代码示例演示了如何使用 describe-spot-fleet-requests。

### AWS CLI

描述您的 Spot 队列请求

此示例描述了您的所有 Spot 队列请求。

命令:

```
aws ec2 describe-spot-fleet-requests
```

输出:

```

{
  "SpotFleetRequestConfigs": [
    {
      "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
      "SpotFleetRequestConfig": {
        "TargetCapacity": 20,
        "LaunchSpecifications": [
          {
            "EbsOptimized": false,
            "NetworkInterfaces": [
              {
                "SubnetId": "subnet-a61dafcf",
                "DeviceIndex": 0,
                "DeleteOnTermination": false,

```

```
        "AssociatePublicIpAddress": true,
        "SecondaryPrivateIpAddressCount": 0
      }
    ],
    "InstanceType": "cc2.8xlarge",
    "ImageId": "ami-1a2b3c4d"
  },
  {
    "EbsOptimized": false,
    "NetworkInterfaces": [
      {
        "SubnetId": "subnet-a61dafcf",
        "DeviceIndex": 0,
        "DeleteOnTermination": false,
        "AssociatePublicIpAddress": true,
        "SecondaryPrivateIpAddressCount": 0
      }
    ],
    "InstanceType": "r3.8xlarge",
    "ImageId": "ami-1a2b3c4d"
  }
],
"SpotPrice": "0.05",
"IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role"
},
"SpotFleetRequestState": "active"
},
{
  "SpotFleetRequestId": "sfr-306341ed-9739-402e-881b-ce47bEXAMPLE",
  "SpotFleetRequestConfig": {
    "TargetCapacity": 20,
    "LaunchSpecifications": [
      {
        "EbsOptimized": false,
        "NetworkInterfaces": [
          {
            "SubnetId": "subnet-6e7f829e",
            "DeviceIndex": 0,
            "DeleteOnTermination": false,
            "AssociatePublicIpAddress": true,
            "SecondaryPrivateIpAddressCount": 0
          }
        ],
        "InstanceType": "m3.medium",
```

```

        "ImageId": "ami-1a2b3c4d"
      }
    ],
    "SpotPrice": "0.05",
    "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role"
  },
  "SpotFleetRequestState": "active"
}
]
}

```

## 描述 Spot 队列请求

此示例描述了指定的 Spot 队列请求。

### 命令:

```
aws ec2 describe-spot-fleet-requests --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

### 输出:

```

{
  "SpotFleetRequestConfigs": [
    {
      "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
      "SpotFleetRequestConfig": {
        "TargetCapacity": 20,
        "LaunchSpecifications": [
          {
            "EbsOptimized": false,
            "NetworkInterfaces": [
              {
                "SubnetId": "subnet-a61dafcf",
                "DeviceIndex": 0,
                "DeleteOnTermination": false,
                "AssociatePublicIpAddress": true,
                "SecondaryPrivateIpAddressCount": 0
              }
            ],
            "InstanceType": "cc2.8xlarge",
            "ImageId": "ami-1a2b3c4d"
          }
        ]
      }
    }
  ]
}

```



```

        {
            "EbsOptimized": false,
            "NetworkInterfaces": [
                {
                    "SubnetId": "subnet-a61dafcf",
                    "DeviceIndex": 0,
                    "DeleteOnTermination": false,
                    "AssociatePublicIpAddress": true,
                    "SecondaryPrivateIpAddressCount": 0
                }
            ],
            "InstanceType": "r3.8xlarge",
            "ImageId": "ami-1a2b3c4d"
        }
    ],
    "SpotPrice": "0.05",
    "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role"
},
"SpotFleetRequestState": "active"
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSpotFleetRequests](#)中的。

## describe-spot-instance-requests

以下代码示例演示了如何使用 describe-spot-instance-requests。

### AWS CLI

#### 示例 1：描述竞价型实例请求

以下describe-spot-instance-requests示例描述了指定的竞价型实例请求。

```
aws ec2 describe-spot-instance-requests \
  --spot-instance-request-ids sir-08b93456
```

输出：

```
{
  "SpotInstanceRequests": [
    {
```

```
"CreateTime": "2018-04-30T18:14:55.000Z",
"InstanceId": "i-1234567890abcdef1",
"LaunchSpecification": {
  "InstanceType": "t2.micro",
  "ImageId": "ami-003634241a8fcdec0",
  "KeyName": "my-key-pair",
  "SecurityGroups": [
    {
      "GroupName": "default",
      "GroupId": "sg-e38f24a7"
    }
  ],
  "BlockDeviceMappings": [
    {
      "DeviceName": "/dev/sda1",
      "Ebs": {
        "DeleteOnTermination": true,
        "SnapshotId": "snap-0e54a519c999adbbd",
        "VolumeSize": 8,
        "VolumeType": "standard",
        "Encrypted": false
      }
    }
  ],
  "NetworkInterfaces": [
    {
      "DeleteOnTermination": true,
      "DeviceIndex": 0,
      "SubnetId": "subnet-049df61146c4d7901"
    }
  ],
  "Placement": {
    "AvailabilityZone": "us-east-2b",
    "Tenancy": "default"
  },
  "Monitoring": {
    "Enabled": false
  }
},
"LaunchedAvailabilityZone": "us-east-2b",
"ProductDescription": "Linux/UNIX",
"SpotInstanceRequestId": "sir-08b93456",
"SpotPrice": "0.010000"
"State": "active",
```

```

        "Status": {
            "Code": "fulfilled",
            "Message": "Your Spot request is fulfilled.",
            "UpdateTime": "2018-04-30T18:16:21.000Z"
        },
        "Tags": [],
        "Type": "one-time",
        "InstanceInterruptionBehavior": "terminate"
    }
]
}

```

### 示例 2：描述基于筛选条件的竞价型实例请求

以下 `describe-spot-instance-requests` 示例使用筛选条件将结果限定为指定可用区内具有指定实例类型的竞价型实例请求。该示例使用 `--query` 参数仅显示实例 ID。

```

aws ec2 describe-spot-instance-requests \
  --filters Name=launch.instance-type,Values=m3.medium Name=launched-availability-
  zone,Values=us-east-2a \
  --query "SpotInstanceRequests[*].[InstanceId]" \
  --output text

```

输出：

```

i-057750d42936e468a
i-001efd250faaa6ffa
i-027552a73f021f3bd
...

```

有关使用筛选条件的其他示例，请参阅《亚马逊弹性计算云用户指南》中的 [列出和筛选您的资源](#)。

### 示例 3：描述基于标签的竞价型实例请求

以下 `describe-spot-instance-requests` 示例使用标签筛选器将结果范围限定为带有该标签的竞价型实例请求 `cost-center=cc123`。

```

aws ec2 describe-spot-instance-requests \
  --filters Name=tag:cost-center,Values=cc123

```

有关 `describe-spot-instance-requests` 的输出示例，请参阅示例 1。

有关使用标签筛选器的更多示例，请参阅《Amazon EC2 用户指南》中的[使用标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSpotInstanceRequests](#)中的。

## describe-spot-price-history

以下代码示例演示了如何使用 describe-spot-price-history。

### AWS CLI

#### 描述现货价格历史记录

此示例命令返回 m1.xlarge 实例在一月份特定日期的竞价价格历史记录。

命令：

```
aws ec2 describe-spot-price-history --instance-types m1.xlarge --start-time
2014-01-06T07:08:09 --end-time 2014-01-06T08:09:10
```

输出：

```
{
  "SpotPriceHistory": [
    {
      "Timestamp": "2014-01-06T07:10:55.000Z",
      "ProductDescription": "SUSE Linux",
      "InstanceType": "m1.xlarge",
      "SpotPrice": "0.087000",
      "AvailabilityZone": "us-west-1b"
    },
    {
      "Timestamp": "2014-01-06T07:10:55.000Z",
      "ProductDescription": "SUSE Linux",
      "InstanceType": "m1.xlarge",
      "SpotPrice": "0.087000",
      "AvailabilityZone": "us-west-1c"
    },
    {
      "Timestamp": "2014-01-06T05:42:36.000Z",
      "ProductDescription": "SUSE Linux (Amazon VPC)",
      "InstanceType": "m1.xlarge",
      "SpotPrice": "0.087000",
      "AvailabilityZone": "us-west-1a"
    }
  ]
}
```

```
    },  
    ...  
}
```

描述 Linux/UNIX 亚马逊 VPC 的现货价格历史记录

此示例命令返回 m1.xlarge、Linux/UNIX Amazon VPC 实例在一月份特定日期的现货价格历史记录。

命令:

```
aws ec2 describe-spot-price-history --instance-types m1.xlarge --product-  
description "Linux/UNIX (Amazon VPC)" --start-time 2014-01-06T07:08:09 --end-time  
2014-01-06T08:09:10
```

输出:

```
{  
  "SpotPriceHistory": [  
    {  
      "Timestamp": "2014-01-06T04:32:53.000Z",  
      "ProductDescription": "Linux/UNIX (Amazon VPC)",  
      "InstanceType": "m1.xlarge",  
      "SpotPrice": "0.080000",  
      "AvailabilityZone": "us-west-1a"  
    },  
    {  
      "Timestamp": "2014-01-05T11:28:26.000Z",  
      "ProductDescription": "Linux/UNIX (Amazon VPC)",  
      "InstanceType": "m1.xlarge",  
      "SpotPrice": "0.080000",  
      "AvailabilityZone": "us-west-1c"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSpotPriceHistory](#)中的。

## describe-stale-security-groups

以下代码示例演示了如何使用 describe-stale-security-groups。

## AWS CLI

### 描述陈旧的安全组

此示例描述了过时的安全组规则。vpc-11223344 响应显示，您的账户中的 sg-5fa68d3a 有一条在对等 VPC 中引用的过时入口 SSH 规则，而您的账户中 sg-279ab042 有一条在对等 VPC sg-fe6fba9a 中引用的过时出口 SSH 规则。sg-ef6fba8b

命令:

```
aws ec2 describe-stale-security-groups --vpc-id vpc-11223344
```

输出:

```
{
  "StaleSecurityGroupSet": [
    {
      "VpcId": "vpc-11223344",
      "StaleIpPermissionsEgress": [
        {
          "ToPort": 22,
          "FromPort": 22,
          "UserIdGroupPairs": [
            {
              "VpcId": "vpc-7a20e51f",
              "GroupId": "sg-ef6fba8b",
              "VpcPeeringConnectionId": "pcx-b04deed9",
              "PeeringStatus": "active"
            }
          ]
        },
        {
          "IpProtocol": "tcp"
        }
      ],
      "GroupName": "MySG1",
      "StaleIpPermissions": [],
      "GroupId": "sg-fe6fba9a",
      "Description": "MySG1"
    },
    {
      "VpcId": "vpc-11223344",
      "StaleIpPermissionsEgress": [],
      "GroupName": "MySG2",
      "StaleIpPermissions": [
```

```
    {
      "ToPort": 22,
      "FromPort": 22,
      "UserIdGroupPairs": [
        {
          "VpcId": "vpc-7a20e51f",
          "GroupId": "sg-279ab042",
          "Description": "Access from pcx-b04deed9",
          "VpcPeeringConnectionId": "pcx-b04deed9",
          "PeeringStatus": "active"
        }
      ],
      "IpProtocol": "tcp"
    }
  ],
  "GroupId": "sg-5fa68d3a",
  "Description": "MySG2"
}
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStaleSecurityGroups](#)中的。

## describe-store-image-tasks

以下代码示例演示了如何使用 describe-store-image-tasks。

### AWS CLI

描述 AMI 存储任务的进度

以下describe-store-image-tasks示例描述了 AMI 存储任务的进度。

```
aws ec2 describe-store-image-tasks
```

输出：

```
{
  "AmiId": "ami-1234567890abcdef0",
  "Bucket": "my-ami-bucket",
  "ProgressPercentage": 17,
  "S3ObjectKey": "ami-1234567890abcdef0.bin",
```

```
"StoreTaskState": "InProgress",
"StoreTaskFailureReason": null,
"TaskStartTime": "2022-01-01T01:01:01.001Z"
}
```

有关使用 S3 存储和恢复 AMI 的更多信息，请参阅亚马逊 EC2 用户指南中的使用 S3 存储和恢复 AMI < [https://docs.aws.amazon.com/AWS\\_ec2/latest/UserGuide/ami-store-restore.html](https://docs.aws.amazon.com/AWS_ec2/latest/UserGuide/ami-store-restore.html)>。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeStoreImageTasks](#) 中的。

## describe-subnets

以下代码示例演示了如何使用 describe-subnets。

### AWS CLI

示例 1：描述所有子网

以下 describe-subnets 示例显示子网的详细信息。

```
aws ec2 describe-subnets
```

输出：

```
{
  "Subnets": [
    {
      "AvailabilityZone": "us-east-1d",
      "AvailabilityZoneId": "use1-az2",
      "AvailableIpAddressCount": 4089,
      "CidrBlock": "172.31.80.0/20",
      "DefaultForAz": true,
      "MapPublicIpOnLaunch": false,
      "MapCustomerOwnedIpOnLaunch": true,
      "State": "available",
      "SubnetId": "subnet-0bb1c79de3EXAMPLE",
      "VpcId": "vpc-0ee975135dEXAMPLE",
      "OwnerId": "111122223333",
      "AssignIpv6AddressOnCreation": false,
      "Ipv6CidrBlockAssociationSet": [],
      "CustomerOwnedIpv4Pool": "pool-2EXAMPLE",
      "SubnetArn": "arn:aws:ec2:us-east-2:111122223333:subnet/
subnet-0bb1c79de3EXAMPLE",
```



```

    "EnableDns64": false,
    "Ipv6Native": false,
    "PrivateDnsNameOptionsOnLaunch": {
      "HostnameType": "ip-name",
      "EnableResourceNameDnsARecord": false,
      "EnableResourceNameDnsAAAARecord": false
    }
  },
  {
    "AvailabilityZone": "us-east-1d",
    "AvailabilityZoneId": "use1-az2",
    "AvailableIpAddressCount": 4089,
    "CidrBlock": "172.31.80.0/20",
    "DefaultForAz": true,
    "MapPublicIpOnLaunch": true,
    "MapCustomerOwnedIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-8EXAMPLE",
    "VpcId": "vpc-3EXAMPLE",
    "OwnerId": "1111222233333",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "Tags": [
      {
        "Key": "Name",
        "Value": "MySubnet"
      }
    ],
    "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/
subnet-8EXAMPLE",
    "EnableDns64": false,
    "Ipv6Native": false,
    "PrivateDnsNameOptionsOnLaunch": {
      "HostnameType": "ip-name",
      "EnableResourceNameDnsARecord": false,
      "EnableResourceNameDnsAAAARecord": false
    }
  }
]
}

```

有关更多信息，请参阅《AWS VPC 用户指南》中的[使用 VPC 和子网](#)。

示例 2：描述特定 VPC 的子网

以下 `describe-subnets` 示例使用筛选条件检索指定 VPC 的子网的详细信息。

```
aws ec2 describe-subnets \  
  --filters "Name=vpc-id,Values=vpc-3EXAMPLE"
```

输出：

```
{  
  "Subnets": [  
    {  
      "AvailabilityZone": "us-east-1d",  
      "AvailabilityZoneId": "use1-az2",  
      "AvailableIpAddressCount": 4089,  
      "CidrBlock": "172.31.80.0/20",  
      "DefaultForAz": true,  
      "MapPublicIpOnLaunch": true,  
      "MapCustomerOwnedIpOnLaunch": false,  
      "State": "available",  
      "SubnetId": "subnet-8EXAMPLE",  
      "VpcId": "vpc-3EXAMPLE",  
      "OwnerId": "111122223333",  
      "AssignIpv6AddressOnCreation": false,  
      "Ipv6CidrBlockAssociationSet": [],  
      "Tags": [  
        {  
          "Key": "Name",  
          "Value": "MySubnet"  
        }  
      ],  
      "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/  
subnet-8EXAMPLE",  
      "EnableDns64": false,  
      "Ipv6Native": false,  
      "PrivateDnsNameOptionsOnLaunch": {  
        "HostnameType": "ip-name",  
        "EnableResourceNameDnsARecord": false,  
        "EnableResourceNameDnsAAAARecord": false  
      }  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS VPC 用户指南》中的[使用 VPC 和子网](#)。

### 示例 3：描述带有特定标签的子网

以下 `describe-subnets` 示例使用筛选器检索带有标签 `CostCenter=123` 的子网的详细信息，并使用 `--query` 参数显示带有此标签子网的子网 ID。

```
aws ec2 describe-subnets \
  --filters "Name=tag:CostCenter,Values=123" \
  --query "Subnets[*].SubnetId" \
  --output text
```

输出：

```
subnet-0987a87c8b37348ef
subnet-02a95061c45f372ee
subnet-03f720e7de2788d73
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的[使用 VPC 和子网](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeSubnets](#) 中的。

## describe-tags

以下代码示例演示了如何使用 `describe-tags`。

### AWS CLI

#### 示例 1：描述单个资源的所有标签

以下 `describe-tags` 示例描述了指定实例的标签。

```
aws ec2 describe-tags \
  --filters "Name=resource-id,Values=i-1234567890abcdef8"
```

输出：

```
{
  "Tags": [
    {
      "ResourceType": "instance",
      "ResourceId": "i-1234567890abcdef8",
      "Value": "Test",
```

```
        "Key": "Stack"
      },
      {
        "ResourceType": "instance",
        "ResourceId": "i-1234567890abcdef8",
        "Value": "Beta Server",
        "Key": "Name"
      }
    ]
  }
}
```

### 示例 2：描述资源类型的所有标签

以下describe-tags示例描述了您的卷的标签。

```
aws ec2 describe-tags \
  --filters "Name=resource-type,Values=volume"
```

输出：

```
{
  "Tags": [
    {
      "ResourceType": "volume",
      "ResourceId": "vol-1234567890abcdef0",
      "Value": "Project1",
      "Key": "Purpose"
    },
    {
      "ResourceType": "volume",
      "ResourceId": "vol-049df61146c4d7901",
      "Value": "Logs",
      "Key": "Purpose"
    }
  ]
}
```

### 示例 3：描述您的所有标签

以下describe-tags示例描述了您的所有资源的标签。

```
aws ec2 describe-tags
```

#### 示例 4：根据标签密钥描述资源的标签

以下describe-tags示例描述了带有密钥标签的资源的标签Stack。

```
aws ec2 describe-tags \  
  --filters Name=key,Values=Stack
```

输出：

```
{  
  "Tags": [  
    {  
      "ResourceType": "volume",  
      "ResourceId": "vol-027552a73f021f3b",  
      "Value": "Production",  
      "Key": "Stack"  
    },  
    {  
      "ResourceType": "instance",  
      "ResourceId": "i-1234567890abcdef8",  
      "Value": "Test",  
      "Key": "Stack"  
    }  
  ]  
}
```

#### 示例 5：根据标签键和标签值描述资源的标签

以下describe-tags示例描述了带有该标签的资源的标签Stack=Test。

```
aws ec2 describe-tags \  
  --filters Name=key,Values=Stack Name=value,Values=Test
```

输出：

```
{  
  "Tags": [  
    {  
      "ResourceType": "image",  
      "ResourceId": "ami-3ac336533f021f3bd",  
      "Value": "Test",  
      "Key": "Stack"  
    }  
  ]  
}
```

```

        "Key": "Stack"
      },
      {
        "ResourceType": "instance",
        "ResourceId": "i-1234567890abcdef8",
        "Value": "Test",
        "Key": "Stack"
      }
    ]
  }
}

```

以下describe-tags示例使用替代语法来描述带有标签的资源Stack=Test。

```

aws ec2 describe-tags \
  --filters "Name=tag:Stack,Values=Test"

```

以下describe-tags示例描述了所有带有密钥但没有值的标签的实例Purpose的标签。

```

aws ec2 describe-tags \
  --filters "Name=resource-type,Values=instance" "Name=key,Values=Purpose"
  "Name=value,Values="

```

输出：

```

{
  "Tags": [
    {
      "ResourceType": "instance",
      "ResourceId": "i-1234567890abcdef5",
      "Value": null,
      "Key": "Purpose"
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTags](#)中的。

## describe-traffic-mirror-filters

以下代码示例演示了如何使用 describe-traffic-mirror-filters。

## AWS CLI

### 查看您的交通镜像过滤器

以下describe-traffic-mirror-filters示例显示了所有流量镜像过滤器的详细信息。

```
aws ec2 describe-traffic-mirror-filters
```

输出：

```
{
  "TrafficMirrorFilters": [
    {
      "TrafficMirrorFilterId": "tmf-0293f26e86EXAMPLE",
      "IngressFilterRules": [
        {
          "TrafficMirrorFilterRuleId": "tmfr-0ca76e0e08EXAMPLE",
          "TrafficMirrorFilterId": "tmf-0293f26e86EXAMPLE",
          "TrafficDirection": "ingress",
          "RuleNumber": 100,
          "RuleAction": "accept",
          "Protocol": 6,
          "DestinationCidrBlock": "10.0.0.0/24",
          "SourceCidrBlock": "10.0.0.0/24",
          "Description": "TCP Rule"
        }
      ],
      "EgressFilterRules": [],
      "NetworkServices": [],
      "Description": "Example filter",
      "Tags": []
    }
  ]
}
```

有关更多信息，请参阅《[流量镜像指南](#)》中的[查看流量镜像过滤器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTrafficMirrorFilters](#)中的。

## describe-traffic-mirror-sessions

以下代码示例演示了如何使用 describe-traffic-mirror-sessions。

## AWS CLI

### 描述流量镜像会话

以下describe-traffic-mirror-sessions示例显示了您的流量镜像会话的详细信息。

```
aws ec2 describe-traffic-mirror-sessions
```

输出：

```
{
  "TrafficMirrorSessions": [
    {
      "Tags": [],
      "VirtualNetworkId": 42,
      "OwnerId": "111122223333",
      "Description": "TCP Session",
      "NetworkInterfaceId": "eni-0a471a5cf3EXAMPLE",
      "TrafficMirrorTargetId": "tmt-0dabe9b0a6EXAMPLE",
      "TrafficMirrorFilterId": "tmf-083e18f985EXAMPLE",
      "PacketLength": 20,
      "SessionNumber": 1,
      "TrafficMirrorSessionId": "tms-0567a4c684EXAMPLE"
    },
    {
      "Tags": [
        {
          "Key": "Name",
          "Value": "tag test"
        }
      ],
      "VirtualNetworkId": 13314501,
      "OwnerId": "111122223333",
      "Description": "TCP Session",
      "NetworkInterfaceId": "eni-0a471a5cf3EXAMPLE",
      "TrafficMirrorTargetId": "tmt-03665551cbEXAMPLE",
      "TrafficMirrorFilterId": "tmf-06c787846cEXAMPLE",
      "SessionNumber": 2,
      "TrafficMirrorSessionId": "tms-0060101cf8EXAMPLE"
    }
  ]
}
```



有关更多信息，请参阅 [《流量镜像指南》](#) 中的“[查看AWS 流量镜像会话详细信息](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTrafficMirrorSessions](#)中的。

## describe-traffic-mirror-targets

以下代码示例演示了如何使用 describe-traffic-mirror-targets。

### AWS CLI

#### 描述流量镜像目标

以下describe-traffic-mirror-targets示例显示有关指定流量镜像目标的信息。

```
aws ec2 describe-traffic-mirror-targets \  
  --traffic-mirror-target-ids tmt-0dabe9b0a6EXAMPLE
```

输出：

```
{  
  "TrafficMirrorTargets": [  
    {  
      "TrafficMirrorTargetId": "tmt-0dabe9b0a6EXAMPLE",  
      "NetworkLoadBalancerArn": "arn:aws:elasticloadbalancing:us-  
east-1:111122223333:loadbalancer/net/NLB/7cdec873fEXAMPLE",  
      "Type": "network-load-balancer",  
      "Description": "Example Network Load Balancer target",  
      "OwnerId": "111122223333",  
      "Tags": []  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon VPC 流量镜像指南中的[流量镜像目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTrafficMirrorTargets](#)中的。

## describe-transit-gateway-attachments

以下代码示例演示了如何使用 describe-transit-gateway-attachments。

## AWS CLI

### 查看您的公交网关附件

以下describe-transit-gateway-attachments示例显示您的公交网关附件的详细信息。

```
aws ec2 describe-transit-gateway-attachments
```

输出：

```
{
  "TransitGatewayAttachments": [
    {
      "TransitGatewayAttachmentId": "tgw-attach-01f8100bc7EXAMPLE",
      "TransitGatewayId": "tgw-02f776b1a7EXAMPLE",
      "TransitGatewayOwnerId": "123456789012",
      "ResourceOwnerId": "123456789012",
      "ResourceType": "vpc",
      "ResourceId": "vpc-3EXAMPLE",
      "State": "available",
      "Association": {
        "TransitGatewayRouteTableId": "tgw-rtb-002573ed1eEXAMPLE",
        "State": "associated"
      },
      "CreationTime": "2019-08-26T14:59:25.000Z",
      "Tags": [
        {
          "Key": "Name",
          "Value": "Example"
        }
      ]
    },
    {
      "TransitGatewayAttachmentId": "tgw-attach-0b5968d3b6EXAMPLE",
      "TransitGatewayId": "tgw-02f776b1a7EXAMPLE",
      "TransitGatewayOwnerId": "123456789012",
      "ResourceOwnerId": "123456789012",
      "ResourceType": "vpc",
      "ResourceId": "vpc-0065acced4EXAMPLE",
      "State": "available",
      "Association": {
        "TransitGatewayRouteTableId": "tgw-rtb-002573ed1eEXAMPLE",
        "State": "associated"
      }
    }
  ]
}
```

```

    },
    "CreationTime": "2019-08-07T17:03:07.000Z",
    "Tags": []
  },
  {
    "TransitGatewayAttachmentId": "tgw-attach-08e0bc912cEXAMPLE",
    "TransitGatewayId": "tgw-02f776b1a7EXAMPLE",
    "TransitGatewayOwnerId": "123456789012",
    "ResourceOwnerId": "123456789012",
    "ResourceType": "direct-connect-gateway",
    "ResourceId": "11460968-4ac1-4fd3-bdb2-00599EXAMPLE",
    "State": "available",
    "Association": {
      "TransitGatewayRouteTableId": "tgw-rtb-002573ed1eEXAMPLE",
      "State": "associated"
    },
    "CreationTime": "2019-08-14T20:27:44.000Z",
    "Tags": []
  },
  {
    "TransitGatewayAttachmentId": "tgw-attach-0a89069f57EXAMPLE",
    "TransitGatewayId": "tgw-02f776b1a7EXAMPLE",
    "TransitGatewayOwnerId": "123456789012",
    "ResourceOwnerId": "123456789012",
    "ResourceType": "direct-connect-gateway",
    "ResourceId": "8384da05-13ce-4a91-aada-5a1baEXAMPLE",
    "State": "available",
    "Association": {
      "TransitGatewayRouteTableId": "tgw-rtb-002573ed1eEXAMPLE",
      "State": "associated"
    },
    "CreationTime": "2019-08-14T20:33:02.000Z",
    "Tags": []
  }
]
}

```

有关更多信息，请参阅 [《公网网关指南》中的使用](#) 中转网关。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeTransitGatewayAttachments](#) 中的。

## describe-transit-gateway-connect-peers

以下代码示例演示了如何使用 describe-transit-gateway-connect-peers。

### AWS CLI

描述 Transit Gateway Connect 对等方

以下describe-transit-gateway-connect-peers示例描述了指定的 Connect 对等体。

```
aws ec2 describe-transit-gateway-connect-peers \  
  --transit-gateway-connect-peer-ids tgw-connect-peer-0666adbac4EXAMPLE
```

输出：

```
{  
  "TransitGatewayConnectPeers": [  
    {  
      "TransitGatewayAttachmentId": "tgw-attach-0f0927767cEXAMPLE",  
      "TransitGatewayConnectPeerId": "tgw-connect-peer-0666adbac4EXAMPLE",  
      "State": "available",  
      "CreationTime": "2021-10-13T03:35:17.000Z",  
      "ConnectPeerConfiguration": {  
        "TransitGatewayAddress": "10.0.0.234",  
        "PeerAddress": "172.31.1.11",  
        "InsideCidrBlocks": [  
          "169.254.6.0/29"  
        ],  
        "Protocol": "gre",  
        "BgpConfigurations": [  
          {  
            "TransitGatewayAsn": 64512,  
            "PeerAsn": 64512,  
            "TransitGatewayAddress": "169.254.6.2",  
            "PeerAddress": "169.254.6.1",  
            "BgpStatus": "down"  
          },  
          {  
            "TransitGatewayAsn": 64512,  
            "PeerAsn": 64512,  
            "TransitGatewayAddress": "169.254.6.3",  
            "PeerAddress": "169.254.6.1",  
            "BgpStatus": "down"  
          }  
        ]  
      }  
    }  
  ]  
}
```

```

    }
  ],
  "Tags": []
}
]
}

```

有关更多信息，请参阅《公交网关指南》中的[公交网关 Connect 附件和 Transit Gateway Connect 对等体](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTransitGatewayConnectPeers](#)中的。

## describe-transit-gateway-connects

以下代码示例演示了如何使用 describe-transit-gateway-connects。

### AWS CLI

描述公交网关 Connect 附件

以下describe-transit-gateway-connects示例描述了指定的 Connect 附件。

```

aws ec2 describe-transit-gateway-connects \
  --transit-gateway-attachment-ids tgw-attach-037012e5dcEXAMPLE

```

输出：

```

{
  "TransitGatewayConnects": [
    {
      "TransitGatewayAttachmentId": "tgw-attach-037012e5dcEXAMPLE",
      "TransportTransitGatewayAttachmentId": "tgw-attach-0a89069f57EXAMPLE",
      "TransitGatewayId": "tgw-02f776b1a7EXAMPLE",
      "State": "available",
      "CreationTime": "2021-03-09T19:59:17+00:00",
      "Options": {
        "Protocol": "gre"
      },
      "Tags": []
    }
  ]
}

```

```
}
```

有关更多信息，请参阅《公交网关指南》中的[公交网关 Connect 附件和 Transit Gateway Connect 对等体](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTransitGatewayConnects](#)中的。

## describe-transit-gateway-multicast-domains

以下代码示例演示了如何使用 describe-transit-gateway-multicast-domains。

### AWS CLI

描述您的传输网关多播域

以下describe-transit-gateway-multicast-domains示例显示您的所有传输网关组播域的详细信息。

```
aws ec2 describe-transit-gateway-multicast-domains
```

输出：

```
{
  "TransitGatewayMulticastDomains": [
    {
      "TransitGatewayMulticastDomainId": "tgw-mcast-domain-000fb24d04EXAMPLE",
      "TransitGatewayId": "tgw-0bf0bffefaEXAMPLE",
      "TransitGatewayMulticastDomainArn": "arn:aws:ec2:us-east-1:123456789012:transit-gateway-multicast-domain/tgw-mcast-domain-000fb24d04EXAMPLE",
      "OwnerId": "123456789012",
      "Options": {
        "Icmpv2Support": "disable",
        "StaticSourcesSupport": "enable",
        "AutoAcceptSharedAssociations": "disable"
      },
      "State": "available",
      "CreationTime": "2019-12-10T18:32:50+00:00",
      "Tags": [
        {
```

```
        "Key": "Name",
        "Value": "mc1"
      }
    ]
  }
]
```

有关更多信息，请参阅《传输网关指南》中的[管理多播域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTransitGatewayMulticastDomains](#)中的。

## describe-transit-gateway-peering-attachments

以下代码示例演示了如何使用 describe-transit-gateway-peering-attachments。

### AWS CLI

描述您的公交网关对等连接附件

以下describe-transit-gateway-peering-attachments示例显示了所有公交网关对等连接附件的详细信息。

```
aws ec2 describe-transit-gateway-peering-attachments
```

输出：

```
{
  "TransitGatewayPeeringAttachments": [
    {
      "TransitGatewayAttachmentId": "tgw-attach-4455667788aabbccd",
      "RequesterTgwInfo": {
        "TransitGatewayId": "tgw-123abc05e04123abc",
        "OwnerId": "123456789012",
        "Region": "us-west-2"
      },
      "AcceptorTgwInfo": {
        "TransitGatewayId": "tgw-11223344aabbcc112",
        "OwnerId": "123456789012",
        "Region": "us-east-2"
      }
    }
  ]
}
```

```
    },
    "State": "pendingAcceptance",
    "CreationTime": "2019-12-09T11:38:05.000Z",
    "Tags": []
  }
]
}
```

有关更多信息，请参阅 [《公交网关指南》](#) 中的 [Transit Gateway 对等连接附件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeTransitGatewayPeeringAttachments](#) 中的。

## describe-transit-gateway-policy-tables

以下代码示例演示了如何使用 describe-transit-gateway-policy-tables。

### AWS CLI

描述传输网关策略表

以下 describe-transit-gateway-policy-tables 示例描述了指定的传输网关策略表。

```
aws ec2 describe-transit-gateway-policy-tables \
  --transit-gateway-policy-table-ids tgw-ptb-0a16f134b78668a81
```

输出：

```
{
  "TransitGatewayPolicyTables": [
    {
      "TransitGatewayPolicyTableId": "tgw-ptb-0a16f134b78668a81",
      "TransitGatewayId": "tgw-067f8505c18f0bd6e",
      "State": "available",
      "CreationTime": "2023-11-28T16:36:43+00:00",
      "Tags": []
    }
  ]
}
```

有关更多信息，请参阅 [Transit Gateway 用户指南中的公交网关策略表](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTransitGatewayPolicyTables](#)中的。

## describe-transit-gateway-route-tables

以下代码示例演示了如何使用 describe-transit-gateway-route-tables。

### AWS CLI

描述您的公交网关路由表

以下describe-transit-gateway-route-tables示例显示您的公交网关路由表的详细信息。

```
aws ec2 describe-transit-gateway-route-tables
```

输出：

```
{
  "TransitGatewayRouteTables": [
    {
      "TransitGatewayRouteTableId": "tgw-rtb-0ca78a549EXAMPLE",
      "TransitGatewayId": "tgw-0bc994abffEXAMPLE",
      "State": "available",
      "DefaultAssociationRouteTable": true,
      "DefaultPropagationRouteTable": true,
      "CreationTime": "2018-11-28T14:24:49.000Z",
      "Tags": []
    },
    {
      "TransitGatewayRouteTableId": "tgw-rtb-0e8f48f148EXAMPLE",
      "TransitGatewayId": "tgw-0043d72bb4EXAMPLE",
      "State": "available",
      "DefaultAssociationRouteTable": true,
      "DefaultPropagationRouteTable": true,
      "CreationTime": "2018-11-28T14:24:00.000Z",
      "Tags": []
    }
  ]
}
```

有关更多信息，请参阅《[公交网关指南](#)》中的[查看公交网关路由表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTransitGatewayRouteTables](#)中的。

## describe-transit-gateway-vpc-attachments

以下代码示例演示了如何使用 describe-transit-gateway-vpc-attachments。

### AWS CLI

描述您的传输网关 VPC 附件

以下 describe-transit-gateway-vpc-attachments 示例显示您的传输网关 VPC 附件的详细信息。

```
aws ec2 describe-transit-gateway-vpc-attachments
```

输出：

```
{
  "TransitGatewayVpcAttachments": [
    {
      "TransitGatewayAttachmentId": "tgw-attach-0a08e88308EXAMPLE",
      "TransitGatewayId": "tgw-0043d72bb4EXAMPLE",
      "VpcId": "vpc-0f501f7ee8EXAMPLE",
      "VpcOwnerId": "111122223333",
      "State": "available",
      "SubnetIds": [
        "subnet-045d586432EXAMPLE",
        "subnet-0a0ad478a6EXAMPLE"
      ],
      "CreationTime": "2019-02-13T11:04:02.000Z",
      "Options": {
        "DnsSupport": "enable",
        "Ipv6Support": "disable"
      },
      "Tags": [
        {
          "Key": "Name",
          "Value": "attachment name"
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅 [中转网关指南中的查看 VPC 附件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTransitGatewayVpcAttachments](#)中的。

## describe-transit-gateways

以下代码示例演示了如何使用 describe-transit-gateways。

### AWS CLI

描述您的中转网关

以下describe-transit-gateways示例检索有关您的中转网关的详细信息。

```
aws ec2 describe-transit-gateways
```

输出：

```
{
  "TransitGateways": [
    {
      "TransitGatewayId": "tgw-0262a0e521EXAMPLE",
      "TransitGatewayArn": "arn:aws:ec2:us-east-2:111122223333:transit-gateway/tgw-0262a0e521EXAMPLE",
      "State": "available",
      "OwnerId": "111122223333",
      "Description": "MyTGW",
      "CreationTime": "2019-07-10T14:02:12.000Z",
      "Options": {
        "AmazonSideAsn": 64516,
        "AutoAcceptSharedAttachments": "enable",
        "DefaultRouteTableAssociation": "enable",
        "AssociationDefaultRouteTableId": "tgw-rtb-018774adf3EXAMPLE",
        "DefaultRouteTablePropagation": "enable",
        "PropagationDefaultRouteTableId": "tgw-rtb-018774adf3EXAMPLE",
        "VpnEcmpSupport": "enable",
        "DnsSupport": "enable"
      },
      "Tags": []
    },
    {
      "TransitGatewayId": "tgw-0fb8421e2dEXAMPLE",
      "TransitGatewayArn": "arn:aws:ec2:us-east-2:111122223333:transit-gateway/tgw-0fb8421e2da853bf3",

```

```

    "State": "available",
    "OwnerId": "111122223333",
    "CreationTime": "2019-03-15T22:57:33.000Z",
    "Options": {
      "AmazonSideAsn": 65412,
      "AutoAcceptSharedAttachments": "disable",
      "DefaultRouteTableAssociation": "enable",
      "AssociationDefaultRouteTableId": "tgw-rtb-06a241a3d8EXAMPLE",
      "DefaultRouteTablePropagation": "enable",
      "PropagationDefaultRouteTableId": "tgw-rtb-06a241a3d8EXAMPLE",
      "VpnEcmpSupport": "enable",
      "DnsSupport": "enable"
    },
    "Tags": [
      {
        "Key": "Name",
        "Value": "TGW1"
      }
    ]
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTransitGateways](#)中的。

## describe-verified-access-endpoints

以下代码示例演示了如何使用 describe-verified-access-endpoints。

### AWS CLI

描述已验证访问终端节点

以下delete-verified-access-endpoints示例描述了指定的已验证访问终端节点。

```

aws ec2 describe-verified-access-endpoints \
  --verified-access-endpoint-ids vae-066fac616d4d546f2

```

输出：

```

{
  "VerifiedAccessEndpoints": [

```

```
{
  "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",
  "VerifiedAccessGroupId": "vagr-0dbe967baf14b7235",
  "VerifiedAccessEndpointId": "vae-066fac616d4d546f2",
  "ApplicationDomain": "example.com",
  "EndpointType": "network-interface",
  "AttachmentType": "vpc",
  "DomainCertificateArn": "arn:aws:acm:us-east-2:123456789012:certificate/
eb065ea0-26f9-4e75-a6ce-0a1a7EXAMPLE",
  "EndpointDomain": "my-ava-
app.edge-00c3372d53b1540bb.vai-0ce000c0b7643abea.prod.verified-access.us-
east-2.amazonaws.com",
  "SecurityGroupIds": [
    "sg-004915970c4c8f13a"
  ],
  "NetworkInterfaceOptions": {
    "NetworkInterfaceId": "eni-0aec70418c8d87a0f",
    "Protocol": "https",
    "Port": 443
  },
  "Status": {
    "Code": "active"
  },
  "Description": "",
  "CreationTime": "2023-08-25T20:54:43",
  "LastUpdatedTime": "2023-08-25T22:17:26",
  "Tags": [
    {
      "Key": "Name",
      "Value": "my-va-endpoint"
    }
  ]
}
```

有关更多信息，请参阅 [《已验证访问用户指南》](#) 中的 [AWS 已验证访问终端节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeVerifiedAccessEndpoints](#) 中的。

## describe-verified-access-groups

以下代码示例演示了如何使用 `describe-verified-access-groups`。

## AWS CLI

### 描述已验证访问权限组

以下describe-verified-access-groups示例描述了指定的已验证访问权限组。

```
aws ec2 describe-verified-access-groups \
  --verified-access-group-ids vagr-0dbe967baf14b7235
```

输出：

```
{
  "VerifiedAccessGroups": [
    {
      "VerifiedAccessGroupId": "vagr-0dbe967baf14b7235",
      "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",
      "Description": "Testing Verified Access",
      "Owner": "123456789012",
      "VerifiedAccessGroupArn": "arn:aws:ec2:us-east-2:123456789012:verified-
access-group/vagr-0dbe967baf14b7235",
      "CreationTime": "2023-08-25T19:55:19",
      "LastUpdatedTime": "2023-08-25T22:17:25",
      "Tags": [
        {
          "Key": "Name",
          "Value": "my-va-group"
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVerifiedAccessGroups](#)中的。

### describe-verified-access-instance-logging-configurations

以下代码示例演示了如何使用 describe-verified-access-instance-logging-configurations。

## AWS CLI

描述已验证访问权限实例的日志配置

以下describe-verified-access-instance-logging-configurations示例描述了指定的 Verified Access 实例的日志配置。

```
aws ec2 describe-verified-access-instance-logging-configurations \
  --verified-access-instance-ids vai-0ce000c0b7643abea
```

输出：

```
{
  "LoggingConfigurations": [
    {
      "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",
      "AccessLogs": {
        "S3": {
          "Enabled": false
        },
        "CloudWatchLogs": {
          "Enabled": true,
          "DeliveryStatus": {
            "Code": "success"
          },
          "LogGroup": "my-log-group"
        },
        "KinesisDataFirehose": {
          "Enabled": false
        },
        "LogVersion": "ocsf-1.0.0-rc.2",
        "IncludeTrustContext": false
      }
    }
  ]
}
```

有关更多信息，请参阅 [《已验证访问用户指南》](#) 中的 [AWS 已验证访问日志](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 DescribeVerifiedAccessInstanceLoggingConfigurations](#) 中的。

## describe-verified-access-instances

以下代码示例演示了如何使用 describe-verified-access-instances。

### AWS CLI

#### 描述已验证访问权限实例

以下describe-verified-access-instances示例描述了指定的已验证访问实例。

```
aws ec2 describe-verified-access-instances \
  --verified-access-instance-ids vai-0ce000c0b7643abea
```

输出：

```
{
  "VerifiedAccessInstances": [
    {
      "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",
      "Description": "Testing Verified Access",
      "VerifiedAccessTrustProviders": [
        {
          "VerifiedAccessTrustProviderId": "vatp-0bb32de759a3e19e7",
          "TrustProviderType": "user",
          "UserTrustProviderType": "iam-identity-center"
        }
      ],
      "CreationTime": "2023-08-25T18:27:56",
      "LastUpdatedTime": "2023-08-25T19:03:32",
      "Tags": [
        {
          "Key": "Name",
          "Value": "my-ava-instance"
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅 [《已验证访问用户指南》](#) 中的 [AWS 已验证访问实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeVerifiedAccessInstances](#) 中的。



## describe-verified-access-trust-providers

以下代码示例演示了如何使用 `describe-verified-access-trust-providers`。

### AWS CLI

描述已验证访问信任提供商

以下 `describe-verified-access-trust-providers` 示例描述了指定的已验证访问信任提供商。

```
aws ec2 describe-verified-access-trust-providers \
  --verified-access-trust-provider-ids vatp-0bb32de759a3e19e7
```

输出：

```
{
  "VerifiedAccessTrustProviders": [
    {
      "VerifiedAccessTrustProviderId": "vatp-0bb32de759a3e19e7",
      "Description": "Testing Verified Access",
      "TrustProviderType": "user",
      "UserTrustProviderType": "iam-identity-center",
      "PolicyReferenceName": "idc",
      "CreationTime": "2023-08-25T19:00:38",
      "LastUpdatedTime": "2023-08-25T19:03:32",
      "Tags": [
        {
          "Key": "Name",
          "Value": "my-va-trust-provider"
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅《已验证访问用户指南》中的“AWS 验证访问权限的[信任提供商](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeVerifiedAccessTrustProviders](#) 中的。

## describe-volume-attribute

以下代码示例演示了如何使用 `describe-volume-attribute`。

## AWS CLI

### 描述音量属性

此示例命令描述带有 ID 的卷的 `autoEnableIo` 属性 `vol-049df61146c4d7901`。

命令:

```
aws ec2 describe-volume-attribute --volume-id vol-049df61146c4d7901 --attribute
autoEnableIO
```

输出:

```
{
  "AutoEnableIO": {
    "Value": false
  },
  "VolumeId": "vol-049df61146c4d7901"
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeVolumeAttribute](#) 中的。

## describe-volume-status

以下代码示例演示了如何使用 `describe-volume-status`。

## AWS CLI

### 描述单个卷的状态

此示例命令描述了卷的状态 `vol-1234567890abcdef0`。

命令:

```
aws ec2 describe-volume-status --volume-ids vol-1234567890abcdef0
```

输出:

```
{
  "VolumeStatuses": [
    {
```

```
    "VolumeStatus": {
      "Status": "ok",
      "Details": [
        {
          "Status": "passed",
          "Name": "io-enabled"
        },
        {
          "Status": "not-applicable",
          "Name": "io-performance"
        }
      ]
    },
    "AvailabilityZone": "us-east-1a",
    "VolumeId": "vol-1234567890abcdef0",
    "Actions": [],
    "Events": []
  }
]
```

### 描述受损卷的状态

此示例命令描述了所有受损卷的状态。在此示例输出中，没有受损的音量。

命令:

```
aws ec2 describe-volume-status --filters Name=volume-status.status,Values=impaired
```

输出:

```
{
  "VolumeStatuses": []
}
```

如果您的卷状态检查失败（状态受损），请参阅 Amazon EC2 用户指南中的处理受损卷。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeVolumeStatus](#) 中的。

## describe-volumes-modifications

以下代码示例演示了如何使用 describe-volumes-modifications。

## AWS CLI

### 描述卷的修改状态

以下describe-volumes-modifications示例描述了指定卷的卷修改状态。

```
aws ec2 describe-volumes-modifications \  
  --volume-ids vol-1234567890abcdef0
```

输出：

```
{  
  "VolumeModification": {  
    "TargetSize": 150,  
    "TargetVolumeType": "io1",  
    "ModificationState": "optimizing",  
    "VolumeId": " vol-1234567890abcdef0",  
    "TargetIops": 100,  
    "StartTime": "2019-05-17T11:27:19.000Z",  
    "Progress": 70,  
    "OriginalVolumeType": "io1",  
    "OriginalIops": 100,  
    "OriginalSize": 100  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVolumesModifications](#)中的。

## describe-volumes

以下代码示例演示了如何使用 describe-volumes。

## AWS CLI

### 示例 1：描述卷

以下describe-volumes示例描述了当前区域中的指定卷。

```
aws ec2 describe-volumes \  
  --volume-ids vol-049df61146c4d7901 vol-1234567890abcdef0
```

输出：

```
{
  "Volumes": [
    {
      "AvailabilityZone": "us-east-1a",
      "Attachments": [
        {
          "AttachTime": "2013-12-18T22:35:00.000Z",
          "InstanceId": "i-1234567890abcdef0",
          "VolumeId": "vol-049df61146c4d7901",
          "State": "attached",
          "DeleteOnTermination": true,
          "Device": "/dev/sda1"
        }
      ],
      "Encrypted": true,
      "KmsKeyId": "arn:aws:kms:us-east-2a:123456789012:key/8c5b2c63-b9bc-45a3-a87a-5513eEXAMPLE",
      "VolumeType": "gp2",
      "VolumeId": "vol-049df61146c4d7901",
      "State": "in-use",
      "Iops": 100,
      "SnapshotId": "snap-1234567890abcdef0",
      "CreateTime": "2019-12-18T22:35:00.084Z",
      "Size": 8
    },
    {
      "AvailabilityZone": "us-east-1a",
      "Attachments": [],
      "Encrypted": false,
      "VolumeType": "gp2",
      "VolumeId": "vol-1234567890abcdef0",
      "State": "available",
      "Iops": 300,
      "SnapshotId": "",
      "CreateTime": "2020-02-27T00:02:41.791Z",
      "Size": 100
    }
  ]
}
```

## 示例 2：描述连接到特定实例的卷

以下describe-volumes示例描述了所有既连接到指定实例又设置为在实例终止时删除的卷。

```
aws ec2 describe-volumes \  
  --region us-east-1 \  
  --filters Name=attachment.instance-id,Values=i-1234567890abcdef0  
  Name=attachment.delete-on-termination,Values=true
```

有关 `describe-volumes` 的输出示例，请参阅示例 1。

### 示例 3：描述特定可用区中的可用卷

以下 `describe-volumes` 示例描述了状态为 `available` 且位于指定可用区的所有卷。

```
aws ec2 describe-volumes \  
  --filters Name=status,Values=available Name=availability-zone,Values=us-east-1a
```

有关 `describe-volumes` 的输出示例，请参阅示例 1。

### 示例 4：根据标签描述卷

以下 `describe-volumes` 示例描述了所有具有标签键 `Name` 和以开头的值的卷 `Test`。然后，使用 `仅` 显示卷标签和 ID 的查询筛选输出。

```
aws ec2 describe-volumes \  
  --filters Name=tag:Name,Values=Test* \  
  --query "Volumes[*].{ID:VolumeId,Tag:Tags}"
```

输出：

```
[  
  {  
    "Tag": [  
      {  
        "Value": "Test2",  
        "Key": "Name"  
      }  
    ],  
    "ID": "vol-1234567890abcdef0"  
  },  
  {  
    "Tag": [  
      {  
        "Value": "Test1",
```

```
        "Key": "Name"
      }
    ],
    "ID": "vol-049df61146c4d7901"
  }
]
```

有关使用标签筛选器的更多示例，请参阅《Amazon EC2 用户指南》中的[使用标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVolumes](#)中的。

## describe-vpc-attribute

以下代码示例演示了如何使用 describe-vpc-attribute。

### AWS CLI

描述该 enableDnsSupport 属性

此示例描述了该enableDnsSupport属性。此属性指示 VPC 是否已启用 DNS 解析。如果该属性为 true，则 Amazon DNS 服务器会将您实例的 DNS 主机名称解析为相应的 IP 地址，否则不会解析。

命令:

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --attribute enableDnsSupport
```

输出:

```
{
  "VpcId": "vpc-a01106c2",
  "EnableDnsSupport": {
    "Value": true
  }
}
```

描述该 enableDnsHostnames 属性

此示例描述了该enableDnsHostnames属性。此属性表示在 VPC 中启动的实例是否获得 DNS 主机名。如果该属性为 true，则 VPC 内的实例可获得 DNS 主机名称，否则将无法获得。

命令:

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --attribute enableDnsHostnames
```

输出:

```
{
  "VpcId": "vpc-a01106c2",
  "EnableDnsHostnames": {
    "Value": true
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVpcAttribute](#)中的。

## describe-vpc-classic-link-dns-support

以下代码示例演示了如何使用 describe-vpc-classic-link-dns-support。

AWS CLI

描述您的 VPC 的 ClassicLink DNS 支持

此示例描述了您的所有 VPC 的 ClassicLink DNS 支持状态。

命令:

```
aws ec2 describe-vpc-classic-link-dns-support
```

输出:

```
{
  "Vpcs": [
    {
      "VpcId": "vpc-88888888",
      "ClassicLinkDnsSupported": true
    },
    {
      "VpcId": "vpc-1a2b3c4d",
      "ClassicLinkDnsSupported": false
    }
  ]
}
```



```
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVpcClassicLinkDnsSupport](#)中的。

## describe-vpc-classic-link

以下代码示例演示了如何使用 describe-vpc-classic-link。

### AWS CLI

描述您的 VPC 的 ClassicLink 状态

此示例列出了 vpc- ClassicLink 88888888 的状态。

命令:

```
aws ec2 describe-vpc-classic-link --vpc-id vpc-88888888
```

输出 :

```
{  
  "Vpcs": [  
    {  
      "ClassicLinkEnabled": true,  
      "VpcId": "vpc-88888888",  
      "Tags": [  
        {  
          "Value": "classiclinkvpc",  
          "Key": "Name"  
        }  
      ]  
    }  
  ]  
}
```

此示例仅列出已启用 Classiclink 的 VPC ( 的筛选值设置is-classic-link-enabled为 )。true

命令:

```
aws ec2 describe-vpc-classic-link --filter "Name=is-classic-link-
enabled,Values=true"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVpcClassicLink](#)中的。

## describe-vpc-endpoint-connection-notifications

以下代码示例演示了如何使用 `describe-vpc-endpoint-connection-notifications`。

### AWS CLI

#### 描述端点连接通知

以下`describe-vpc-endpoint-connection-notifications`示例描述了您的所有终端节点连接通知。

```
aws ec2 describe-vpc-endpoint-connection-notifications
```

输出：

```
{
  "ConnectionNotificationSet": [
    {
      "ConnectionNotificationState": "Enabled",
      "ConnectionNotificationType": "Topic",
      "ConnectionEvents": [
        "Accept",
        "Reject",
        "Delete",
        "Connect"
      ],
      "ConnectionNotificationId": "vpce-nfn-04bcb952bc8af7abc",
      "ConnectionNotificationArn": "arn:aws:sns:us-
east-1:123456789012:VpceNotification",
      "VpcEndpointId": "vpce-0324151a02f327123"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVpcEndpointConnectionNotifications](#)中的。

## describe-vpc-endpoint-connections

以下代码示例演示了如何使用 `describe-vpc-endpoint-connections`。

### AWS CLI

#### 描述 VPC 终端节点连接

此示例描述了与您的终端节点服务的接口端点连接，并筛选结果以显示终端节点 `PendingAcceptance`。

#### 命令:

```
aws ec2 describe-vpc-endpoint-connections --filters Name=vpc-endpoint-  
state,Values=pendingAcceptance
```

#### 输出 :

```
{  
  "VpcEndpointConnections": [  
    {  
      "VpcEndpointId": "vpce-0abed31004e618123",  
      "ServiceId": "vpce-svc-0abced088d20def56",  
      "CreationTimestamp": "2017-11-30T10:00:24.350Z",  
      "VpcEndpointState": "pendingAcceptance",  
      "VpcEndpointOwner": "123456789012"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeVpcEndpointConnections](#) 中的。

## describe-vpc-endpoint-service-configurations

以下代码示例演示了如何使用 `describe-vpc-endpoint-service-configurations`。

### AWS CLI

#### 描述终端节点服务配置

以下 `describe-vpc-endpoint-service-configurations` 示例描述了您的终端节点服务配置。

```
aws ec2 describe-vpc-endpoint-service-configurations
```

输出：

```
{
  "ServiceConfigurations": [
    {
      "ServiceType": [
        {
          "ServiceType": "GatewayLoadBalancer"
        }
      ],
      "ServiceId": "vpce-svc-012d33a1c4321cab",
      "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-
svc-012d33a1c4321cab",
      "ServiceState": "Available",
      "AvailabilityZones": [
        "us-east-1d"
      ],
      "AcceptanceRequired": false,
      "ManagesVpcEndpoints": false,
      "GatewayLoadBalancerArns": [
        "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
gwy/GWLBService/123210844e429123"
      ],
      "Tags": []
    },
    {
      "ServiceType": [
        {
          "ServiceType": "Interface"
        }
      ],
      "ServiceId": "vpce-svc-123cab125efa123",
      "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-123cab125efa123",
      "ServiceState": "Available",
      "AvailabilityZones": [
        "us-east-1a"
      ],
      "AcceptanceRequired": true,
      "ManagesVpcEndpoints": false,
      "NetworkLoadBalancerArns": [
```

```

        "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
net/NLBforService/1238753950b25123"
    ],
    "BaseEndpointDnsNames": [
        "vpce-svc-123cab125efa123.us-east-1.vpce.amazonaws.com"
    ],
    "PrivateDnsName": "example.com",
    "PrivateDnsNameConfiguration": {
        "State": "failed",
        "Type": "TXT",
        "Value": "vpce:qUAth3FdeABCaPuiXabc",
        "Name": "_1d367jvbg34znqvyeFrj"
    },
    "Tags": []
}
]
}

```

有关更多信息，请参阅 Amazon VPC 用户指南中的 VPC [终端节点服务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeVpcEndpointServiceConfigurations](#) 中的。

## describe-vpc-endpoint-service-permissions

以下代码示例演示了如何使用 `describe-vpc-endpoint-service-permissions`。

### AWS CLI

#### 描述终端节点服务权限

此示例描述了指定终端节点服务的权限。

#### 命令:

```
aws ec2 describe-vpc-endpoint-service-permissions --service-id vpce-
svc-03d5ebb7d9579a2b3
```

#### 输出:

```
{
  "AllowedPrincipals": [
    {
```

```
        "PrincipalType": "Account",
        "Principal": "arn:aws:iam::123456789012:root"
    }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVpcEndpointServicePermissions](#)中的。

## describe-vpc-endpoint-services

以下代码示例演示了如何使用 describe-vpc-endpoint-services。

### AWS CLI

示例 1：描述所有 VPC 终端节点服务

以下 “describe-vpc-endpoint-services” 示例列出了某个 AWS 区域的所有 VPC 终端节点服务。

```
aws ec2 describe-vpc-endpoint-services
```

输出：

```
{
  "ServiceDetails": [
    {
      "ServiceType": [
        {
          "ServiceType": "Gateway"
        }
      ],
      "AcceptanceRequired": false,
      "ServiceName": "com.amazonaws.us-east-1.dynamodb",
      "VpcEndpointPolicySupported": true,
      "Owner": "amazon",
      "AvailabilityZones": [
        "us-east-1a",
        "us-east-1b",
        "us-east-1c",
        "us-east-1d",
        "us-east-1e",
        "us-east-1f"
      ]
    }
  ]
}
```

```
    ],
    "BaseEndpointDnsNames": [
        "dynamodb.us-east-1.amazonaws.com"
    ]
},
{
    "ServiceType": [
        {
            "ServiceType": "Interface"
        }
    ],
    "PrivateDnsName": "ec2.us-east-1.amazonaws.com",
    "ServiceName": "com.amazonaws.us-east-1.ec2",
    "VpcEndpointPolicySupported": false,
    "Owner": "amazon",
    "AvailabilityZones": [
        "us-east-1a",
        "us-east-1b",
        "us-east-1c",
        "us-east-1d",
        "us-east-1e",
        "us-east-1f"
    ],
    "AcceptanceRequired": false,
    "BaseEndpointDnsNames": [
        "ec2.us-east-1.vpce.amazonaws.com"
    ]
},
{
    "ServiceType": [
        {
            "ServiceType": "Interface"
        }
    ],
    "PrivateDnsName": "ssm.us-east-1.amazonaws.com",
    "ServiceName": "com.amazonaws.us-east-1.ssm",
    "VpcEndpointPolicySupported": true,
    "Owner": "amazon",
    "AvailabilityZones": [
        "us-east-1a",
        "us-east-1b",
        "us-east-1c",
        "us-east-1d",
        "us-east-1e"
    ]
}
```

```

    ],
    "AcceptanceRequired": false,
    "BaseEndpointDnsNames": [
        "ssm.us-east-1.vpce.amazonaws.com"
    ]
  }
],
"ServiceNames": [
  "com.amazonaws.us-east-1.dynamodb",
  "com.amazonaws.us-east-1.ec2",
  "com.amazonaws.us-east-1.ec2messages",
  "com.amazonaws.us-east-1.elasticloadbalancing",
  "com.amazonaws.us-east-1.kinesis-streams",
  "com.amazonaws.us-east-1.s3",
  "com.amazonaws.us-east-1.ssm"
]
}

```

有关更多信息，请参阅《用户指南》中的[查看可用 AWS 服务名称](#) AWS PrivateLink。

示例 2：描述终端节点服务的详细信息

以下 “describe-vpc-endpoint-services” 示例列出了 Amazon S3 接口终端节点服务的详细信息

```

aws ec2 describe-vpc-endpoint-services \
  --filter "Name=service-type,Values=Interface" Name=service-
name,Values=com.amazonaws.us-east-1.s3

```

输出：

```

{
  "ServiceDetails": [
    {
      "ServiceName": "com.amazonaws.us-east-1.s3",
      "ServiceId": "vpce-svc-081d84efcdEXAMPLE",
      "ServiceType": [
        {
          "ServiceType": "Interface"
        }
      ],
      "AvailabilityZones": [
        "us-east-1a",
        "us-east-1b",

```



```

        "us-east-1c",
        "us-east-1d",
        "us-east-1e",
        "us-east-1f"
    ],
    "Owner": "amazon",
    "BaseEndpointDnsNames": [
        "s3.us-east-1.vpce.amazonaws.com"
    ],
    "VpcEndpointPolicySupported": true,
    "AcceptanceRequired": false,
    "ManagesVpcEndpoints": false,
    "Tags": []
    }
],
"ServiceNames": [
    "com.amazonaws.us-east-1.s3"
]
}

```

有关更多信息，请参阅《用户指南》中的[查看可用 AWS 服务名称](#) AWS PrivateLink。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeVpcEndpointServices](#)中的。

## describe-vpc-endpoints

以下代码示例演示了如何使用 describe-vpc-endpoints。

### AWS CLI

描述您的 VPC 终端节点

以下 describe-vpc-endpoints 示例显示了您的所有 VPC 终端节点的详细信息。

```
aws ec2 describe-vpc-endpoints
```

输出：

```

{
  "VpcEndpoints": [
    {
      "PolicyDocument": "{\"Version\":\"2008-10-17\",\"Statement\":[{\"Effect\": \"Allow\", \"Principal\": \"*\", \"Action\": \"*\", \"Resource\": \"*\"}]}",

```

```

    "VpcId": "vpc-aabb1122",
    "NetworkInterfaceIds": [],
    "SubnetIds": [],
    "PrivateDnsEnabled": true,
    "State": "available",
    "ServiceName": "com.amazonaws.us-east-1.dynamodb",
    "RouteTableIds": [
        "rtb-3d560345"
    ],
    "Groups": [],
    "VpcEndpointId": "vpce-032a826a",
    "VpcEndpointType": "Gateway",
    "CreationTimestamp": "2017-09-05T20:41:28Z",
    "DnsEntries": [],
    "OwnerId": "123456789012"
},
{
    "PolicyDocument": "{\n  \"Statement\": [\n    {\n      \"Action\": \"*\n\", \n      \"Effect\": \"Allow\", \n      \"Principal\": \"*\", \n      \"Resource\n\": \"*\">\n    }\n  ]\n}",
    "VpcId": "vpc-1a2b3c4d",
    "NetworkInterfaceIds": [
        "eni-2ec2b084",
        "eni-1b4a65cf"
    ],
    "SubnetIds": [
        "subnet-d6fcaa8d",
        "subnet-7b16de0c"
    ],
    "PrivateDnsEnabled": false,
    "State": "available",
    "ServiceName": "com.amazonaws.us-east-1.elasticloadbalancing",
    "RouteTableIds": [],
    "Groups": [
        {
            "GroupName": "default",
            "GroupId": "sg-54e8bf31"
        }
    ],
    "VpcEndpointId": "vpce-0f89a33420c1931d7",
    "VpcEndpointType": "Interface",
    "CreationTimestamp": "2017-09-05T17:55:27.583Z",
    "DnsEntries": [
        {

```

```

        "HostedZoneId": "Z7HUB22UULQXV",
        "DnsName": "vpce-0f89a33420c1931d7-
bluzidnv.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
    },
    {
        "HostedZoneId": "Z7HUB22UULQXV",
        "DnsName": "vpce-0f89a33420c1931d7-bluzidnv-us-
east-1b.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
    },
    {
        "HostedZoneId": "Z7HUB22UULQXV",
        "DnsName": "vpce-0f89a33420c1931d7-bluzidnv-us-
east-1a.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
    }
],
"OwnerId": "123456789012"
},
{
    "VpcEndpointId": "vpce-aabbaabbaabbaabba",
    "VpcEndpointType": "GatewayLoadBalancer",
    "VpcId": "vpc-111122223333aabbc",
    "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-
svc-123123a1c43abc123",
    "State": "available",
    "SubnetIds": [
        "subnet-0011aabbcc2233445"
    ],
    "RequesterManaged": false,
    "NetworkInterfaceIds": [
        "eni-01010120203030405"
    ],
    "CreationTimestamp": "2020-11-11T08:06:03.522Z",
    "Tags": [],
    "OwnerId": "123456789012"
}
]
}

```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 [VPC 端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeVpcEndpoints](#) 中的。

## describe-vpc-peering-connections

以下代码示例演示了如何使用 describe-vpc-peering-connections。

### AWS CLI

描述您的 VPC 对等连接

此示例描述了您的所有 VPC 对等连接。

命令:

```
aws ec2 describe-vpc-peering-connections
```

输出:

```
{
  "VpcPeeringConnections": [
    {
      "Status": {
        "Message": "Active",
        "Code": "active"
      },
      "Tags": [
        {
          "Value": "Peering-1",
          "Key": "Name"
        }
      ],
      "AccepterVpcInfo": {
        "OwnerId": "111122223333",
        "VpcId": "vpc-1a2b3c4d",
        "CidrBlock": "10.0.1.0/28"
      },
      "VpcPeeringConnectionId": "pcx-11122233",
      "RequesterVpcInfo": {
        "PeeringOptions": {
          "AllowEgressFromLocalVpcToRemoteClassicLink": false,
          "AllowEgressFromLocalClassicLinkToRemoteVpc": false
        },
        "OwnerId": "444455556666",
        "VpcId": "vpc-123abc45",
        "CidrBlock": "192.168.0.0/16"
      }
    }
  ]
}
```

```

    }
  },
  {
    "Status": {
      "Message": "Pending Acceptance by 444455556666",
      "Code": "pending-acceptance"
    },
    "Tags": [],
    "RequesterVpcInfo": {
      "PeeringOptions": {
        "AllowEgressFromLocalVpcToRemoteClassicLink": false,
        "AllowEgressFromLocalClassicLinkToRemoteVpc": false
      },
      "OwnerId": "444455556666",
      "VpcId": "vpc-11aa22bb",
      "CidrBlock": "10.0.0.0/28"
    },
    "VpcPeeringConnectionId": "pcx-abababab",
    "ExpirationTime": "2014-04-03T09:12:43.000Z",
    "AccepterVpcInfo": {
      "OwnerId": "444455556666",
      "VpcId": "vpc-33cc44dd"
    }
  }
]
}

```

### 描述特定的 VPC 对等连接

此示例描述了所有处于“待接受”状态的 VPC 对等连接。

命令:

```
aws ec2 describe-vpc-peering-connections --filters Name=status-code,Values=pending-acceptance
```

此示例描述了所有标有 Owner=Finance 标签的 VPC 对等连接。

命令:

```
aws ec2 describe-vpc-peering-connections --filters Name=tag:Owner,Values=Finance
```

此示例描述了您为指定 VPC vpc-1a2b3c4d 请求的所有 VPC 对等连接。

**命令:**

```
aws ec2 describe-vpc-peering-connections --filters Name=requester-vpc-info.vpc-id,Values=vpc-1a2b3c4d
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVpcPeeringConnections](#)中的。

**describe-vpcs**

以下代码示例演示了如何使用 describe-vpcs。

**AWS CLI****示例 1：描述所有 VPC**

以下 describe-vpcs 示例检索 VPC 的详细信息。

```
aws ec2 describe-vpcs
```

**输出：**

```
{
  "Vpcs": [
    {
      "CidrBlock": "30.1.0.0/16",
      "DhcpOptionsId": "dopt-19edf471",
      "State": "available",
      "VpcId": "vpc-0e9801d129EXAMPLE",
      "OwnerId": "111122223333",
      "InstanceTenancy": "default",
      "CidrBlockAssociationSet": [
        {
          "AssociationId": "vpc-cidr-assoc-062c64cfafEXAMPLE",
          "CidrBlock": "30.1.0.0/16",
          "CidrBlockState": {
            "State": "associated"
          }
        }
      ]
    },
    "IsDefault": false,
    "Tags": [
      {
```

```

        "Key": "Name",
        "Value": "Not Shared"
      }
    ]
  },
  {
    "CidrBlock": "10.0.0.0/16",
    "DhcpOptionsId": "dopt-19edf471",
    "State": "available",
    "VpcId": "vpc-06e4ab6c6cEXAMPLE",
    "OwnerId": "222222222222",
    "InstanceTenancy": "default",
    "CidrBlockAssociationSet": [
      {
        "AssociationId": "vpc-cidr-assoc-00b17b4eddEXAMPLE",
        "CidrBlock": "10.0.0.0/16",
        "CidrBlockState": {
          "State": "associated"
        }
      }
    ],
    "IsDefault": false,
    "Tags": [
      {
        "Key": "Name",
        "Value": "Shared VPC"
      }
    ]
  }
]
}

```

## 示例 2：描述指定的 VPC

以下 `describe-vpcs` 示例检索指定 VPC 的详细信息。

```
aws ec2 describe-vpcs \
  --vpc-ids vpc-06e4ab6c6cEXAMPLE
```

输出：

```
{
  "Vpcs": [
```

```
{
  "CidrBlock": "10.0.0.0/16",
  "DhcpOptionsId": "dopt-19edf471",
  "State": "available",
  "VpcId": "vpc-06e4ab6c6cEXAMPLE",
  "OwnerId": "111122223333",
  "InstanceTenancy": "default",
  "CidrBlockAssociationSet": [
    {
      "AssociationId": "vpc-cidr-assoc-00b17b4eddEXAMPLE",
      "CidrBlock": "10.0.0.0/16",
      "CidrBlockState": {
        "State": "associated"
      }
    }
  ],
  "IsDefault": false,
  "Tags": [
    {
      "Key": "Name",
      "Value": "Shared VPC"
    }
  ]
}
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVpcs](#)中的。

## describe-vpn-connections

以下代码示例演示了如何使用 `describe-vpn-connections`。

### AWS CLI

#### 示例 1：描述您的 VPN 连接

以下 `describe-vpn-connections` 示例描述了您的所有站点到站点 VPN 连接。

```
aws ec2 describe-vpn-connections
```

输出：



```
{
  "VpnConnections": [
    {
      "CustomerGatewayConfiguration": "...configuration information...",
      "CustomerGatewayId": "cgw-01234567abcde1234",
      "Category": "VPN",
      "State": "available",
      "Type": "ipsec.1",
      "VpnConnectionId": "vpn-1122334455aabbccd",
      "TransitGatewayId": "tgw-00112233445566aab",
      "Options": {
        "EnableAcceleration": false,
        "StaticRoutesOnly": true,
        "LocalIpv4NetworkCidr": "0.0.0.0/0",
        "RemoteIpv4NetworkCidr": "0.0.0.0/0",
        "TunnelInsideIpVersion": "ipv4"
      },
      "Routes": [],
      "Tags": [
        {
          "Key": "Name",
          "Value": "CanadaVPN"
        }
      ],
      "VgwTelemetry": [
        {
          "AcceptedRouteCount": 0,
          "LastStatusChange": "2020-07-29T10:35:11.000Z",
          "OutsideIpAddress": "203.0.113.3",
          "Status": "DOWN",
          "StatusMessage": ""
        },
        {
          "AcceptedRouteCount": 0,
          "LastStatusChange": "2020-09-02T09:09:33.000Z",
          "OutsideIpAddress": "203.0.113.5",
          "Status": "UP",
          "StatusMessage": ""
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅站点到站点 VPN 用户[指南中的 AWS 站点到站点 VPN 的工作原理](#)。AWS

示例 2：描述您的可用的 VPN 连接

以下describe-vpn-connections示例描述了您的站点到站点 VPN 连接，其状态为。available

```
aws ec2 describe-vpn-connections \  
  --filters "Name=state,Values=available"
```

有关更多信息，请参阅站点到站点 VPN 用户[指南中的 AWS 站点到站点 VPN 的工作原理](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVpnConnections](#)中的。

## describe-vpn-gateways

以下代码示例演示了如何使用 describe-vpn-gateways。

### AWS CLI

描述您的虚拟专用网关

此示例描述您的虚拟专用网关。

命令：

```
aws ec2 describe-vpn-gateways
```

输出：

```
{  
  "VpnGateways": [  
    {  
      "State": "available",  
      "Type": "ipsec.1",  
      "VpnGatewayId": "vgw-f211f09b",  
      "VpcAttachments": [  
        {  
          "State": "attached",  
          "VpcId": "vpc-98eb5ef5"  
        }  
      ]  
    }  
  ],  
}
```

```
{
  "State": "available",
  "Type": "ipsec.1",
  "VpnGatewayId": "vgw-9a4cacf3",
  "VpcAttachments": [
    {
      "State": "attaching",
      "VpcId": "vpc-a01106c2"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVpnGateways](#)中的。

## detach-classic-link-vpc

以下代码示例演示了如何使用 detach-classic-link-vpc。

### AWS CLI

解除 EC2-Classical 实例与 VPC 的关联（分离）

此示例取消了实例 i-0598c7d356eba48d7 与 VPC vpc-88888888 的链接。

命令:

```
aws ec2 detach-classic-link-vpc --instance-id i-0598c7d356eba48d7 --vpc-id
vpc-88888888
```

输出：

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachClassicLinkVpc](#)中的。

## detach-internet-gateway

以下代码示例演示了如何使用 detach-internet-gateway。

## AWS CLI

将互联网网关与您的 VPC 分离

以下detach-internet-gateway示例将指定的互联网网关与特定 VPC 分离。

```
aws ec2 detach-internet-gateway \  
  --internet-gateway-id igw-0d0fb496b3EXAMPLE \  
  --vpc-id vpc-0a60eb65b4EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon VPC 用户指南》中的[互联网网关](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachInternetGateway](#)中的。

## detach-network-interface

以下代码示例演示了如何使用 detach-network-interface。

## AWS CLI

将网络接口与您的实例分离

此示例将指定的网络接口与指定实例分离。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 detach-network-interface --attachment-id eni-attach-66c4350a
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachNetworkInterface](#)中的。

## detach-verified-access-trust-provider

以下代码示例演示了如何使用 detach-verified-access-trust-provider。

## AWS CLI

将信任提供者与实例分离

以下detach-verified-access-trust-provider示例将指定的已验证访问信任提供者与指定的已验证访问实例分离。

```
aws ec2 detach-verified-access-trust-provider \  
  --verified-access-instance-id vai-0ce000c0b7643abea \  
  --verified-access-trust-provider-id vatp-0bb32de759a3e19e7
```

输出：

```
{  
  "VerifiedAccessTrustProvider": {  
    "VerifiedAccessTrustProviderId": "vatp-0bb32de759a3e19e7",  
    "Description": "Testing Verified Access",  
    "TrustProviderType": "user",  
    "UserTrustProviderType": "iam-identity-center",  
    "PolicyReferenceName": "idc",  
    "CreationTime": "2023-08-25T19:00:38",  
    "LastUpdatedTime": "2023-08-25T19:00:38"  
  },  
  "VerifiedAccessInstance": {  
    "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",  
    "Description": "Testing Verified Access",  
    "VerifiedAccessTrustProviders": [],  
    "CreationTime": "2023-08-25T18:27:56",  
    "LastUpdatedTime": "2023-08-25T18:27:56"  
  }  
}
```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachVerifiedAccessTrustProvider](#)中的。

## detach-volume

以下代码示例演示了如何使用 detach-volume。

### AWS CLI

将卷与实例分离

此示例命令将卷 (vol-049df61146c4d7901) 与其连接的实例分离。

命令：

```
aws ec2 detach-volume --volume-id vol-1234567890abcdef0
```

输出：

```
{
  "AttachTime": "2014-02-27T19:23:06.000Z",
  "InstanceId": "i-1234567890abcdef0",
  "VolumeId": "vol-049df61146c4d7901",
  "State": "detaching",
  "Device": "/dev/sdb"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachVolume](#)中的。

## detach-vpn-gateway

以下代码示例演示了如何使用 detach-vpn-gateway。

### AWS CLI

将虚拟私有网关与您的 VPC 分离

此示例将指定的虚拟私有网关与指定 VPC 分离。如果命令成功，则不返回任何输出。

命令：

```
aws ec2 detach-vpn-gateway --vpn-gateway-id vgw-9a4cacf3 --vpc-id vpc-a01106c2
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachVpnGateway](#)中的。

## disable-address-transfer

以下代码示例演示了如何使用 disable-address-transfer。

### AWS CLI

禁用弹性 IP 地址传输

以下disable-address-transfer示例禁用了指定弹性 IP 地址的弹性 IP 地址传输。

```
aws ec2 disable-address-transfer \
  --allocation-id eipalloc-09ad461b0d03f6aaf
```

输出：

```
{
  "AddressTransfer": {
    "PublicIp": "100.21.184.216",
    "AllocationId": "eipalloc-09ad461b0d03f6aaf",
    "AddressTransferStatus": "disabled"
  }
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的[传输弹性 IP 地址](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableAddressTransfer](#)中的。

## disable-aws-network-performance-metric-subscription

以下代码示例演示了如何使用 disable-aws-network-performance-metric-subscription。

### AWS CLI

#### 禁用指标订阅

以下disable-aws-network-performance-metric-subscription示例禁用了对指定源区域和目标区域之间聚合网络延迟的监控。

```
aws ec2 disable-aws-network-performance-metric-subscription \
  --source us-east-1 \
  --destination eu-west-1 \
  --metric aggregate-latency \
  --statistic p50
```

输出：

```
{
  "Output": true
}
```

有关更多信息，请参阅[《基础架构性能用户指南》](#)中的[管理订阅](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableAwsNetworkPerformanceMetricSubscription](#)中的。

## disable-efs-encryption-by-default

以下代码示例演示了如何使用 `disable-efs-encryption-by-default`。

### AWS CLI

#### 默认禁用 EFS 加密

以下 `disable-efs-encryption-by-default` 示例默认禁用当前区域中您的 AWS 账户的 EFS 加密。

```
aws efs disable-efs-encryption-by-default
```

输出：

```
{
  "EfsEncryptionByDefault": false
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisableEfsEncryptionByDefault](#) 中的。

## disable-fast-launch

以下代码示例演示了如何使用 `disable-fast-launch`。

### AWS CLI

#### 停止快速启动图像

以下 `disable-fast-launch` 示例停止在指定 AMI 上快速启动，并清理现有的预配置快照。

```
aws ec2 disable-fast-launch \
  --image-id ami-01234567890abcdef
```

输出：

```
{
  "ImageId": "ami-01234567890abcdef",
  "ResourceType": "snapshot",
  "SnapshotConfiguration": {},
  "LaunchTemplate": {
```



```

    "LaunchTemplateId": "lt-01234567890abcdef",
    "LaunchTemplateName": "EC2FastLaunchDefaultResourceCreation-
a8c6215d-94e6-441b-9272-dbd1f87b07e2",
    "Version": "1"
  },
  "MaxParallelLaunches": 6,
  "OwnerId": "0123456789123",
  "State": "disabling",
  "StateTransitionReason": "Client.UserInitiated",
  "StateTransitionTime": "2022-01-27T22:47:29.265000+00:00"
}

```

有关配置 Windows AMI 以加快启动速度的更多信息，请参阅 Amazon EC2 用户指南中的[配置 AMI 以加快启动速度](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DisableFastLaunch](#)中的。

## disable-fast-snapshot-restores

以下代码示例演示了如何使用 `disable-fast-snapshot-restores`。

### AWS CLI

#### 禁用快速快照恢复

以下 `disable-fast-snapshot-restores` 示例禁用指定可用区中指定快照的快速快照恢复。

```

aws ec2 disable-fast-snapshot-restores \
  --availability-zones us-east-2a \
  --source-snapshot-ids snap-1234567890abcdef0

```

输出：

```

{
  "Successful": [
    {
      "SnapshotId": "snap-1234567890abcdef0"
      "AvailabilityZone": "us-east-2a",
      "State": "disabling",
      "StateTransitionReason": "Client.UserInitiated",
      "OwnerId": "123456789012",
      "EnablingTime": "2020-01-25T23:57:49.602Z"
    }
  ]
}

```

```
    }
  ],
  "Unsuccessful": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableFastSnapshotRestores](#)中的。

## disable-image-block-public-access

以下代码示例演示了如何使用 `disable-image-block-public-access`。

### AWS CLI

在指定区域禁用 AMI 的公共访问屏蔽

以下`disable-image-block-public-access`示例在指定区域的账户级别禁用 AMI 的封锁公共访问权限。

```
aws ec2 disable-image-block-public-access \
  --region us-east-1
```

输出：

```
{
  "ImageBlockPublicAccessState": "unblocked"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[阻止公众访问您的 AMI](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableImageBlockPublicAccess](#)中的。

## disable-image-deprecation

以下代码示例演示了如何使用 `disable-image-deprecation`。

### AWS CLI

取消弃用 AMI

以下`disable-image-deprecation`示例取消了对 AMI 的弃用，这会将该`DeprecationTime`字段从输出中删除。`describe-images`您必须是 AMI 所有者才能执行此过程。

```
aws ec2 disable-image-deprecation \  
  --image-id ami-1234567890abcdef0
```

输出：

```
{  
  "RequestID": "11aabb229-4eac-35bd-99ed-be587EXAMPLE",  
  "Return": "true"  
}
```

有关更多信息，请参阅亚马逊 EC2 用户指南中的弃用 AMI < <https://docs.aws.amazon.com/AWS/ec2/latest/UserGuide/ami-deprecate.html#deprecate-ami>>。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableImageDeprecation](#)中的。

## disable-image

以下代码示例演示了如何使用 disable-image。

### AWS CLI

#### 禁用 AMI

以下disable-image示例禁用指定的 AMI。

```
aws ec2 disable-image \  
  --image-id ami-1234567890abcdef0
```

输出：

```
{  
  "Return": "true"  
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[禁用 AMI](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableImage](#)中的。

## disable-ipam-organization-admin-account

以下代码示例演示了如何使用 disable-ipam-organization-admin-account。

## AWS CLI

### 禁用委派的 IPAM 管理员

在某些情况下，您需要将 IPAM 与 Organizations 集成。AWS 当您这样做时，Organizations 管理账户会委托一个 AWS 组织成员账户作为 IPAM 管理员。

在此示例中，您是委托 IPAM 管理员帐户的 Organizations 管理帐户，并且您希望禁止该帐户成为 IPAM 管理员。AWS

在提出此请求 `--region` 时，您可以使用任何 AWS 区域。您不必使用最初委派管理员的区域、创建 IPAM 的区域或 IPAM 操作区域。如果您禁用了委托管理员帐户，则可以随时将其重新启用或委派新帐户作为 IPAM 管理员。

以下 `disable-ipam-organization-admin-account` 示例在您的帐户中禁用委托的 IPAM 管理员。AWS

```
aws ec2 disable-ipam-organization-admin-account \  
  --delegated-admin-account-id 320805250157 \  
  --region ap-south-1
```

输出：

```
{  
  "Success": true  
}
```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的将 IPAM [M 与 AWS 组织中的账户](#) 集成。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisableIpamOrganizationAdminAccount](#) 中的。

### **disable-serial-console-access**

以下代码示例演示了如何使用 `disable-serial-console-access`。

## AWS CLI

为您的帐户禁用对 EC2 串行控制台的访问权限

以下 `disable-serial-console-access` 示例禁用了对串行控制台的帐户访问权限。

```
aws ec2 disable-serial-console-access
```

输出：

```
{
  "SerialConsoleAccessEnabled": false
}
```

有关更多信息，请参阅 [Amazon EC2 用户指南中的 EC2 串行控制台](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableSerialConsoleAccess](#)中的。

## disable-transit-gateway-route-table-propagation

以下代码示例演示了如何使用 `disable-transit-gateway-route-table-propagation`。

### AWS CLI

禁用传输网关连接以将路由传播到指定的传播路由表

以下`disable-transit-gateway-route-table-propagation`示例禁止指定连接将路由传播到指定传播路由表。

```
aws ec2 disable-transit-gateway-route-table-propagation \
  --transit-gateway-route-table-id tgw-rtb-0a823edbdeEXAMPLE \
  --transit-gateway-attachment-id tgw-attach-09b52ccdb5EXAMPLE
```

输出：

```
{
  "Propagation": {
    "TransitGatewayAttachmentId": "tgw-attach-09b52ccdb5EXAMPLE",
    "ResourceId": "vpc-4d7de228",
    "ResourceType": "vpc",
    "TransitGatewayRouteTableId": "tgw-rtb-0a823edbdeEXAMPLE",
    "State": "disabled"
  }
}
```

有关更多信息，请参阅 [《公交网关指南》中的 Transit Gateway 路由表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableTransitGatewayRouteTablePropagation](#)中的。

## disable-vgw-route-propagation

以下代码示例演示了如何使用 `disable-vgw-route-propagation`。

### AWS CLI

#### 禁用路由传播

此示例禁止指定的虚拟专用网关向指定路由表传播静态路由。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 disable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-id
vgw-9a4cacf3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableVgwRoutePropagation](#)中的。

## disable-vpc-classic-link-dns-support

以下代码示例演示了如何使用 `disable-vpc-classic-link-dns-support`。

### AWS CLI

#### 禁用 VPC 的 ClassicLink DNS 支持

此示例禁用对 ClassicLink 的 DNS 支持。vpc-88888888

命令:

```
aws ec2 disable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

输出:

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableVpcClassicLinkDnsSupport](#)中的。

## disable-vpc-classic-link

以下代码示例演示了如何使用 `disable-vpc-classic-link`。

## AWS CLI

为 VPC 禁用 ClassicLink

此示例 ClassicLink 对于 vpc-88888888 禁用。

命令:

```
aws ec2 disable-vpc-classic-link --vpc-id vpc-88888888
```

输出:

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableVpcClassicLink](#)中的。

## disassociate-address

以下代码示例演示了如何使用 disassociate-address。

### AWS CLI

解除关联 EC2-Classical 中的弹性 IP 地址

以下示例将弹性 IP 地址与 EC2 Classic 中的实例解除关联。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 disassociate-address --public-ip 198.51.100.0
```

解除关联 EC2-VPC 中的弹性 IP 地址

以下示例将弹性 IP 地址与 VPC 中的实例解除关联。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 disassociate-address --association-id eipassoc-2bebb745
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateAddress](#)中的。

## disassociate-client-vpn-target-network

以下代码示例演示了如何使用 disassociate-client-vpn-target-network。

### AWS CLI

取消网络与 Client VPN 终端节点的关联

以下disassociate-client-vpn-target-network示例取消与指定 Client VPN 终端节点的cvpn-assoc-12312312312312312关联 ID 关联的目标网络。

```
aws ec2 disassociate-client-vpn-target-network \
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde \
  --association-id cvpn-assoc-12312312312312312
```

输出：

```
{
  "AssociationId": "cvpn-assoc-12312312312312312",
  "Status": {
    "Code": "disassociating"
  }
}
```

有关更多信息，请参阅《AWS Client VPN 管理员指南》中的“[目标网络](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateClientVpnTargetNetwork](#)中的。

## disassociate-iam-instance-profile

以下代码示例演示了如何使用 disassociate-iam-instance-profile。

### AWS CLI

取消关联 IAM 实例配置文件

此示例取消了 IAM 实例配置文件与关联 ID 的关联。iip-assoc-05020b59952902f5f

命令：



```
aws ec2 disassociate-iam-instance-profile --association-id iip-
assoc-05020b59952902f5f
```

输出：

```
{
  "IamInstanceProfileAssociation": {
    "InstanceId": "i-123456789abcde123",
    "State": "disassociating",
    "AssociationId": "iip-assoc-05020b59952902f5f",
    "IamInstanceProfile": {
      "Id": "AIPAI5IVIHMFFYY2DKV5Y",
      "Arn": "arn:aws:iam::123456789012:instance-profile/admin-role"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateIamInstanceProfile](#)中的。

## disassociate-instance-event-window

以下代码示例演示了如何使用 disassociate-instance-event-window。

### AWS CLI

示例 1：取消一个或多个实例与事件窗口的关联

以下disassociate-instance-event-window示例取消一个或多个实例与事件窗口的关联。指定instance-event-window-id参数以指定事件窗口。要取消关联实例，请指定association-target 参数，并为参数值指定一个或多个实例 ID。

```
aws ec2 disassociate-instance-event-window \
  --region us-east-1 \
  --instance-event-window-id iew-0abcdef1234567890 \
  --association-target "InstanceIds=i-1234567890abcdef0,i-0598c7d356eba48d7"
```

输出：

```
{
  "InstanceEventWindow": {
```

```

    "InstanceEventWindowId": "iew-0abcdef1234567890",
    "Name": "myEventWindowName",
    "CronExpression": "* 21-23 * * 2,3",
    "AssociationTarget": {
      "InstanceIds": [],
      "Tags": [],
      "DedicatedHostIds": []
    },
    "State": "creating"
  }
}

```

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中计划事件部分的[注意事项](#)。

### 示例 2：取消实例标签与事件窗口的关联

以下 `disassociate-instance-event-window` 示例取消实例标签与事件窗口的关联。指定 `instance-event-window-id` 参数以指定事件窗口。要取消关联实例标签，请指定 `association-target` 参数，并为参数值指定一个或多个标签。

```

aws ec2 disassociate-instance-event-window \
  --region us-east-1 \
  --instance-event-window-id iew-0abcdef1234567890 \
  --association-target "InstanceTags=[{Key=k2,Value=v2},{Key=k1,Value=v1}]"

```

输出：

```

{
  "InstanceEventWindow": {
    "InstanceEventWindowId": "iew-0abcdef1234567890",
    "Name": "myEventWindowName",
    "CronExpression": "* 21-23 * * 2,3",
    "AssociationTarget": {
      "InstanceIds": [],
      "Tags": [],
      "DedicatedHostIds": []
    },
    "State": "creating"
  }
}

```

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中计划事件部分的[注意事项](#)。

### 示例 3：取消专用主机与事件窗口的关联

以下 `disassociate-instance-event-window` 示例取消专用主机与事件窗口的关联。指定 `instance-event-window-id` 参数以指定事件窗口。要取消关联专属主机，请指定 `association-target` 参数，并为参数值指定一个或多个专属主机 ID。

```
aws ec2 disassociate-instance-event-window \
  --region us-east-1 \
  --instance-event-window-id iew-0abcdef1234567890 \
  --association-target DedicatedHostIds=h-029fa35a02b99801d
```

输出：

```
{
  "InstanceEventWindow": {
    "InstanceEventWindowId": "iew-0abcdef1234567890",
    "Name": "myEventWindowName",
    "CronExpression": "* 21-23 * * 2,3",
    "AssociationTarget": {
      "InstanceIds": [],
      "Tags": [],
      "DedicatedHostIds": []
    },
    "State": "creating"
  }
}
```

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中计划事件部分的[注意事项](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateInstanceEventWindow](#) 中的。

## disassociate-ipam-resource-discovery

以下代码示例演示了如何使用 `disassociate-ipam-resource-discovery`。

### AWS CLI

#### 取消资源发现与 IPAM 的关联

在此示例中，您是 IPAM 委托的管理员帐户，您想解除 IPAM 资源发现与 IPAM 的关联。您运行了 `describe` 命令 `"ResourceDiscoveryStatus": "not-found"` 并注意到您想取消它与 IPAM 的关联，以便为其他关联腾出空间。

以下`disassociate-ipam-resource-discovery`示例取消关联您账户中的 IPAM 资源发现。

## AWS

```
aws ec2 disassociate-ipam-resource-discovery \
  --ipam-resource-discovery-association-id ipam-res-disco-assoc-04382a6346357cf82 \
  --region us-east-1
```

输出：

```
{
  "IpamResourceDiscoveryAssociation": {
    "OwnerId": "320805250157",
    "IpamResourceDiscoveryAssociationId": "ipam-res-disco-
assoc-04382a6346357cf82",
    "IpamResourceDiscoveryAssociationArn":
"arn:aws:ec2::320805250157:ipam-resource-discovery-association/ipam-res-disco-
assoc-04382a6346357cf82",
    "IpamResourceDiscoveryId": "ipam-res-disco-0365d2977fc1672fe",
    "IpamId": "ipam-005f921c17ebd5107",
    "IpamArn": "arn:aws:ec2::320805250157:ipam/ipam-005f921c17ebd5107",
    "IpamRegion": "us-east-1",
    "IsDefault": false,
    "ResourceDiscoveryStatus": "not-found",
    "State": "disassociate-in-progress"
  }
}
```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的将 IPA [M 与组织以外的账户集成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateIpamResourceDiscovery](#) 中的。

## disassociate-nat-gateway-address

以下代码示例演示了如何使用 `disassociate-nat-gateway-address`。

### AWS CLI

取消弹性 IP 地址与公有 NAT 网关的关联

以下`disassociate-nat-gateway-address`示例取消指定弹性 IP 地址与指定公有 NAT 网关的关联。

```
aws ec2 disassociate-nat-gateway-address \  
  --nat-gateway-id nat-1234567890abcdef0 \  
  --association-ids eipassoc-0f96bdca17EXAMPLE
```

输出：

```
{  
  "NatGatewayId": "nat-1234567890abcdef0",  
  "NatGatewayAddresses": [  
    {  
      "AllocationId": "eipalloc-0be6ecac95EXAMPLE",  
      "NetworkInterfaceId": "eni-09cc4b2558794f7f9",  
      "PrivateIp": "10.0.0.74",  
      "PublicIp": "3.211.231.218",  
      "AssociationId": "eipassoc-0f96bdca17EXAMPLE",  
      "IsPrimary": false,  
      "Status": "disassociating"  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 [NAT 网关](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateNatGatewayAddress](#)中的。

## disassociate-route-table

以下代码示例演示了如何使用 disassociate-route-table。

### AWS CLI

#### 取消关联路由表

此示例取消指定路由表与指定子网的关联。如果命令成功，则不返回任何输出。

命令：

```
aws ec2 disassociate-route-table --association-id rtbassoc-781d0d1a
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateRouteTable](#)中的。

## disassociate-subnet-cidr-block

以下代码示例演示了如何使用 `disassociate-subnet-cidr-block`。

### AWS CLI

解除 IPv6 CIDR 块与子网的关联

此示例使用 IPv6 CIDR 块的关联 ID 将该 IPv6 CIDR 块与子网断开关联。

命令:

```
aws ec2 disassociate-subnet-cidr-block --association-id subnet-cidr-assoc-3aa54053
```

输出 :

```
{
  "SubnetId": "subnet-5f46ec3b",
  "Ipv6CidrBlockAssociation": {
    "Ipv6CidrBlock": "2001:db8:1234:1a00::/64",
    "AssociationId": "subnet-cidr-assoc-3aa54053",
    "Ipv6CidrBlockState": {
      "State": "disassociating"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateSubnetCidrBlock](#)中的。

## disassociate-transit-gateway-multicast-domain

以下代码示例演示了如何使用 `disassociate-transit-gateway-multicast-domain`。

### AWS CLI

取消子网与多播域的关联

以下`disassociate-transit-gateway-multicast-domain`示例取消子网与指定多播域的关联。

```
aws ec2 disassociate-transit-gateway-multicast-domain \
```

```
--transit-gateway-attachment-id tgw-attach-070e571cd1EXAMPLE \  
--subnet-id subnet-000de86e3bEXAMPLE \  
--transit-gateway-multicast-domain-id tgw-mcast-domain-0c4905cef7EXAMPLE
```

输出：

```
{  
  "Associations": [  
    {  
      "TransitGatewayMulticastDomainId": "tgw-mcast-domain-0c4905cef7EXAMPLE",  
      "TransitGatewayAttachmentId": "tgw-attach-070e571cd1EXAMPLE",  
      "ResourceId": "vpc-7EXAMPLE",  
      "ResourceType": "vpc",  
      "Subnets": [  
        {  
          "SubnetId": "subnet-000de86e3bEXAMPLE",  
          "State": "disassociating"  
        }  
      ]  
    }  
  ]  
}
```

有关更多信息，请参阅《传输网关指南》中的[“使用多播”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateTransitGatewayMulticastDomain](#)中的。

## disassociate-transit-gateway-route-table

以下代码示例演示了如何使用 disassociate-transit-gateway-route-table。

### AWS CLI

取消网关路由表与资源附件的关联

以下disassociate-transit-gateway-route-table示例取消指定附件与网关路由表的关联。

```
aws ec2 disassociate-transit-gateway-route-table \  
--transit-gateway-route-table-id tgw-rtb-002573ed1eEXAMPLE \  
--transit-gateway-attachment-id tgw-attach-08e0bc912cEXAMPLE
```

输出：

```
{
  "Association": {
    "TransitGatewayRouteTableId": "tgw-rtb-002573ed1eEXAMPLE",
    "TransitGatewayAttachmentId": "tgw-attach-08e0bc912cEXAMPLE",
    "ResourceId": "11460968-4ac1-4fd3-bdb2-00599EXAMPLE",
    "ResourceType": "direct-connect-gateway",
    "State": "disassociating"
  }
}
```

有关更多信息，请参阅 [《公网网关指南》中的 Transit Gateway 路由表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateTransitGatewayRouteTable](#) 中的。

## disassociate-vpc-cidr-block

以下代码示例演示了如何使用 disassociate-vpc-cidr-block。

### AWS CLI

解除 IPv6 CIDR 块与 VPC 的关联

此示例使用 IPv6 CIDR 块的关联 ID 将 IPv6 CIDR 块与 VPC 断开关联。

命令：

```
aws ec2 disassociate-vpc-cidr-block --association-id vpc-cidr-assoc-eca54085
```

输出：

```
{
  "Ipv6CidrBlockAssociation": {
    "Ipv6CidrBlock": "2001:db8:1234:1a00::/56",
    "AssociationId": "vpc-cidr-assoc-eca54085",
    "Ipv6CidrBlockState": {
      "State": "disassociating"
    }
  },
  "VpcId": "vpc-a034d6c4"
}
```

解除 IPv4 网段与 VPC 的关联



此示例将 IPv4 CIDR 块与 VPC 断开关联。

命令:

```
aws ec2 disassociate-vpc-cidr-block --association-id vpc-cidr-assoc-0287ac6b
```

输出:

```
{
  "CidrBlockAssociation": {
    "AssociationId": "vpc-cidr-assoc-0287ac6b",
    "CidrBlock": "172.18.0.0/16",
    "CidrBlockState": {
      "State": "disassociating"
    }
  },
  "VpcId": "vpc-27621243"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateVpcCidrBlock](#)中的。

## enable-address-transfer

以下代码示例演示了如何使用 enable-address-transfer。

AWS CLI

启用弹性 IP 地址传输

以下enable-address-transfer示例启用了将指定弹性 IP 地址的弹性 IP 地址传输到指定账户。

```
aws ec2 enable-address-transfer \
  --allocation-id eipalloc-09ad461b0d03f6aaf \
  --transfer-account-id 123456789012
```

输出:

```
{
  "AddressTransfer": {
    "PublicIp": "100.21.184.216",
  }
}
```

```
    "AllocationId": "eipalloc-09ad461b0d03f6aaf",
    "TransferAccountId": "123456789012",
    "TransferOfferExpirationTimestamp": "2023-02-22T20:51:01.000Z",
    "AddressTransferStatus": "pending"
  }
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的[传输弹性 IP 地址](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[EnableAddressTransfer](#)中的。

## enable-aws-network-performance-metric-subscription

以下代码示例演示了如何使用 enable-aws-network-performance-metric-subscription。

### AWS CLI

#### 启用指标订阅

以下 enable-aws-network-performance-metric-subscription 示例启用了对于指定源区域和目标区域之间的聚合网络延迟的监控。

```
aws ec2 enable-aws-network-performance-metric-subscription \
  --source us-east-1 \
  --destination eu-west-1 \
  --metric aggregate-latency \
  --statistic p50
```

输出：

```
{
  "Output": true
}
```

有关更多信息，请参阅[《基础架构性能用户指南》](#)中的[管理订阅](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[EnableAwsNetworkPerformanceMetricSubscription](#)中的。

## enable-efs-encryption-by-default

以下代码示例演示了如何使用 enable-efs-encryption-by-default。

## AWS CLI

### 默认启用 EBS 加密

以下`enable-efs-encryption-by-default`示例默认为您的当前区域中的 AWS 账户启用 EBS 加密。

```
aws ec2 enable-efs-encryption-by-default
```

输出：

```
{
  "EbsEncryptionByDefault": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableEbsEncryptionByDefault](#)中的。

### **enable-fast-launch**

以下代码示例演示了如何使用 `enable-fast-launch`。

## AWS CLI

### 开始快速启动图像

以下`enable-fast-launch`示例在指定的 AMI 上开始快速启动，并将要启动的最大并行实例数设置为 6。用于预置 AMI 的资源类型设置为 `snapshot`，它也是默认值。

```
aws ec2 enable-fast-launch \
  --image-id ami-01234567890abcdef \
  --max-parallel-launches 6 \
  --resource-type snapshot
```

输出：

```
{
  "ImageId": "ami-01234567890abcdef",
  "ResourceType": "snapshot",
  "SnapshotConfiguration": {
    "TargetResourceCount": 10
  },
}
```

```
"LaunchTemplate": {},
"MaxParallelLaunches": 6,
"OwnerId": "0123456789123",
"State": "enabling",
"StateTransitionReason": "Client.UserInitiated",
"StateTransitionTime": "2022-01-27T22:16:03.199000+00:00"
}
```

有关配置 Windows AMI 以加快启动速度的更多信息，请参阅 Amazon EC2 用户指南中的[配置 AMI 以加快启动速度](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[EnableFastLaunch](#)中的。

## enable-fast-snapshot-restores

以下代码示例演示了如何使用 enable-fast-snapshot-restores。

### AWS CLI

#### 启用快速快照恢复

以下 enable-fast-snapshot-restores 示例为指定可用区中的指定快照启用快速快照恢复。

```
aws ec2 enable-fast-snapshot-restores \
  --availability-zones us-east-2a us-east-2b \
  --source-snapshot-ids snap-1234567890abcdef0
```

输出：

```
{
  "Successful": [
    {
      "SnapshotId": "snap-1234567890abcdef0"
      "AvailabilityZone": "us-east-2a",
      "State": "enabling",
      "StateTransitionReason": "Client.UserInitiated",
      "OwnerId": "123456789012",
      "EnablingTime": "2020-01-25T23:57:49.602Z"
    },
    {
      "SnapshotId": "snap-1234567890abcdef0"
      "AvailabilityZone": "us-east-2b",
      "State": "enabling",
```

```
        "StateTransitionReason": "Client.UserInitiated",
        "OwnerId": "123456789012",
        "EnablingTime": "2020-01-25T23:57:49.596Z"
    }
  ],
  "Unsuccessful": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableFastSnapshotRestores](#)中的。

## enable-image-block-public-access

以下代码示例演示了如何使用 `enable-image-block-public-access`。

### AWS CLI

为指定区域的 AMI 启用封锁公共访问功能

以下`enable-image-block-public-access`示例为指定区域的账户级别的 AMI 启用了封锁公共访问权限。

```
aws ec2 enable-image-block-public-access \
  --region us-east-1 \
  --image-block-public-access-state block-new-sharing
```

输出：

```
{
  "ImageBlockPublicAccessState": "block-new-sharing"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[阻止公众访问您的 AMI](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableImageBlockPublicAccess](#)中的。

## enable-image-deprecation

以下代码示例演示了如何使用 `enable-image-deprecation`。

### AWS CLI

示例 1：弃用 AMI

以下enable-image-deprecation示例在特定日期和时间弃用了 AMI。如果指定以秒为单位的值，Amazon EC2 会将秒四舍五入到最近的分钟数。您必须是 AMI 所有者才能执行此过程。

```
aws ec2 enable-image-deprecation \  
  --image-id ami-1234567890abcdef0 \  
  --deprecate-at "2022-10-15T13:17:12.000Z"
```

输出：

```
{  
  "RequestID": "59dbff89-35bd-4eac-99ed-be587EXAMPLE",  
  "Return": "true"  
}
```

有关更多信息，请参阅亚马逊 EC2 用户指南中的弃用 AMI < <https://docs.aws.amazon.com/AWS/ec2/latest/UserGuide/ami-deprecate.html#deprecate-ami>>。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableImageDeprecation](#)中的。

## enable-image

以下代码示例演示了如何使用 enable-image。

### AWS CLI

启用 AMI

以下enable-image示例启用指定的 AMI。

```
aws ec2 enable-image \  
  --image-id ami-1234567890abcdef0
```

输出：

```
{  
  "Return": "true"  
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[禁用 AMI](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableImage](#)中的。

## enable-ipam-organization-admin-account

以下代码示例演示了如何使用 `enable-ipam-organization-admin-account`。

### AWS CLI

与 Organizations 集成并委托成员账户作为 IPAM 账户

以下 `enable-ipam-organization-admin-account` 示例将 IPAM 与 Organizations 集成，并委托一个成员账户作为 IPAM 账户。

```
aws ec2 enable-ipam-organization-admin-account \
  --delegated-admin-account-id 320805250157
```

输出：

```
{
  "Success": true
}
```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的 [将 IPAM 与组织集成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [EnableIpamOrganizationAdminAccount](#) 中的。

## enable-reachability-analyzer-organization-sharing

以下代码示例演示了如何使用 `enable-reachability-analyzer-organization-sharing`。

### AWS CLI

启用 Reachability Analyzer 的可信访问权限

以下 `enable-reachability-analyzer-organization-sharing` 示例为 Reachability Analyzer 启用可信访问。

```
aws ec2 enable-reachability-analyzer-organization-sharing
```

此命令不生成任何输出。

有关更多信息，请参阅 Reachability Analyzer 用户指南中的 [跨账户分析](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableReachabilityAnalyzerOrganizationSharing](#)中的。

## **enable-serial-console-access**

以下代码示例演示了如何使用 `enable-serial-console-access`。

### AWS CLI

为您的账户启用对串行控制台的访问权限

以下`enable-serial-console-access`示例启用了串行控制台的账户访问权限。

```
aws ec2 enable-serial-console-access
```

输出：

```
{
  "SerialConsoleAccessEnabled": true
}
```

有关更多信息，请参阅 [Amazon EC2 用户指南中的 EC2 串行控制台](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableSerialConsoleAccess](#)中的。

## **enable-transit-gateway-route-table-propagation**

以下代码示例演示了如何使用 `enable-transit-gateway-route-table-propagation`。

### AWS CLI

允许传输网关连接将路由传播到指定的传播路由表

以下`enable-transit-gateway-route-table-propagation`示例允许指定的连接将路由传播到指定的传播路由表。

```
aws ec2 enable-transit-gateway-route-table-propagation \
  --transit-gateway-route-table-id tgw-rtb-0a823eddbdeEXAMPLE \
  --transit-gateway-attachment-id tgw-attach-09b52ccdb5EXAMPLE
```

输出：



```
{
  "Propagation": {
    "TransitGatewayAttachmentId": "tgw-attach-09b52ccdb5EXAMPLE",
    "ResourceId": "vpc-4d7de228",
    "ResourceType": "vpc",
    "TransitGatewayRouteTableId": "tgw-rtb-0a823edbdeEXAMPLE",
    "State": "disabled"
  }
}
```

有关更多信息，请参阅《[公网网关指南](#)》中的 [Transit Gateway 路由表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableTransitGatewayRouteTablePropagation](#)中的。

## enable-vgw-route-propagation

以下代码示例演示了如何使用 enable-vgw-route-propagation。

### AWS CLI

#### 启用路由传播

此示例允许指定的虚拟专用网关将静态路由传播到指定的路由表。如果命令成功，则不返回任何输出。

#### 命令:

```
aws ec2 enable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-id
vgw-9a4cacf3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableVgwRoutePropagation](#)中的。

## enable-volume-io

以下代码示例演示了如何使用 enable-volume-io。

### AWS CLI

#### 为卷启用 I/O

此示例在卷上启用 I/O vol-1234567890abcdef0。

命令:

```
aws ec2 enable-volume-io --volume-id vol-1234567890abcdef0
```

输出:

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableVolumeIo](#)中的。

### **enable-vpc-classic-link-dns-support**

以下代码示例演示了如何使用 `enable-vpc-classic-link-dns-support`。

AWS CLI

为 VPC 启用 ClassicLink DNS 支持

此示例启用了 ClassicLink DNS 支持 `vpc-88888888`。

命令:

```
aws ec2 enable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

输出:

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableVpcClassicLinkDnsSupport](#)中的。

### **enable-vpc-classic-link**

以下代码示例演示了如何使用 `enable-vpc-classic-link`。

AWS CLI

为其启用 VPC ClassicLink

此示例为启用了 vpc-88888888。 ClassicLink

命令:

```
aws ec2 enable-vpc-classic-link --vpc-id vpc-88888888
```

输出:

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableVpcClassicLink](#)中的。

## export-client-vpn-client-certificate-revocation-list

以下代码示例演示了如何使用 `export-client-vpn-client-certificate-revocation-list`。

AWS CLI

导出客户证书吊销列表

以下`export-client-vpn-client-certificate-revocation-list`示例导出指定 Client VPN 端点的客户端证书吊销列表。在此示例中，为了便于阅读，输出以文本格式返回。

```
aws ec2 export-client-vpn-client-certificate-revocation-list \
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde \
  --output text
```

输出:

```
-----BEGIN X509 CRL-----
MIICiTCCAfICCD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBASTC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYW11ZAd
BgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI1MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBASTC0lBTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYW11ZAdBgkqhkiG9w0BCQEWEG5vb251QGft
```

```

YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcVQAaRhhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END X509 CRL-----
STATUS      pending

```

有关更多信息，请参阅 [《Client VPN 管理员指南》](#) 中的“AWS 客户证书吊销列表”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ExportClientVpnClientCertificateRevocationList](#) 中的。

## export-client-vpn-client-configuration

以下代码示例演示了如何使用 `export-client-vpn-client-configuration`。

### AWS CLI

#### 导出客户机配置

以下 `export-client-vpn-client-configuration` 示例导出指定 Client VPN 端点的客户端配置。在此示例中，为了便于阅读，输出以文本格式返回。

```

aws ec2 export-client-vpn-client-configuration \
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde \
  --output text

```

#### 输出：

```

client
dev tun
proto udp
remote cvpn-endpoint-123456789123abcde.prod.clientvpn.ap-south-1.amazonaws.com 443
remote-random-hostname
resolv-retry infinite
nobind
persist-key
persist-tun
remote-cert-tls server

```

```

cipher AES-256-GCM
verb 3
<ca>
-----BEGIN CERTIFICATE-----
MIICiTCCAfICCCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAlldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBA5TC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWx1eHAd
BgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAlldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC01BTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWx1eHAdBgkqhkiG9w0BCQEWEG5vb251QGft
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySwTc2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVvxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----
</ca>
reneg-sec 0

```

有关更多信息，请参阅 [《客户端 VPN 管理员指南》](#) 中的 [AWS Client VPN 端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ExportClientVpnClientConfiguration](#) 中的。

## export-image

以下代码示例演示了如何使用 `export-image`。

### AWS CLI

从 AMI 导出虚拟机

以下 `export-image` 示例以指定格式将指定的 AMI 导出到指定的存储桶。

```

aws ec2 export-image \
  --image-id ami-1234567890abcdef0 \
  --disk-image-format VMDK \
  --s3-export-location S3Bucket=my-export-bucket,S3Prefix=exports/

```

输出：

```
{
  "DiskImageFormat": "vmdk",
  "ExportImageTaskId": "export-ami-1234567890abcdef0"
  "ImageId": "ami-1234567890abcdef0",
  "RoleName": "vmimport",
  "Progress": "0",
  "S3ExportLocation": {
    "S3Bucket": "my-export-bucket",
    "S3Prefix": "exports/"
  },
  "Status": "active",
  "StatusMessage": "validating"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ExportImage](#)中的。

## get-associated-ipv6-pool-cidrs

以下代码示例演示了如何使用 `get-associated-ipv6-pool-cidrs`。

### AWS CLI

获取 IPv6 地址池的关联

以下`get-associated-ipv6-pool-cidrs`示例获取指定 IPv6 地址池的关联。

```
aws ec2 get-associated-ipv6-pool-cidrs \
  --pool-id ipv6pool-ec2-012345abc12345abc
```

输出：

```
{
  "Ipv6CidrAssociations": [
    {
      "Ipv6Cidr": "2001:db8:1234:1a00::/56",
      "AssociatedResource": "vpc-111111222222333ab"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅PoolCidrs 《AWS CLI 命令参考》中的 [GetAssociatedIpv6](#)。

## get-aws-network-performance-data

以下代码示例演示了如何使用 `get-aws-network-performance-data`。

### AWS CLI

#### 获取网络性能数据

以下 `get-aws-network-performance-data` 示例检索指定时间段内指定区域之间的网络性能数据。

```
aws ec2 get-aws-network-performance-data \
  --start-time 2022-10-26T12:00:00.000Z \
  --end-time 2022-10-26T12:30:00.000Z \
  --data-queries Id=my-query,Source=us-east-1,Destination=eu-
west-1,Metric=aggregate-latency,Statistic=p50,Period=five-minutes
```

输出：

```
{
  "DataResponses": [
    {
      "Id": "my-query",
      "Source": "us-east-1",
      "Destination": "eu-west-1",
      "Metric": "aggregate-latency",
      "Statistic": "p50",
      "Period": "five-minutes",
      "MetricPoints": [
        {
          "StartDate": "2022-10-26T12:00:00+00:00",
          "EndDate": "2022-10-26T12:05:00+00:00",
          "Value": 62.44349,
          "Status": "OK"
        },
        {
          "StartDate": "2022-10-26T12:05:00+00:00",
          "EndDate": "2022-10-26T12:10:00+00:00",
          "Value": 62.483498,
          "Status": "OK"
        },
        {
          "StartDate": "2022-10-26T12:10:00+00:00",
```

```
        "EndDate": "2022-10-26T12:15:00+00:00",
        "Value": 62.51248,
        "Status": "OK"
    },
    {
        "StartDate": "2022-10-26T12:15:00+00:00",
        "EndDate": "2022-10-26T12:20:00+00:00",
        "Value": 62.635475,
        "Status": "OK"
    },
    {
        "StartDate": "2022-10-26T12:20:00+00:00",
        "EndDate": "2022-10-26T12:25:00+00:00",
        "Value": 62.733974,
        "Status": "OK"
    },
    {
        "StartDate": "2022-10-26T12:25:00+00:00",
        "EndDate": "2022-10-26T12:30:00+00:00",
        "Value": 62.773975,
        "Status": "OK"
    },
    {
        "StartDate": "2022-10-26T12:30:00+00:00",
        "EndDate": "2022-10-26T12:35:00+00:00",
        "Value": 62.75349,
        "Status": "OK"
    }
  ]
}
```

有关更多信息，请参阅《基础架构[性能用户指南](#)》中的[监控网络性能](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAwsNetworkPerformanceData](#)中的。

## get-capacity-reservation-usage

以下代码示例演示了如何使用 get-capacity-reservation-usage。

### AWS CLI

查看各 AWS 账户的容量预留使用情况



以下`get-capacity-reservation-usage`示例显示了指定容量预留的使用信息。

```
aws ec2 get-capacity-reservation-usage \  
  --capacity-reservation-id cr-1234abcd56EXAMPLE
```

输出：

```
{  
  "CapacityReservationId": "cr-1234abcd56EXAMPLE ",  
  "InstanceUsages": [  
    {  
      "UsedInstanceCount": 1,  
      "AccountId": "123456789012"  
    }  
  ],  
  "AvailableInstanceCount": 4,  
  "TotalInstanceCount": 5,  
  "State": "active",  
  "InstanceType": "t2.medium"  
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[查看共享容量预留使用情况](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCapacityReservationUsage](#)中的。

## get-coip-pool-usage

以下代码示例演示了如何使用 `get-coip-pool-usage`。

### AWS CLI

获取客户自有的 IP 地址池使用情况

以下`get-coip-pool-usage`示例获取指定客户拥有的 IP 地址池的使用情况详细信息。

```
aws ec2 get-coip-pool-usage \  
  --pool-id ipv4pool-coip-123a45678bEXAMPLE
```

输出：

```
{
```

```
"CoipPoolId": "ipv4pool-coip-123a45678bEXAMPLE",
"CoipAddressUsages": [
  {
    "CoIp": "0.0.0.0"
  },
  {
    "AllocationId": "eipalloc-123ab45c6dEXAMPLE",
    "AwsAccountId": "123456789012",
    "CoIp": "0.0.0.0"
  },
  {
    "AllocationId": "eipalloc-123ab45c6dEXAMPLE",
    "AwsAccountId": "123456789111",
    "CoIp": "0.0.0.0"
  }
],
"LocalGatewayRouteTableId": "lgw-rtb-059615ef7dEXAMPLE"
}
```

有关更多信息，请参阅 [《AWS Outposts 用户指南》](#) 中的客户拥有的 IP 地址。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetCoipPoolUsage](#) 中的。

## get-console-output

以下代码示例演示了如何使用 get-console-output。

### AWS CLI

示例 1：获取控制台输出

以下 get-console-output 示例获取指定 Linux 实例的控制台输出。

```
aws ec2 get-console-output \
  --instance-id i-1234567890abcdef0
```

输出：

```
{
  "InstanceId": "i-1234567890abcdef0",
  "Timestamp": "2013-07-25T21:23:53.000Z",
  "Output": "..."
```

```
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[实例控制台输出](#)。

示例 2：获取最新的控制台输出

以下 `get-console-output` 示例获取指定 Linux 实例的最新控制台输出。

```
aws ec2 get-console-output \  
  --instance-id i-1234567890abcdef0 \  
  --latest \  
  --output text
```

输出：

```
i-1234567890abcdef0 [ 0.000000] Command line: root=LABEL=/ console=tty1  
console=ttyS0 selinux=0 nvme_core.io_timeout=4294967295  
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point  
registers'  
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'  
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'  
...  
Cloud-init v. 0.7.6 finished at Wed, 09 May 2018 19:01:13 +0000. Datasource  
DataSourceEc2. Up 21.50 seconds  
Amazon Linux AMI release 2018.03  
Kernel 4.14.26-46.32.amzn1.x
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[实例控制台输出](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetConsoleOutput](#) 中的。

## get-console-screenshot

以下代码示例演示了如何使用 `get-console-screenshot`。

### AWS CLI

检索正在运行的实例的屏幕截图

以下 `get-console-screenshot` 示例以 `.jpg` 格式检索指定实例的屏幕截图。屏幕截图以 Base64 编码的字符串形式返回。

```
aws ec2 get-console-screenshot \  
  --instance-id i-1234567890abcdef0
```

输出：

```
{  
  "ImageData": "997987/8kgj49ikjhewkwwe0008084EXAMPLE",  
  "InstanceId": "i-1234567890abcdef0"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetConsoleScreenshot](#)中的。

## get-default-credit-specification

以下代码示例演示了如何使用 `get-default-credit-specification`。

AWS CLI

描述默认积分选项

以下`get-default-credit-specification`示例描述了 T2 实例的默认积分选项。

```
aws ec2 get-default-credit-specification \  
  --instance-family t2
```

输出：

```
{  
  "InstanceFamilyCreditSpecification": {  
    "InstanceFamily": "t2",  
    "CpuCredits": "standard"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDefaultCreditSpecification](#)中的。

## get-efs-default-kms-key-id

以下代码示例演示了如何使用 `get-efs-default-kms-key-id`。

## AWS CLI

描述用于 EBS 加密的默认 CMK

以下`get-ebs-default-kms-key-id`示例描述了您的 AWS 账户的 EBS 加密的默认 CMK。

```
aws ec2 get-ebs-default-kms-key-id
```

输出显示了 EBS 加密的默认 CMK，即带有别名的 AWS 托管 CMK。 `alias/aws/ebs`

```
{
  "KmsKeyId": "alias/aws/ebs"
}
```

以下输出显示了用于 EBS 加密的自定义 CMK。

```
{
  "KmsKeyId": "arn:aws:kms:us-
west-2:123456789012:key/0ea3fef3-80a7-4778-9d8c-1c0c6EXAMPLE"
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetEbsDefaultKmsKeyId](#) 中的。

## `get-ebs-encryption-by-default`

以下代码示例演示了如何使用 `get-ebs-encryption-by-default`。

### AWS CLI

描述默认情况下是否启用 EBS 加密

以下`get-ebs-encryption-by-default`示例说明当前区域中您的 AWS 账户是否默认启用 EBS 加密。

```
aws ec2 get-ebs-encryption-by-default
```

以下输出表明 EBS 加密在默认情况下处于禁用状态。

```
{
```

```
"EbsEncryptionByDefault": false
}
```

以下输出表明默认情况下已启用 EBS 加密。

```
{
  "EbsEncryptionByDefault": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetEbsEncryptionByDefault](#)中的。

## get-flow-logs-integration-template

以下代码示例演示了如何使用 `get-flow-logs-integration-template`。

### AWS CLI

创建 CloudFormation 模板以自动将 VPC 流日志与 Amazon Athena 集成

以下`get-flow-logs-integration-template`示例创建了一个 CloudFormation 模板，用于自动将 VPC 流日志与 Amazon Athena 集成。

Linux :

```
aws ec2 get-flow-logs-integration-template \
  --flow-log-id fl-1234567890abcdef0 \
  --config-delivery-s3-destination-arn arn:aws:s3:::DOC-EXAMPLE-BUCKET \
  --integrate-services
  AthenaIntegrations='[{IntegrationResultS3DestinationArn=arn:aws:s3:::DOC-EXAMPLE-
BUCKET,PartitionLoadFrequency=none,PartitionStartDate=2021-07-21T00:40:00,PartitionEndDate=2
}{IntegrationResultS3DestinationArn=arn:aws:s3:::DOC-EXAMPLE-
BUCKET,PartitionLoadFrequency=none,PartitionStartDate=2021-07-21T00:40:00,PartitionEndDate=2
```

Windows:

```
aws ec2 get-flow-logs-integration-template ^
  --flow-log-id fl-1234567890abcdef0 ^
  --config-delivery-s3-destination-arn arn:aws:s3:::DOC-EXAMPLE-BUCKET ^
  --integrate-services
  AthenaIntegrations=[{IntegrationResultS3DestinationArn=arn:aws:s3:::DOC-EXAMPLE-
BUCKET,PartitionLoadFrequency=none,PartitionStartDate=2021-07-21T00:40:00,PartitionEndDate=2
```

```
{IntegrationResultS3DestinationArn=arn:aws:s3:::DOC-EXAMPLE-
BUCKET,PartitionLoadFrequency=none,PartitionStartDate=2021-07-21T00:40:00,PartitionEndDate=2
```

输出：

```
{
  "Result": "https://DOC-EXAMPLE-BUCKET.s3.us-east-2.amazonaws.com/
VPCFlowLogsIntegrationTemplate_f1-1234567890abcdef0_Wed%20Jul
%2021%2000%3A57%3A56%20UTC%202021.yml"
}
```

有关使用 CloudFormation 模板的信息，请参阅《AWS CloudFormation 用户指南》中的[使用 AWS CloudFormation 模板](#)。

有关使用 Amazon Athena 和流日志的信息，[请参阅亚马逊虚拟私有云用户指南中的使用亚马逊 Athena 查询流](#)日志。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetFlowLogsIntegrationTemplate](#)中的。

## get-groups-for-capacity-reservation

以下代码示例演示了如何使用 get-groups-for-capacity-reservation。

AWS CLI

列出具有容量预留的资源组

以下 get-groups-for-capacity-reservation 示例列出了向其添加了指定容量预留的资源组。

```
aws ec2 get-groups-for-capacity-reservation \
  --capacity-reservation-id cr-1234abcd56EXAMPLE
```

输出：

```
{
  "CapacityReservationsGroup": [
    {
      "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/my-
resource-group",
      "OwnerId": "123456789012"
    }
  ]
}
```

```
]
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[使用容量预留](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetGroupsForCapacityReservation](#)中的。

## get-host-reservation-purchase-preview

以下代码示例演示了如何使用 get-host-reservation-purchase-preview。

### AWS CLI

获取专用主机预留的购买预览

此示例预览了您账户中指定专用主机的指定专用主机预留费用。

命令:

```
aws ec2 get-host-reservation-purchase-preview --offering-id hro-03f707bf363b6b324 --
host-id-set h-013abcd2a00cbd123
```

输出:

```
{
  "TotalHourlyPrice": "1.499",
  "Purchase": [
    {
      "HourlyPrice": "1.499",
      "InstanceFamily": "m4",
      "PaymentOption": "NoUpfront",
      "HostIdSet": [
        "h-013abcd2a00cbd123"
      ],
      "UpfrontPrice": "0.000",
      "Duration": 31536000
    }
  ],
  "TotalUpfrontPrice": "0.000"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetHostReservationPurchasePreview](#)中的。



## get-image-block-public-access-state

以下代码示例演示了如何使用 `get-image-block-public-access-state`。

### AWS CLI

获取指定区域中 AMI 的封锁公共访问状态

以下 `get-image-block-public-access-state` 示例获取指定区域中账户级别 AMI 的封锁公共访问状态。

```
aws ec2 get-image-block-public-access-state \
  --region us-east-1
```

输出：

```
{
  "ImageBlockPublicAccessState": "block-new-sharing"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的 [阻止公众访问您的 AMI](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetImageBlockPublicAccessState](#) 中的。

## get-instance-types-from-instance-requirements

以下代码示例演示了如何使用 `get-instance-types-from-instance-requirements`。

### AWS CLI

预览与指定属性匹配的实例类型

以下 `get-instance-types-from-instance-requirements` 示例首先生成可使用 `--generate-cli-skeleton` 参数指定的所有可能属性的列表，并将列表保存到 JSON 文件中。然后，使用 JSON 文件自定义要预览匹配的实例类型的属性。

要生成所有可能的属性并将输出直接保存到 JSON 文件中，请使用以下命令。

```
aws ec2 get-instance-types-from-instance-requirements \
  --region us-east-1 \
  --generate-cli-skeleton input > attributes.json
```

输出：

```
{
  "DryRun": true,
  "ArchitectureTypes": [
    "x86_64_mac"
  ],
  "VirtualizationTypes": [
    "paravirtual"
  ],
  "InstanceRequirements": {
    "VCpuCount": {
      "Min": 0,
      "Max": 0
    },
    "MemoryMiB": {
      "Min": 0,
      "Max": 0
    },
    "CpuManufacturers": [
      "intel"
    ],
    "MemoryGiBPerVCpu": {
      "Min": 0.0,
      "Max": 0.0
    },
    "ExcludedInstanceTypes": [
      ""
    ],
    "InstanceGenerations": [
      "current"
    ],
    "SpotMaxPricePercentageOverLowestPrice": 0,
    "OnDemandMaxPricePercentageOverLowestPrice": 0,
    "BareMetal": "included",
    "BurstablePerformance": "excluded",
    "RequireHibernateSupport": true,
    "NetworkInterfaceCount": {
      "Min": 0,
      "Max": 0
    },
    "LocalStorage": "required",
    "LocalStorageTypes": [
      "hdd"
    ]
  }
}
```

```
    ],
    "TotalLocalStorageGB": {
      "Min": 0.0,
      "Max": 0.0
    },
    "BaselineEbsBandwidthMbps": {
      "Min": 0,
      "Max": 0
    },
    "AcceleratorTypes": [
      "inference"
    ],
    "AcceleratorCount": {
      "Min": 0,
      "Max": 0
    },
    "AcceleratorManufacturers": [
      "xilinx"
    ],
    "AcceleratorNames": [
      "t4"
    ],
    "AcceleratorTotalMemoryMiB": {
      "Min": 0,
      "Max": 0
    }
  },
  "MaxResults": 0,
  "NextToken": ""
}
```

配置 JSON 文件。您必须提供 `ArchitectureTypes`、`VirtualizationTypes`、`VCpuCount` 和 `MemoryMiB` 的值。您可以省略其他属性。省略时，将使用默认值。有关每个属性及其默认值的描述，请参阅 `get-instance-types-from-instance-requirements` < <https://docs.aws.amazon.com/cli/latest/reference/ec2/instance-requirements.html> >。 `get-instance-types-from`

预览具有中指定属性的实例类型 `attributes.json`。使用 `--cli-input-json` 参数指定 JSON 文件的名称和路径。在以下请求中，输出格式化为表格。

```
aws ec2 get-instance-types-from-instance-requirements \
  --cli-input-json file://attributes.json \
  --output table
```

`attributes.json` 文件的内容：

```
{
  "ArchitectureTypes": [
    "x86_64"
  ],
  "VirtualizationTypes": [
    "hvm"
  ],
  "InstanceRequirements": {
    "VCpuCount": {
      "Min": 4,
      "Max": 6
    },
    "MemoryMiB": {
      "Min": 2048
    },
    "InstanceGenerations": [
      "current"
    ]
  }
}
```

输出：

```
-----
|GetInstanceTypesFromInstanceRequirements|
+-----+
||           InstanceTypes           ||
|+-----+|
||           InstanceType           ||
|+-----+|
|| c4.xlarge                          ||
|| c5.xlarge                          ||
|| c5a.xlarge                         ||
|| c5ad.xlarge                       ||
|| c5d.xlarge                        ||
|| c5n.xlarge                        ||
|| d2.xlarge                         ||
...

```

有关基于属性的实例类型选择的更多信息，[请参阅 Amazon EC2 用户指南中的基于属性的实例类型选择的工作原理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInstanceTypesFromInstanceRequirements](#)中的。

## get-instance-uefi-data

以下代码示例演示了如何使用 get-instance-uefi-data。

### AWS CLI

从实例检索 UEFI 数据

以下get-instance-uefi-data示例从实例检索 UEFI 数据。如果输出为空，则实例不包含 UEFI 数据。

```
aws ec2 get-instance-uefi-data \  
  --instance-id i-0123456789example
```

输出：

```
{  
  "InstanceId": "i-0123456789example",  
  "UefiData": "QU1aTlVFRkkf+uLXAAAAAHj5a7fZ9+3dBzxXb/.  
  <snipped>  
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAD4L/J/A0Dshho="  
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的 [UEFI 安全启动](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInstanceUefiData](#)中的。

## get-ipam-address-history

以下代码示例演示了如何使用 get-ipam-address-history。

### AWS CLI

获取 CIDR 的历史记录

以下get-ipam-address-history示例获取 CIDR 的历史记录。

(Linux) :

```
aws ec2 get-ipam-address-history \  
  --cidr 10.0.0.0/16 \  
  --ipam-scope-id ipam-scope-02fc38cd4c48e7d38 \  
  --start-time 2021-12-08T01:00:00.000Z \  
  --end-time 2021-12-10T01:00:00.000Z
```

( 视窗 ) :

```
aws ec2 get-ipam-address-history ^  
  --cidr 10.0.0.0/16 ^  
  --ipam-scope-id ipam-scope-02fc38cd4c48e7d38 ^  
  --start-time 2021-12-08T01:00:00.000Z ^  
  --end-time 2021-12-10T01:00:00.000Z
```

输出 :

```
{  
  "HistoryRecords": [  
    {  
      "ResourceOwnerId": "123456789012",  
      "ResourceRegion": "us-west-1",  
      "ResourceType": "vpc",  
      "ResourceId": "vpc-06cbefa9ee907e1c0",  
      "ResourceCidr": "10.0.0.0/16",  
      "ResourceName": "Demo",  
      "ResourceComplianceStatus": "unmanaged",  
      "ResourceOverlapStatus": "overlapping",  
      "VpcId": "vpc-06cbefa9ee907e1c0",  
      "SampledStartTime": "2021-12-08T19:54:57.675000+00:00"  
    },  
    {  
      "ResourceOwnerId": "123456789012",  
      "ResourceRegion": "us-east-2",  
      "ResourceType": "vpc",  
      "ResourceId": "vpc-042702f474812c9ad",  
      "ResourceCidr": "10.0.0.0/16",  
      "ResourceName": "test",  
      "ResourceComplianceStatus": "unmanaged",  
      "ResourceOverlapStatus": "overlapping",  
      "VpcId": "vpc-042702f474812c9ad",
```

```

        "SampledStartTime": "2021-12-08T19:54:59.019000+00:00"
    },
    {
        "ResourceOwnerId": "123456789012",
        "ResourceRegion": "us-east-2",
        "ResourceType": "vpc",
        "ResourceId": "vpc-042b8a44f64267d67",
        "ResourceCidr": "10.0.0.0/16",
        "ResourceName": "tester",
        "ResourceComplianceStatus": "unmanaged",
        "ResourceOverlapStatus": "overlapping",
        "VpcId": "vpc-042b8a44f64267d67",
        "SampledStartTime": "2021-12-08T19:54:59.019000+00:00"
    }
]
}

```

有关更多信息，请参阅 Amazon VPC IPAM 用户 [指南中的查看 IP 地址历史记录](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetIpamAddressHistory](#) 中的。

## get-ipam-discovered-accounts

以下代码示例演示了如何使用 `get-ipam-discovered-accounts`。

### AWS CLI

#### 查看 IPAM 发现的账户

在这种情况下，您是 IPAM 授权的管理员，想要查看拥有 IPAM 发现的资源的 AWS 账户。

`--discovery-region` 是您要在其中查看受监控账户状态的 IPAM 操作区域。例如，如果您有三个 IPAM 操作区域，则可能需要发出三次此请求，以查看每个特定区域中针对发现的时间戳。

以下 `get-ipam-discovered-accounts` 示例列出了拥有 IPAM 正在发现的资源的 AWS 账户。

```

aws ec2 get-ipam-discovered-accounts \
  --ipam-resource-discovery-id ipam-res-disco-0365d2977fc1672fe \
  --discovery-region us-east-1

```

输出：

```
{
```

```
"IpamDiscoveredAccounts": [  
  {  
    "AccountId": "149977607591",  
    "DiscoveryRegion": "us-east-1",  
    "LastAttemptedDiscoveryTime": "2024-02-09T19:04:31.379000+00:00",  
    "LastSuccessfulDiscoveryTime": "2024-02-09T19:04:31.379000+00:00"  
  }  
]  
}
```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的将 IPAM [M 与组织以外的账户](#)集成。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetIpamDiscoveredAccounts](#)中的。

## get-ipam-discovered-public-addresses

以下代码示例演示了如何使用 get-ipam-discovered-public-addresses。

### AWS CLI

查看发现的公有 IP 地址

在此示例中，您是 IPAM 授权的管理员，想要查看 IPAM 发现的资源的 IP 地址。您可以通过获取资源发现 ID [describe-ipam-resource-discoveries](#)。

以下get-ipam-discovered-public-addresses示例显示了为资源发现而发现的公有 IP 地址。

```
aws ec2 get-ipam-discovered-public-addresses \  
  --ipam-resource-discovery-id ipam-res-disco-0f4ef577a9f37a162 \  
  --address-region us-east-1 \  
  --region us-east-1
```

输出：

```
{  
  "IpamDiscoveredPublicAddresses": [  
    {  
      "IpamResourceDiscoveryId": "ipam-res-disco-0f4ef577a9f37a162",  
      "AddressRegion": "us-east-1",  
      "Address": "54.208.155.7",  
      "AddressOwnerId": "320805250157",
```



```

    "AssociationStatus": "associated",
    "AddressType": "ec2-public-ip",
    "VpcId": "vpc-073b294916198ce49",
    "SubnetId": "subnet-0b6c8a8839e9a4f15",
    "NetworkInterfaceId": "eni-081c446b5284a5e06",
    "NetworkInterfaceDescription": "",
    "InstanceId": "i-07459a6fca5b35823",
    "Tags": {},
    "NetworkBorderGroup": "us-east-1c",
    "SecurityGroups": [
      {
        "GroupName": "launch-wizard-2",
        "GroupId": "sg-0a489dd6a65c244ce"
      }
    ],
    "SampleTime": "2024-04-05T15:13:59.228000+00:00"
  },
  {
    "IpamResourceDiscoveryId": "ipam-res-disco-0f4ef577a9f37a162",
    "AddressRegion": "us-east-1",
    "Address": "44.201.251.218",
    "AddressOwnerId": "470889052923",
    "AssociationStatus": "associated",
    "AddressType": "ec2-public-ip",
    "VpcId": "vpc-6c31a611",
    "SubnetId": "subnet-062f47608b99834b1",
    "NetworkInterfaceId": "eni-024845359c2c3ae9b",
    "NetworkInterfaceDescription": "",
    "InstanceId": "i-04ef786d9c4e03f41",
    "Tags": {},
    "NetworkBorderGroup": "us-east-1a",
    "SecurityGroups": [
      {
        "GroupName": "launch-wizard-32",
        "GroupId": "sg-0ed1a426e96a68374"
      }
    ],
    "SampleTime": "2024-04-05T15:13:59.145000+00:00"
  }
}

```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的[查看公有 IP 见解](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetIpamDiscoveredPublicAddresses](#)中的。

## get-ipam-discovered-resource-cidrs

以下代码示例演示了如何使用 `get-ipam-discovered-resource-cidrs`。

### AWS CLI

查看 IPAM 发现的 IP 地址 CIDR

在此示例中，您是 IPAM 授权的管理员，想要查看与 IPAM 正在发现的资源的 IP 地址 CIDR 相关的详细信息。

要完成此请求，请执行以下操作：

您选择的资源发现必须与 IPAM 关联。 `--resource-region` 这是创建资源的 AWS 区域。

以下 `get-ipam-discovered-resource-cidrs` 示例列出了 IPAM 正在发现的资源的 IP 地址。

```
aws ec2 get-ipam-discovered-resource-cidrs \  
  --ipam-resource-discovery-id ipam-res-disco-0365d2977fc1672fe \  
  --resource-region us-east-1
```

输出：

```
{  
  {  
    "IpamDiscoveredResourceCidrs": [  
      {  
        "IpamResourceDiscoveryId": "ipam-res-disco-0365d2977fc1672fe",  
        "ResourceRegion": "us-east-1",  
        "ResourceId": "vpc-0c974c95ca7ceef4a",  
        "ResourceOwnerId": "149977607591",  
        "ResourceCidr": "172.31.0.0/16",  
        "ResourceType": "vpc",  
        "ResourceTags": [],  
        "IpUsage": 0.375,  
        "VpcId": "vpc-0c974c95ca7ceef4a",  
        "SampleTime": "2024-02-09T19:15:16.529000+00:00"  
      },  
      {  
        "IpamResourceDiscoveryId": "ipam-res-disco-0365d2977fc1672fe",  
        "ResourceRegion": "us-east-1",  
        "ResourceId": "subnet-07fe028119082a8c1",
```

```

    "ResourceOwnerId": "149977607591",
    "ResourceCidr": "172.31.0.0/20",
    "ResourceType": "subnet",
    "ResourceTags": [],
    "IpUsage": 0.0012,
    "VpcId": "vpc-0c974c95ca7ceef4a",
    "SampleTime": "2024-02-09T19:15:16.529000+00:00"
  },
  {
    "IpamResourceDiscoveryId": "ipam-res-disco-0365d2977fc1672fe",
    "ResourceRegion": "us-east-1",
    "ResourceId": "subnet-0a96893763984cc4e",
    "ResourceOwnerId": "149977607591",
    "ResourceCidr": "172.31.64.0/20",
    "ResourceType": "subnet",
    "ResourceTags": [],
    "IpUsage": 0.0012,
    "VpcId": "vpc-0c974c95ca7ceef4a",
    "SampleTime": "2024-02-09T19:15:16.529000+00:00"
  }
}

```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的按资源监控 CIDR [使用情况](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetIpamDiscoveredResourceCidrs](#) 中的。

## get-ipam-pool-allocations

以下代码示例演示了如何使用 get-ipam-pool-allocations。

### AWS CLI

获取从 IPAM 池中分配的 CIDR

以下 get-ipam-pool-allocations 示例获取从 IPAM 池中分配的 CIDR。

(Linux) :

```

aws ec2 get-ipam-pool-allocations \
  --ipam-pool-id ipam-pool-0533048da7d823723 \
  --filters Name=ipam-pool-allocation-id,Values=ipam-pool-
alloc-0e6186d73999e47389266a5d6991e6220

```

( 视窗 ) :

```
aws ec2 get-ipam-pool-allocations ^
  --ipam-pool-id ipam-pool-0533048da7d823723 ^
  --filters Name=ipam-pool-allocation-id,Values=ipam-pool-
alloc-0e6186d73999e47389266a5d6991e6220
```

输出 :

```
{
  "IpamPoolAllocations": [
    {
      "Cidr": "10.0.0.0/16",
      "IpamPoolAllocationId": "ipam-pool-
alloc-0e6186d73999e47389266a5d6991e6220",
      "ResourceType": "custom",
      "ResourceOwner": "123456789012"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetIpamPoolAllocations](#)中的。

## get-ipam-pool-cidrs

以下代码示例演示了如何使用 `get-ipam-pool-cidrs`。

### AWS CLI

将 CIDR 配置到 IPAM 池

以下`get-ipam-pool-cidrs`示例获取了配置到 IPAM 池的 CIDR。

(Linux) :

```
aws ec2 get-ipam-pool-cidrs \
  --ipam-pool-id ipam-pool-0533048da7d823723 \
  --filters 'Name=cidr,Values=10.*'
```

( 视窗 ) :

```
aws ec2 get-ipam-pool-cidrs ^
```

```
--ipam-pool-id ipam-pool-0533048da7d823723 ^
--filters Name=cidr,Values=10.*
```

输出：

```
{
  "IpamPoolCidr": {
    "Cidr": "10.0.0.0/24",
    "State": "provisioned"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetIpamPoolCidrs](#)中的。

## get-ipam-resource-cidrs

以下代码示例演示了如何使用 `get-ipam-resource-cidrs`。

### AWS CLI

获取分配给资源的 CIDR

以下`get-ipam-resource-cidrs`示例获取分配给资源的 CIDR。

(Linux)：

```
aws ec2 get-ipam-resource-cidrs \
  --ipam-scope-id ipam-scope-02fc38cd4c48e7d38 \
  --filters Name=management-state,Values=unmanaged
```

( 视窗 )：

```
aws ec2 get-ipam-resource-cidrs ^
  --ipam-scope-id ipam-scope-02fc38cd4c48e7d38 ^
  --filters Name=management-state,Values=unmanaged
```

输出：

```
{
  "IpamResourceCidrs": [
    {
```

```

    "IpamId": "ipam-08440e7a3acde3908",
    "IpamScopeId": "ipam-scope-02fc38cd4c48e7d38",
    "ResourceRegion": "us-east-2",
    "ResourceOwnerId": "123456789012",
    "ResourceId": "vpc-621b8709",
    "ResourceName": "Default AWS VPC",
    "ResourceCidr": "172.33.0.0/16",
    "ResourceType": "vpc",
    "ResourceTags": [
      {
        "Key": "Environment",
        "Value": "Test"
      },
      {
        "Key": "Name",
        "Value": "Default AWS VPC"
      }
    ],
    "IpUsage": 0.0039,
    "ComplianceStatus": "unmanaged",
    "ManagementState": "unmanaged",
    "OverlapStatus": "nonoverlapping",
    "VpcId": "vpc-621b8709"
  }
]
}

```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的按资源监控 CID R [使用情况](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetIpamResourceCidrs](#) 中的。

## get-launch-template-data

以下代码示例演示了如何使用 `get-launch-template-data`。

### AWS CLI

#### 获取启动模板的实例数据

此示例获取有关指定实例的数据，并使用 `--query` 选项返回中的内容 `LaunchTemplateData`。您可以将输出作为基础以创建新的启动模板或启动模板版本。

命令:

```
aws ec2 get-launch-template-data --instance-id i-0123d646e8048bab0 --query
'LaunchTemplateData'
```

输出：

```
{
  "Monitoring": {},
  "ImageId": "ami-8c1be5f6",
  "BlockDeviceMappings": [
    {
      "DeviceName": "/dev/xvda",
      "Ebs": {
        "DeleteOnTermination": true
      }
    }
  ],
  "EbsOptimized": false,
  "Placement": {
    "Tenancy": "default",
    "GroupName": "",
    "AvailabilityZone": "us-east-1a"
  },
  "InstanceType": "t2.micro",
  "NetworkInterfaces": [
    {
      "Description": "",
      "NetworkInterfaceId": "eni-35306abc",
      "PrivateIpAddresses": [
        {
          "Primary": true,
          "PrivateIpAddress": "10.0.0.72"
        }
      ],
      "SubnetId": "subnet-7b16de0c",
      "Groups": [
        "sg-7c227019"
      ],
      "Ipv6Addresses": [
        {
          "Ipv6Address": "2001:db8:1234:1a00::123"
        }
      ],
      "PrivateIpAddress": "10.0.0.72"
    }
  ]
}
```

```
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLaunchTemplateData](#)中的。

## get-managed-prefix-list-associations

以下代码示例演示了如何使用 `get-managed-prefix-list-associations`。

### AWS CLI

获取前缀列表关联

以下`get-managed-prefix-list-associations`示例获取与指定前缀列表关联的资源。

```
aws ec2 get-managed-prefix-list-associations \  
  --prefix-list-id pl-0123456abcabc1
```

输出：

```
{  
  "PrefixListAssociations": [  
    {  
      "ResourceId": "sg-0abc123456abc12345",  
      "ResourceOwner": "123456789012"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的[托管前缀列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetManagedPrefixListAssociations](#)中的。

## get-managed-prefix-list-entries

以下代码示例演示了如何使用 `get-managed-prefix-list-entries`。

### AWS CLI

获取前缀列表的条目



以下`get-managed-prefix-list-entries`是指定前缀列表的条目。

```
aws ec2 get-managed-prefix-list-entries \  
  --prefix-list-id pl-0123456abcabc1
```

输出：

```
{  
  "Entries": [  
    {  
      "Cidr": "10.0.0.0/16",  
      "Description": "vpc-a"  
    },  
    {  
      "Cidr": "10.2.0.0/16",  
      "Description": "vpc-b"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的[托管前缀列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetManagedPrefixListEntries](#)中的。

## get-network-insights-access-scope-analysis-findings

以下代码示例演示了如何使用 `get-network-insights-access-scope-analysis-findings`。

AWS CLI

获取 Network Insights 访问范围分析的结果

以下`get-network-insights-access-scope-analysis-findings`示例获取您 AWS 账户中选定的范围分析结果。

```
aws ec2 get-network-insights-access-scope-analysis-findings \  
  --region us-east-1 \  
  --network-insights-access-scope-analysis-id nis \  
  --nis-123456789111
```

输出：

```
{
  "NetworkInsightsAccessScopeAnalysisId": "nisa-123456789222",
  "AnalysisFindings": [
    {
      "NetworkInsightsAccessScopeAnalysisId": "nisa-123456789222",
      "NetworkInsightsAccessScopeId": "nis-123456789111",
      "FindingComponents": [
        {
          "SequenceNumber": 1,
          "Component": {
            "Id": "eni-02e3d42d5cceca67d",
            "Arn": "arn:aws:ec2:us-east-1:936459623503:network-
interface/eni-02e3d32d9cceca17d"
          },
          "OutboundHeader": {
            "DestinationAddresses": [
              "0.0.0.0/5",
              "11.0.0.0/8",
              "12.0.0.0/6",
              "128.0.0.0/3",
              "16.0.0.0/4",
              "160.0.0.0/5",
              "168.0.0.0/6",
              "172.0.0.0/12"
              "8.0.0.0/7"
            ],
            "DestinationPortRanges": [
              {
                "From": 0,
                "To": 65535
              }
            ],
            "Protocol": "6",
            "SourceAddresses": [
              "10.0.2.253/32"
            ],
            "SourcePortRanges": [
              {
                "From": 0,
                "To": 65535
              }
            ]
          }, [etc]
        }
      ]
    }
  ]
}
```

```
    ]
  }
}
]
```

有关更多信息，请参阅《[网络访问分析器指南](#)》中的“[使用 AWS CLI 开始使用网络访问分析器](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetNetworkInsightsAccessScopeAnalysisFindings](#)中的。

## get-network-insights-access-scope-content

以下代码示例演示了如何使用 `get-network-insights-access-scope-content`。

### AWS CLI

要获取“网络见解”，请访问范围内容

以下`get-network-insights-access-scope-content`示例获取您 AWS 账户中选定范围分析 ID 的内容。

```
aws ec2 get-network-insights-access-scope-content \
  --region us-east-1 \
  --network-insights-access-scope-id nis-123456789222
```

输出：

```
{
  "NetworkInsightsAccessScopeContent": {
    "NetworkInsightsAccessScopeId": "nis-123456789222",
    "MatchPaths": [
      {
        "Source": {
          "ResourceStatement": {
            "ResourceTypes": [
              "AWS::EC2::NetworkInterface"
            ]
          }
        },
        "Destination": {
          "ResourceStatement": {
            "ResourceTypes": [
```

```
    "AWS::EC2::InternetGateway"
  ]
}
}
}
```

有关更多信息，请参阅 [《网络访问分析器指南》](#) 中的“使用 AWS CLI 开始使用网络访问分析器”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetNetworkInsightsAccessScopeContent](#) 中的。

## get-password-data

以下代码示例演示了如何使用 get-password-data。

### AWS CLI

要获取加密的密码

此示例获取加密的密码。

命令：

```
aws ec2 get-password-data --instance-id i-1234567890abcdef0
```

输出：

```
{
  "InstanceId": "i-1234567890abcdef0",
  "Timestamp": "2013-08-07T22:18:38.000Z",
  "PasswordData": "gSlJFq+VpcZXqy+iktXMF6NyxQ4qCrT4+gaOuN0enX1MmgXPTj7XEXAMPLE
UQ+YeFfb+L1U4C4AKv652Ux1iRB3CPTY7WmU3TUnhsuBd+p6LVk7T2lKUm160Xbk6WPW1VYYm/TRPB1
e1DQ7PY4an/DgZT4mwcpRFigzhniQgDDe01InvSDcwoUTwNs0Y1S8ouri2W4n5GNlriM3Q0AnNVe1Vz/
53TkDtxbNoU606M1gK9zUWSxqEgwvbV2j8c5rP0WCuaMWSF14ziDu4bd7q+4RSyi8NUsVWnKZ4aEZffu
DPGzKrF5yL1f3etP2L4ZR6CvG7K1hx7VK0QVN32Dajw=="
}
```

获取解密后的密码

此示例获取解密后的密码。

**命令:**

```
aws ec2 get-password-data --instance-id i-1234567890abcdef0 --priv-launch-key C:\Keys\MyKeyPair.pem
```

**输出:**

```
{
  "InstanceId": "i-1234567890abcdef0",
  "Timestamp": "2013-08-30T23:18:05.000Z",
  "PasswordData": "&ViJ652e*u"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPasswordData](#)中的。

**get-reserved-instances-exchange-quote**

以下代码示例演示了如何使用 `get-reserved-instances-exchange-quote`。

**AWS CLI**

获取交换可转换预留实例的报价

此示例获取指定可转换预留实例的交换信息。

**命令:**

```
aws ec2 get-reserved-instances-exchange-quote --reserved-instance-ids
7b8750c3-397e-4da4-bbcb-a45ebexample --target-configurations OfferingId=6fea5434-
b379-434c-b07b-a7abexample
```

**输出:**

```
{
  "CurrencyCode": "USD",
  "ReservedInstanceValueSet": [
    {
      "ReservedInstanceId": "7b8750c3-397e-4da4-bbcb-a45ebexample",
      "ReservationValue": {
        "RemainingUpfrontValue": "0.000000",
        "HourlyPrice": "0.027800",

```

```

        "RemainingTotalValue": "730.556200"
      }
    ],
    "PaymentDue": "424.983828",
    "TargetConfigurationValueSet": [
      {
        "TargetConfiguration": {
          "InstanceCount": 5,
          "OfferingId": "6fea5434-b379-434c-b07b-a7abexample"
        },
        "ReservationValue": {
          "RemainingUpfrontValue": "424.983828",
          "HourlyPrice": "0.016000",
          "RemainingTotalValue": "845.447828"
        }
      }
    ],
    "IsValidExchange": true,
    "OutputReservedInstancesWillExpireAt": "2020-10-01T13:03:39Z",
    "ReservedInstanceValueRollup": {
      "RemainingUpfrontValue": "0.000000",
      "HourlyPrice": "0.027800",
      "RemainingTotalValue": "730.556200"
    },
    "TargetConfigurationValueRollup": {
      "RemainingUpfrontValue": "424.983828",
      "HourlyPrice": "0.016000",
      "RemainingTotalValue": "845.447828"
    }
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetReservedInstancesExchangeQuote](#)中的。

## get-security-groups-for-vpc

以下代码示例演示了如何使用 `get-security-groups-for-vpc`。

### AWS CLI

查看可与指定 VPC 中的网络接口关联的安全组。

以下`get-security-groups-for-vpc`示例显示了可以与 VPC 中的网络接口关联的安全组。

```
aws ec2 get-security-groups-for-vpc \  
  --vpc-id vpc-6c31a611 \  
  --region us-east-1
```

输出：

```
{  
  "SecurityGroupForVpcs": [  
    {  
      "Description": "launch-wizard-36 created 2022-08-29T15:59:35.338Z",  
      "GroupName": "launch-wizard-36",  
      "OwnerId": "470889052923",  
      "GroupId": "sg-007e0c3027ee885f5",  
      "Tags": [],  
      "PrimaryVpcId": "vpc-6c31a611"  
    },  
    {  
      "Description": "launch-wizard-18 created 2024-01-19T20:22:27.527Z",  
      "GroupName": "launch-wizard-18",  
      "OwnerId": "470889052923",  
      "GroupId": "sg-0147193bef51c9eef",  
      "Tags": [],  
      "PrimaryVpcId": "vpc-6c31a611"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSecurityGroupsForVpc](#)中的。

## get-serial-console-access-status

以下代码示例演示了如何使用 `get-serial-console-access-status`。

### AWS CLI

查看账户访问串行控制台的状态

以下`get-serial-console-access-status`示例确定您的账户是否启用串行控制台访问权限。

```
aws ec2 get-serial-console-access-status
```

输出：

```
{
  "SerialConsoleAccessEnabled": true
}
```

有关更多信息，请参阅 [Amazon EC2 用户指南中的 EC2 串行控制台](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetSerialConsoleAccessStatus](#) 中的。

## get-spot-placement-scores

以下代码示例演示了如何使用 `get-spot-placement-scores`。

### AWS CLI

计算指定要求的 Spot 投放分数

以下 `get-spot-placement-scores` 示例首先生成可使用参数为竞价投放分数配置指定的所有可能 `--generate-cli-skeleton` 参数的列表，并将该列表保存到 JSON 文件中。然后，使用 JSON 文件配置用于计算竞价投放分数的要求。

生成可以为竞价投放分数配置指定的所有可能参数，并将输出直接保存到 JSON 文件中。

```
aws ec2 get-spot-placement-scores \
  --region us-east-1 \
  --generate-cli-skeleton input > attributes.json
```

输出：

```
{
  "InstanceTypes": [
    ""
  ],
  "TargetCapacity": 0,
  "TargetCapacityUnitType": "vcpu",
  "SingleAvailabilityZone": true,
  "RegionNames": [
    ""
  ],
  "InstanceRequirementsWithMetadata": {
    "ArchitectureTypes": [
      "x86_64_mac"
    ],

```



```
"VirtualizationTypes": [
  "hvm"
],
"InstanceRequirements": {
  "VCpuCount": {
    "Min": 0,
    "Max": 0
  },
  "MemoryMiB": {
    "Min": 0,
    "Max": 0
  },
  "CpuManufacturers": [
    "amd"
  ],
  "MemoryGiBPerVCpu": {
    "Min": 0.0,
    "Max": 0.0
  },
  "ExcludedInstanceTypes": [
    ""
  ],
  "InstanceGenerations": [
    "previous"
  ],
  "SpotMaxPricePercentageOverLowestPrice": 0,
  "OnDemandMaxPricePercentageOverLowestPrice": 0,
  "BareMetal": "excluded",
  "BurstablePerformance": "excluded",
  "RequireHibernateSupport": true,
  "NetworkInterfaceCount": {
    "Min": 0,
    "Max": 0
  },
  "LocalStorage": "included",
  "LocalStorageTypes": [
    "hdd"
  ],
  "TotalLocalStorageGB": {
    "Min": 0.0,
    "Max": 0.0
  },
  "BaselineEbsBandwidthMbps": {
    "Min": 0,
```

```

        "Max": 0
      },
      "AcceleratorTypes": [
        "fpga"
      ],
      "AcceleratorCount": {
        "Min": 0,
        "Max": 0
      },
      "AcceleratorManufacturers": [
        "amd"
      ],
      "AcceleratorNames": [
        "vu9p"
      ],
      "AcceleratorTotalMemoryMiB": {
        "Min": 0,
        "Max": 0
      }
    }
  },
  "DryRun": true,
  "MaxResults": 0,
  "NextToken": ""
}

```

配置 JSON 文件。您必须为 TargetCapacity 提供一个值。有关每个参数及其默认值的描述，请参阅计算竞价投放分数 (AWS CLI) < [https://docs.aws.amazon.com/AWS\\_ec2/latest/ /.html#UserGuide](https://docs.aws.amazon.com/AWS_ec2/latest/ /.html#UserGuide) >。spot-placement-score calculate-sps-cli

计算中指定要求的 Spot 投放分数attributes.json。使用--cli-input-json参数指定 JSON 文件的名称和路径。

```

aws ec2 get-spot-placement-scores \
  --region us-east-1 \
  --cli-input-json file://attributes.json

```

如果设置SingleAvailabilityZone为false或省略则输出（如果省略，则默认为false）。返回区域的分数列表。

```

"Recommendation": [
  {

```

```

    "Region": "us-east-1",
    "Score": 7
  },
  {
    "Region": "us-west-1",
    "Score": 5
  },
  ...

```

如果设置SingleAvailabilityZone为，则输出true。返回 SingleAvailability 区域的分数列表。

```

"Recommendation": [
  {
    "Region": "us-east-1",
    "AvailabilityZoneId": "use1-az1"
    "Score": 8
  },
  {
    "Region": "us-east-1",
    "AvailabilityZoneId": "usw2-az3"
    "Score": 6
  },
  ...

```

有关计算竞价投放分数的更多信息，以及配置示例，请参阅 Amazon EC2 用户指南中的[计算竞价投放分数](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSpotPlacementScores](#)中的。

## get-subnet-cidr-reservations

以下代码示例演示了如何使用 get-subnet-cidr-reservations。

### AWS CLI

获取有关子网 CIDR 预留的信息

以下get-subnet-cidr-reservations示例显示有关指定子网 CIDR 预留的信息。

```

aws ec2 get-subnet-cidr-reservations \
  --subnet-id subnet-03c51e2e6cEXAMPLE

```

输出：

```
{
  "SubnetIpv4CidrReservations": [
    {
      "SubnetCidrReservationId": "scr-044f977c4eEXAMPLE",
      "SubnetId": "subnet-03c51e2e6cEXAMPLE",
      "Cidr": "10.1.0.16/28",
      "ReservationType": "prefix",
      "OwnerId": "123456789012"
    }
  ],
  "SubnetIpv6CidrReservations": []
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的[子网 CIDR 预留](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSubnetCidrReservations](#)中的。

## get-transit-gateway-attachment-propagations

以下代码示例演示了如何使用 get-transit-gateway-attachment-propagations。

AWS CLI

列出指定资源附件将路由传播到的路由表

以下get-transit-gateway-attachment-propagations示例列出了指定资源附件将路由传播到的路由表。

```
aws ec2 get-transit-gateway-attachment-propagations \
  --transit-gateway-attachment-id tgw-attach-09fbd47ddfEXAMPLE
```

输出：

```
{
  "TransitGatewayAttachmentPropagations": [
    {
      "TransitGatewayRouteTableId": "tgw-rtb-0882c61b97EXAMPLE",
      "State": "enabled"
    }
  ]
}
```

```
}
```

有关更多信息，请参阅《[公网网关指南](#)》中的 [Transit Gateway 路由表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetTransitGatewayAttachmentPropagations](#) 中的。

## get-transit-gateway-multicast-domain-associations

以下代码示例演示了如何使用 `get-transit-gateway-multicast-domain-associations`。

### AWS CLI

查看有关传输网关组播域关联的信息

以下 `get-transit-gateway-multicast-domain-associations` 示例返回指定多播域的关联。

```
aws ec2 get-transit-gateway-multicast-domain-associations \  
--transit-gateway-multicast-domain-id tgw-mcast-domain-0c4905cef7EXAMPLE
```

输出：

```
{  
  "MulticastDomainAssociations": [  
    {  
      "TransitGatewayAttachmentId": "tgw-attach-028c1dd0f8EXAMPLE",  
      "ResourceId": "vpc-01128d2c24EXAMPLE",  
      "ResourceType": "vpc",  
      "Subnet": {  
        "SubnetId": "subnet-000de86e3bEXAMPLE",  
        "State": "associated"  
      }  
    },  
    {  
      "TransitGatewayAttachmentId": "tgw-attach-070e571cd1EXAMPLE",  
      "ResourceId": "vpc-7EXAMPLE",  
      "ResourceType": "vpc",  
      "Subnet": {  
        "SubnetId": "subnet-4EXAMPLE",  
        "State": "associated"  
      }  
    }  
  ]  
}
```

```
    },
    {
      "TransitGatewayAttachmentId": "tgw-attach-070e571cd1EXAMPLE",
      "ResourceId": "vpc-7EXAMPLE",
      "ResourceType": "vpc",
      "Subnet": {
        "SubnetId": "subnet-5EXAMPLE",
        "State": "associated"
      }
    },
    {
      "TransitGatewayAttachmentId": "tgw-attach-070e571cd1EXAMPLE",
      "ResourceId": "vpc-7EXAMPLE",
      "ResourceType": "vpc",
      "Subnet": {
        "SubnetId": "subnet-aEXAMPLE",
        "State": "associated"
      }
    },
    {
      "TransitGatewayAttachmentId": "tgw-attach-070e571cd1EXAMPLE",
      "ResourceId": "vpc-7EXAMPLE",
      "ResourceType": "vpc",
      "Subnet": {
        "SubnetId": "subnet-fEXAMPLE",
        "State": "associated"
      }
    }
  ]
}
```

有关更多信息，请参阅《传输网关指南》中的[管理多播域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTransitGatewayMulticastDomainAssociations](#)中的。

## get-transit-gateway-prefix-list-references

以下代码示例演示了如何使用 `get-transit-gateway-prefix-list-references`。

### AWS CLI

在公交网关路由表中获取前缀列表引用

以下`get-transit-gateway-prefix-list-references`示例获取指定公网网关路由表的前缀列表引用，并按特定前缀列表的 ID 进行过滤。

```
aws ec2 get-transit-gateway-prefix-list-references \
  --transit-gateway-route-table-id tgw-rtb-0123456789abcd123 \
  --filters Name=prefix-list-id,Values=pl-11111122222222333
```

输出：

```
{
  "TransitGatewayPrefixListReferences": [
    {
      "TransitGatewayRouteTableId": "tgw-rtb-0123456789abcd123",
      "PrefixListId": "pl-11111122222222333",
      "PrefixListOwnerId": "123456789012",
      "State": "available",
      "Blackhole": false,
      "TransitGatewayAttachment": {
        "TransitGatewayAttachmentId": "tgw-attach-aabbccddaabbccaab",
        "ResourceType": "vpc",
        "ResourceId": "vpc-112233445566aabbcc"
      }
    }
  ]
}
```

有关更多信息，请参阅 [Transit Gateways 指南中的前缀列表参考](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 GetTransitGatewayPrefixListReferences](#) 中的。

## get-transit-gateway-route-table-associations

以下代码示例演示了如何使用 `get-transit-gateway-route-table-associations`。

### AWS CLI

获取有关指定公网网关路由表关联的信息

以下`get-transit-gateway-route-table-associations`示例显示有关指定公网网关路由表的关联的信息。

```
aws ec2 get-transit-gateway-route-table-associations \
```

```
--transit-gateway-route-table-id tgw-rtb-0a823edbdeEXAMPLE
```

输出：

```
{
  "Associations": [
    {
      "TransitGatewayAttachmentId": "tgw-attach-09b52ccdb5EXAMPLE",
      "ResourceId": "vpc-4d7de228",
      "ResourceType": "vpc",
      "State": "associating"
    }
  ]
}
```

有关更多信息，请参阅《[公交网关指南](#)》中的 [Transit Gateway 路由表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTransitGatewayRouteTableAssociations](#)中的。

## get-transit-gateway-route-table-propagations

以下代码示例演示了如何使用 `get-transit-gateway-route-table-propagations`。

AWS CLI

显示有关指定公交网关路由表的路由表传播的信息

以下`get-transit-gateway-route-table-propagations`示例返回指定路由表的路由表传播。

```
aws ec2 get-transit-gateway-route-table-propagations \
  --transit-gateway-route-table-id tgw-rtb-002573ed1eEXAMPLE
```

输出：

```
{
  "TransitGatewayRouteTablePropagations": [
    {
      "TransitGatewayAttachmentId": "tgw-attach-01f8100bc7EXAMPLE",
      "ResourceId": "vpc-3EXAMPLE",
      "ResourceType": "vpc",

```



```

        "State": "enabled"
    },
    {
        "TransitGatewayAttachmentId": "tgw-attach-08e0bc912cEXAMPLE",
        "ResourceId": "11460968-4ac1-4fd3-bdb2-00599EXAMPLE",
        "ResourceType": "direct-connect-gateway",
        "State": "enabled"
    },
    {
        "TransitGatewayAttachmentId": "tgw-attach-0a89069f57EXAMPLE",
        "ResourceId": "8384da05-13ce-4a91-aada-5a1baEXAMPLE",
        "ResourceType": "direct-connect-gateway",
        "State": "enabled"
    }
]
}

```

有关更多信息，请参阅《[公网网关指南](#)》中的 [Transit Gateway 路由表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetTransitGatewayRouteTablePropagations](#) 中的。

## get-verified-access-endpoint-policy

以下代码示例演示了如何使用 `get-verified-access-endpoint-policy`。

### AWS CLI

获取终端节点的已验证访问策略

以下 `get-verified-access-endpoint-policy` 示例获取指定端点的已验证访问策略。

```
aws ec2 get-verified-access-endpoint-policy \
    --verified-access-endpoint-id vae-066fac616d4d546f2
```

输出：

```
{
  "PolicyEnabled": true,
  "PolicyDocument": "permit(principal,action,resource)\nwhen
{\n  context.identity.groups.contains(\"finance\") &&\n
context.identity.email_verified == true\n};"
```

```
}
```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVerifiedAccessEndpointPolicy](#)中的。

## get-verified-access-group-policy

以下代码示例演示了如何使用 `get-verified-access-group-policy`。

### AWS CLI

获取群组的“已验证访问权限”策略

以下`get-verified-access-group-policy`示例获取指定组的“已验证访问权限”策略。

```
aws ec2 get-verified-access-group-policy \  
  --verified-access-group-id vagr-0dbe967baf14b7235
```

输出：

```
{  
  "PolicyEnabled": true,  
  "PolicyDocument": "permit(principal,action,resource)\nwhen  
{\n  context.identity.groups.contains(\"finance\") &&\n  context.identity.email_verified == true\n};"  
}
```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVerifiedAccessGroupPolicy](#)中的。

## get-vpn-connection-device-sample-configuration

以下代码示例演示了如何使用 `get-vpn-connection-device-sample-configuration`。

### AWS CLI

下载示例配置文件

以下`get-vpn-connection-device-sample-configuration`示例下载指定的示例配置文件。要使用示例配置文件列出网关设备，请调用`get-vpn-connection-device-types`命令。

```
aws ec2 get-vpn-connection-device-sample-configuration \  
  --vpn-connection-id vpn-123456789abc01234 \  
  --vpn-connection-device-type-id 5fb390ba
```

输出：

```
{  
  "VpnConnectionDeviceSampleConfiguration": "contents-of-the-sample-configuration-  
file"  
}
```

有关更多信息，请参阅 [《AWS 站点到站点 VPN 用户指南》](#) 中的 [下载配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetVpnConnectionDeviceSampleConfiguration](#) 中的。

## get-vpn-connection-device-types

以下代码示例演示了如何使用 `get-vpn-connection-device-types`。

### AWS CLI

使用示例配置文件列出网关设备

以下 `get-vpn-connection-device-types` 示例列出了来自 Palo Alto Networks 的具有示例配置文件的网关设备。

```
aws ec2 get-vpn-connection-device-types \  
  --query "VpnConnectionDeviceTypes[?Vendor==`Palo Alto Networks`]"
```

输出：

```
[  
  {  
    "VpnConnectionDeviceTypeId": "754a6372",  
    "Vendor": "Palo Alto Networks",  
    "Platform": "PA Series",  
    "Software": "PANOS 4.1.2+"  
  },  
  {  
    "VpnConnectionDeviceTypeId": "9612cbed",
```

```

    "Vendor": "Palo Alto Networks",
    "Platform": "PA Series",
    "Software": "PANOS 4.1.2+ (GUI)"
  },
  {
    "VpnConnectionDeviceTypeId": "5fb390ba",
    "Vendor": "Palo Alto Networks",
    "Platform": "PA Series",
    "Software": "PANOS 7.0+"
  }
]

```

有关更多信息，请参阅 [《AWS 站点到站点 VPN 用户指南》](#) 中的 [下载配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetVpnConnectionDeviceTypes](#) 中的。

## import-client-vpn-client-certificate-revocation-list

以下代码示例演示了如何使用 `import-client-vpn-client-certificate-revocation-list`。

### AWS CLI

#### 导入客户证书吊销列表

以下 `import-client-vpn-client-certificate-revocation-list` 示例通过指定文件在本地计算机上的位置将客户端证书吊销列表导入到 Client VPN 端点。

```

aws ec2 import-client-vpn-client-certificate-revocation-list \
  --certificate-revocation-list file:///path/to/crl.pem \
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde

```

输出：

```

{
  "Return": true
}

```

有关更多信息，请参阅 [《Client VPN 管理员指南》](#) 中的 [“AWS 客户证书吊销列表”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ImportClientVpnClientCertificateRevocationList](#) 中的。

## import-image

以下代码示例演示了如何使用 `import-image`。

### AWS CLI

将 VM 映像文件作为 AMI 导入

以下 `import-image` 示例导入指定的 OVA。

```
aws ec2 import-image \  
  --disk-containers Format=ova,UserBucket="{S3Bucket=my-import-bucket,S3Key=vms/my-  
server-vm.ova}"
```

输出：

```
{  
  "ImportTaskId": "import-ami-1234567890abcdef0",  
  "Progress": "2",  
  "SnapshotDetails": [  
    {  
      "DiskImageSize": 0.0,  
      "Format": "ova",  
      "UserBucket": {  
        "S3Bucket": "my-import-bucket",  
        "S3Key": "vms/my-server-vm.ova"  
      }  
    }  
  ],  
  "Status": "active",  
  "StatusMessage": "pending"  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ImportImage](#) 中的。

## import-key-pair

以下代码示例演示了如何使用 `import-key-pair`。

### AWS CLI

导入公钥

首先，使用您选择的工具生成 key pair。例如，使用以下 ssh-keygen 命令：

命令：

```
ssh-keygen -t rsa -C "my-key" -f ~/.ssh/my-key
```

输出：

```
Generating public/private rsa key pair.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/ec2-user/.ssh/my-key.  
Your public key has been saved in /home/ec2-user/.ssh/my-key.pub.  
...
```

此示例命令导入指定的公钥。

命令：

```
aws ec2 import-key-pair --key-name "my-key" --public-key-material fileb://~/.ssh/my-key.pub
```

输出：

```
{  
  "KeyName": "my-key",  
  "KeyFingerprint": "1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ImportKeyPair](#)中的。

## import-snapshot

以下代码示例演示了如何使用 import-snapshot。

### AWS CLI

#### 导入快照

以下import-snapshot示例将指定的磁盘作为快照导入。

```
aws ec2 import-snapshot \  
  --description "My server VMDK" \  
  --disk-container Format=VMDK,UserBucket={S3Bucket=my-import-bucket,S3Key=vms/my-server-vm.vmdk}
```

输出：

```
{  
  "Description": "My server VMDK",  
  "ImportTaskId": "import-snap-1234567890abcdef0",  
  "SnapshotTaskDetail": {  
    "Description": "My server VMDK",  
    "DiskImageSize": "0.0",  
    "Format": "VMDK",  
    "Progress": "3",  
    "Status": "active",  
    "StatusMessage": "pending"  
    "UserBucket": {  
      "S3Bucket": "my-import-bucket",  
      "S3Key": "vms/my-server-vm.vmdk"  
    }  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ImportSnapshot](#)中的。

## list-images-in-recycle-bin

以下代码示例演示了如何使用 `list-images-in-recycle-bin`。

AWS CLI

列出回收站中的图像

以下`list-images-in-recycle-bin`示例列出了当前保留在回收站中的所有图像。

```
aws ec2 list-images-in-recycle-bin
```

输出：

```
{
```

```
"Images": [
  {
    "RecycleBinEnterTime": "2022-03-14T15:35:08.000Z",
    "Description": "Monthly AMI One",
    "RecycleBinExitTime": "2022-03-15T15:35:08.000Z",
    "Name": "AMI_01",
    "ImageId": "ami-0111222333444abcd"
  }
]
```

有关更多信息，请参阅《[亚马逊弹性计算云用户指南](#)》中的[从回收站中恢复 AMI](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListImagesInRecycleBin](#) 中的。

## list-snapshots-in-recycle-bin

以下代码示例演示了如何使用 `list-snapshots-in-recycle-bin`。

### AWS CLI

#### 查看回收站中的快照

以下 `list-snapshots-in-recycle-bin` 示例列出了有关回收站中快照的信息，包括快照 ID、快照描述、创建快照的卷的 ID、删除快照并进入回收站的日期和时间，以及保留期到期的日期和时间。

```
aws ec2 list-snapshots-in-recycle-bin \
  --snapshot-id snap-01234567890abcdef
```

输出：

```
{
  "SnapshotRecycleBinInfo": [
    {
      "Description": "Monthly data backup snapshot",
      "RecycleBinEnterTime": "2022-12-01T13:00:00.000Z",
      "RecycleBinExitTime": "2022-12-15T13:00:00.000Z",
      "VolumeId": "vol-abcdef09876543210",
      "SnapshotId": "snap-01234567890abcdef"
    }
  ]
}
```



```
}
```

有关 Amazon EBS 回收站的更多信息，请参阅 Amazon EC2 用户指南中的[从回收站恢复快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListSnapshotsInRecycleBin](#) 中的。

## modify-address-attribute

以下代码示例演示了如何使用 modify-address-attribute。

### AWS CLI

修改与弹性 IP 地址关联的域名属性

以下 modify-address-attribute 示例修改弹性 IP 地址的域名属性。

Linux :

```
aws ec2 modify-address-attribute \  
  --allocation-id eipalloc-abcdef01234567890 \  
  --domain-name example.com
```

Windows:

```
aws ec2 modify-address-attribute ^  
  --allocation-id eipalloc-abcdef01234567890 ^  
  --domain-name example.com
```

输出 :

```
{  
  "Addresses": [  
    {  
      "PublicIp": "192.0.2.0",  
      "AllocationId": "eipalloc-abcdef01234567890",  
      "PtrRecord": "example.net."  
      "PtrRecordUpdate": {  
        "Value": "example.com.",  
        "Status": "PENDING"  
      }  
    }  
  ]  
}
```

要监控待处理的更改并查看弹性 IP 地址的修改属性，请参阅 AWS CLI 命令参考[describe-addresses-attribute](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyAddressAttribute](#)中的。

## modify-availability-zone-group

以下代码示例演示了如何使用 modify-availability-zone-group。

### AWS CLI

#### 启用区域组

以下modify-availability-zone-group示例启用了指定的区域组。

```
aws ec2 modify-availability-zone-group \  
  --group-name us-west-2-lax-1 \  
  --opt-in-status opted-in
```

输出：

```
{  
  "Return": true  
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[区域和区域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyAvailabilityZoneGroup](#)中的。

## modify-capacity-reservation-fleet

以下代码示例演示了如何使用 modify-capacity-reservation-fleet。

### AWS CLI

#### 示例 1：修改容量预留队列的总目标容量

以下modify-capacity-reservation-fleet示例修改了指定容量预留队列的总目标容量。当您修改容量预留机群的总目标容量时，机群会自动创建新的容量预留，或者修改或取消机群中的现有容量预留以满足新的总目标容量。当机群处于 modifying 状态时，您无法尝试对其进行其他修改。

```
aws ec2 modify-capacity-reservation-fleet \  
  --capacity-reservation-fleet-id crf-01234567890abcdef \  
  --total-target-capacity 160
```

输出：

```
{  
  "Return": true  
}
```

### 示例 2：修改容量预留队列的结束日期

以下modify-capacity-reservation-fleet示例修改了指定容量预留队列的结束日期。当您修改机群的结束日期时，所有单个容量预留的结束日期都会相应更新。当机群处于 modifying 状态时，您无法尝试对其进行其他修改。

```
aws ec2 modify-capacity-reservation-fleet \  
  --capacity-reservation-fleet-id crf-01234567890abcdef \  
  --end-date 2022-07-04T23:59:59.000Z
```

输出：

```
{  
  "Return": true  
}
```

有关容量预留队列的更多信息，请参阅 Amazon EC2 用户 [指南中的容量预留队列](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyCapacityReservationFleet](#)中的。

## modify-capacity-reservation

以下代码示例演示了如何使用 modify-capacity-reservation。

### AWS CLI

#### 示例 1：更改现有容量预留的实例数量

以下modify-capacity-reservation示例更改了容量预留为其预留容量的实例数量。

```
aws ec2 modify-capacity-reservation \  
  --instance-count 160
```

```
--capacity-reservation-id cr-1234abcd56EXAMPLE \  
--instance-count 5
```

输出：

```
{  
  "Return": true  
}
```

示例 2：更改现有容量预留的结束日期和时间

以下modify-capacity-reservation示例将现有容量预留修改为在指定的日期和时间结束。

```
aws ec2 modify-capacity-reservation \  
  --capacity-reservation-id cr-1234abcd56EXAMPLE \  
  --end-date-type limited \  
  --end-date 2019-08-31T23:59:59Z
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[修改容量预留](#)。

• 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyCapacityReservation](#)中的。

## modify-client-vpn-endpoint

以下代码示例演示了如何使用 modify-client-vpn-endpoint。

AWS CLI

修改 Client VPN 端点

以下modify-client-vpn-endpoint示例为指定的 Client VPN 端点启用客户端连接日志记录。

```
aws ec2 modify-client-vpn-endpoint \  
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde \  
  --connection-log-options Enabled=true,CloudwatchLogGroup=ClientVPNLogs
```

输出：

```
{  
  "Return": true  
}
```

有关更多信息，请参阅《[客户端 VPN 管理员指南](#)》中的[AWS Client VPN 端点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyClientVpnEndpoint](#)中的。

## modify-default-credit-specification

以下代码示例演示了如何使用 modify-default-credit-specification。

### AWS CLI

#### 修改默认积分选项

以下modify-default-credit-specification示例修改了 T2 实例的默认积分选项。

```
aws ec2 modify-default-credit-specification \  
  --instance-family t2 \  
  --cpu-credits unlimited
```

#### 输出：

```
{  
  "InstanceFamilyCreditSpecification": {  
    "InstanceFamily": "t2",  
    "CpuCredits": "unlimited"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyDefaultCreditSpecification](#)中的。

## modify-ebs-default-kms-key-id

以下代码示例演示了如何使用 modify-ebs-default-kms-key-id。

### AWS CLI

#### 为 EBS 加密设置默认 CMK

以下modify-ebs-default-kms-key-id示例将指定的 CMK 设置为当前区域中 AWS 账户的 EBS 加密的默认 CMK。

```
aws ec2 modify-ebs-default-kms-key-id \  
  --key-id <key-id>
```

```
--kms-key-id alias/my-cmk
```

输出：

```
{
  "KmsKeyId": "arn:aws:kms:us-
west-2:123456789012:key/0ea3fef3-80a7-4778-9d8c-1c0c6EXAMPLE"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyEbsDefaultKmsKeyId](#)中的。

## modify-fleet

以下代码示例演示了如何使用 modify-fleet。

AWS CLI

扩展 EC2 队列

以下modify-fleet示例修改了指定 EC2 队列的目标容量。如果指定的值大于当前容量，则 EC2 队列会启动其他实例。如果指定的值小于当前容量，则 EC2 队列会取消所有未处理的请求；如果终止策略是terminate，EC2 队列将终止任何超过新目标容量的实例。

```
aws ec2 modify-fleet \
  --fleet-ids fleet-12a34b55-67cd-8ef9-ba9b-9208dEXAMPLE \
  --target-capacity-specification TotalTargetCapacity=5
```

输出：

```
{
  "Return": true
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[管理 EC2 队列](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyFleet](#)中的。

## modify-fpga-image-attribute

以下代码示例演示了如何使用 modify-fpga-image-attribute。

## AWS CLI

### 修改 Amazon FPGA 镜像的属性

此示例为指定 AFI 123456789012 的账户 ID 添加加载权限。

命令:

```
aws ec2 modify-fpga-image-attribute --attribute loadPermission --fpga-image-id
afi-0d123e123bfc85abc --load-permission Add=[{UserId=123456789012}]
```

输出:

```
{
  "FpgaImageAttribute": {
    "FpgaImageId": "afi-0d123e123bfc85abc",
    "LoadPermissions": [
      {
        "UserId": "123456789012"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyFpgaImageAttribute](#)中的。

## modify-hosts

以下代码示例演示了如何使用 modify-hosts。

### AWS CLI

示例 1：为专用主机启用自动放置

以下modify-hosts示例为专用主机启用自动放置，以便其接受与其实例类型配置相匹配的任何非定向实例启动。

```
aws ec2 modify-hosts \
  --host-id h-06c2f189b4EXAMPLE \
  --auto-placement on
```

输出:

```
{
  "Successful": [
    "h-06c2f189b4EXAMPLE"
  ],
  "Unsuccessful": []
}
```

## 示例 2：为专用主机启用主机恢复

以下modify-hosts示例为指定的专用主机启用主机恢复。

```
aws ec2 modify-hosts \
  --host-id h-06c2f189b4EXAMPLE \
  --host-recovery on
```

输出：

```
{
  "Successful": [
    "h-06c2f189b4EXAMPLE"
  ],
  "Unsuccessful": []
}
```

有关更多信息，请参阅适用于 Linux 实例的 Amazon 弹性计算云用户指南中的[修改专用主机的自动放置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyHosts](#)中的。

## modify-id-format

以下代码示例演示了如何使用 modify-id-format。

### AWS CLI

为资源启用加长 ID 格式

以下modify-id-format示例为instance资源类型启用了加长 ID 格式。

```
aws ec2 modify-id-format \
  --resource instance \
```



```
--use-long-ids
```

### 禁用资源的加长 ID 格式

以下 `modify-id-format` 示例禁用 `instance` 资源类型的加长 ID 格式。

```
aws ec2 modify-id-format \  
  --resource instance \  
  --no-use-long-ids
```

以下 `modify-id-format` 示例为所有处于选择加入期限内的受支持资源类型启用加长 ID 格式。

```
aws ec2 modify-id-format \  
  --resource all-current \  
  --use-long-ids
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyIdFormat](#) 中的。

## modify-identity-id-format

以下代码示例演示了如何使用 `modify-identity-id-format`。

### AWS CLI

让 IAM 角色能够对资源使用较长 ID

以下 `modify-identity-id-format` 示例允许您 AWS 账户中的 IAM 角色 `EC2Role` 对 `instance` 资源类型使用长 ID 格式。

```
aws ec2 modify-identity-id-format \  
  --principal-arn arn:aws:iam::123456789012:role/EC2Role \  
  --resource instance \  
  --use-long-ids
```

让 IAM 用户能够对资源使用较长 ID

以下 `modify-identity-id-format` 示例使您 AWS 账户中的 IAM 用户 `AdminUser` 能够对 `volume` 资源类型使用加长 ID 格式。

```
aws ec2 modify-identity-id-format \  
  --principal-arn arn:aws:iam::123456789012:user/AdminUser \  
  --resource volume \  
  --use-long-ids
```

```
--principal-arn arn:aws:iam::123456789012:user/AdminUser \  
--resource volume \  
--use-long-ids
```

以下 `modify-identity-id-format` 示例使您 AWS 账户中的 IAM 用户能够对 AdminUser 处于其选择期限内的所有支持的资源类型使用较长的 ID 格式。

```
aws ec2 modify-identity-id-format \  
  --principal-arn arn:aws:iam::123456789012:user/AdminUser \  
  --resource all-current \  
  --use-long-ids
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyIdentityIdFormat](#) 中的。

## modify-image-attribute

以下代码示例演示了如何使用 `modify-image-attribute`。

### AWS CLI

示例 1：将 AMI 公开

以下 `modify-instance-attribute` 示例将指定的 AMI 设为公开。

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Add=[{Group=all}]"
```

此命令不生成任何输出。

示例 2：将 AMI 设为私有

以下 `modify-instance-attribute` 示例将指定的 AMI 设为私有。

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Remove=[{Group=all}]"
```

此命令不生成任何输出。

示例 3：向 AWS 账户授予启动权限

以下 `modify-instance-attribute` 示例向指定 AWS 账户授予启动权限。

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Add=[{UserId=123456789012}]"
```

此命令不生成任何输出。

示例 4：移除 AWS 账户的启动权限

以下 `modify-instance-attribute` 示例删除了指定 AWS 账户的启动权限。

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Remove=[{UserId=123456789012}]"
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyImageAttribute](#) 中的。

## `modify-instance-attribute`

以下代码示例演示了如何使用 `modify-instance-attribute`。

### AWS CLI

示例 1：修改实例类型

以下 `modify-instance-attribute` 示例修改了指定实例的实例类型。该实例必须处于 `stopped` 状态。

```
aws ec2 modify-instance-attribute \  
  --instance-id i-1234567890abcdef0 \  
  --instance-type "{\"Value\": \"m1.small\"}"
```

此命令不生成任何输出。

示例 2：在实例上启用增强联网

以下 `modify-instance-attribute` 示例为指定实例启用增强联网。该实例必须处于 `stopped` 状态。

```
aws ec2 modify-instance-attribute \  
  --instance-id i-1234567890abcdef0 \  
  --instance-attribute "{\"Name\": \"enable-elastic-gpu\"}"
```

```
--instance-id i-1234567890abcdef0 \  
--sriov-net-support simple
```

此命令不生成任何输出。

### 示例 3：修改 sourceDestCheck 属性

以下modify-instance-attribute示例将指定实例的sourceDestCheck属性设置为true。该实例必须位于 VPC 中。

```
aws ec2 modify-instance-attribute --instance-id i-1234567890abcdef0 --source-dest-  
check "{\"Value\": true}"
```

此命令不生成任何输出。

### 示例 4：修改根卷的 deleteOnTermination 属性

以下modify-instance-attribute示例将指定 Amazon EBS 支持的实例的根卷的deleteOnTermination属性设置为。false默认情况下，此属性true适用于根卷。

命令:

```
aws ec2 modify-instance-attribute \  
--instance-id i-1234567890abcdef0 \  
--block-device-mappings "[{\"DeviceName\": \"/dev/sda1\", \"Ebs\":  
{\"DeleteOnTermination\": false}}]"
```

此命令不生成任何输出。

### 示例 5：修改附加到实例的用户数据

以下modify-instance-attribute示例将文件内容添加UserData.txt UserData 为指定实例。

原始文件的内容UserData.txt：

```
#!/bin/bash  
yum update -y  
service httpd start  
chkconfig httpd on
```

该文件的内容必须采用 base64 编码。第一个命令将文本文件转换为 base64 并将其另存为新文件。

该命令的 Linux/macOS 版本：

```
base64 UserData.txt > UserData.base64.txt
```

此命令不生成任何输出。

该命令的 Windows 版本：

```
certutil -encode UserData.txt tmp.b64 && findstr /v /c:- tmp.b64 >
UserData.base64.txt
```

输出：

```
Input Length = 67
Output Length = 152
CertUtil: -encode command completed successfully.
```

现在，你可以在下面的 CLI 命令中引用该文件：

```
aws ec2 modify-instance-attribute \
  --instance-id=i-09b5a14dbca622e76 \
  --attribute userData --value file://UserData.base64.txt
```

此命令不生成任何输出。

有关更多信息，请参阅 EC2 用户指南中的用户[数据](#)和 [AWS CLI](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyInstanceAttribute](#) 中的。

## modify-instance-capacity-reservation-attributes

以下代码示例演示了如何使用 modify-instance-capacity-reservation-attributes。

### AWS CLI

示例 1：修改实例的容量预留目标设置

以下 modify-instance-capacity-reservation-attributes 示例将已停止的实例修改为针对特定的容量预留。

```
aws ec2 modify-instance-capacity-reservation-attributes \  
  --instance-id i-EXAMPLE8765abcd4e \  
  --capacity-reservation-specification  
'CapacityReservationTarget={CapacityReservationId= cr-1234abcd56EXAMPLE }'
```

输出：

```
{  
  "Return": true  
}
```

### 示例 2：修改实例的容量预留目标设置

以下 `modify-instance-capacity-reservation-attributes` 示例修改了以指定容量预留为目标的已停止实例，使其在具有匹配属性（实例类型、平台、可用区）且具有开放实例匹配标准的任何容量预留中启动。

```
aws ec2 modify-instance-capacity-reservation-attributes \  
  --instance-id i-EXAMPLE8765abcd4e \  
  --capacity-reservation-specification 'CapacityReservationPreference=open'
```

输出：

```
{  
  "Return": true  
}
```

有关更多信息，请参阅适用于 Linux 实例的 Amazon 弹性计算云用户指南中的修改实例的[容量预留设置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyInstanceCapacityReservationAttributes](#) 中的。

## modify-instance-credit-specification

以下代码示例演示了如何使用 `modify-instance-credit-specification`。

### AWS CLI

修改实例 CPU 使用率的积分选项

此示例将指定区域中指定实例的 CPU 使用率的积分选项修改为“无限制”。有效的信用选项有“标准”和“无限制”。

命令:

```
aws ec2 modify-instance-credit-specification --instance-credit-specification
"InstanceId=i-1234567890abcdef0,CpuCredits=unlimited"
```

输出:

```
{
  "SuccessfulInstanceCreditSpecifications": [
    {
      "InstanceId": "i-1234567890abcdef0"
    }
  ],
  "UnsuccessfulInstanceCreditSpecifications": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyInstanceCreditSpecification](#)中的。

## modify-instance-event-start-time

以下代码示例演示了如何使用 `modify-instance-event-start-time`。

AWS CLI

修改实例的事件开始时间

以下 `modify-instance-event-start-time` 命令显示如何修改指定实例的事件开始时间。使用 `--instance-event-id` 参数指定事件 ID。使用 `--not-before` 参数指定新的日期和时间。

```
aws ec2 modify-instance-event-start-time --instance-id i-1234567890abcdef0
--instance-event-id instance-event-0abcdef1234567890 --not-before
2019-03-25T10:00:00.000
```

输出:

```
"Event": {
  "InstanceEventId": "instance-event-0abcdef1234567890",
```

```

    "Code": "system-reboot",
    "Description": "scheduled reboot",
    "NotAfter": "2019-03-25T12:00:00.000Z",
    "NotBefore": "2019-03-25T10:00:00.000Z",
    "NotBeforeDeadline": "2019-04-22T21:00:00.000Z"
  }

```

有关更多信息，请参阅 Amazon 弹性计算云用户指南中的使用计划重启的实例

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyInstanceEventStartTime](#) 中的。

## modify-instance-event-window

以下代码示例演示了如何使用 `modify-instance-event-window`。

### AWS CLI

#### 示例 1：修改事件窗口的时间范围

以下 `modify-instance-event-window` 示例修改了事件窗口的时间范围。指定 `time-range` 参数以修改时间范围。您不能同时指定 `cron-expression` 参数。

```

aws ec2 modify-instance-event-window \
  --region us-east-1 \
  --instance-event-window-id iew-0abcdef1234567890
  --time-range StartWeekDay=monday,StartHour=2,EndWeekDay=wednesday,EndHour=8

```

输出：

```

{
  "InstanceEventWindow": {
    "InstanceEventWindowId": "iew-0abcdef1234567890",
    "TimeRanges": [
      {
        "StartWeekDay": "monday",
        "StartHour": 2,
        "EndWeekDay": "wednesday",
        "EndHour": 8
      }
    ],
    "Name": "myEventWindowName",
    "AssociationTarget": {

```



```

        "InstanceIds": [
            "i-0abcdef1234567890",
            "i-0be35f9acb8ba01f0"
        ],
        "Tags": [],
        "DedicatedHostIds": []
    },
    "State": "creating",
    "Tags": [
        {
            "Key": "K1",
            "Value": "V1"
        }
    ]
}
}

```

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中计划事件部分的[注意事项](#)。

#### 示例 2：修改事件窗口的一组时间范围

以下 `modify-instance-event-window` 示例修改了事件窗口的时间范围。指定 `time-range` 参数以修改时间范围。您不能同时指定 `cron-expression` 参数。

```

aws ec2 modify-instance-event-window \
  --region us-east-1 \
  --instance-event-window-id iew-0abcdef1234567890 \
  --time-range '[{"StartWeekDay": "monday", "StartHour": 2, "EndWeekDay":
"wednesday", "EndHour": 8},
  {"StartWeekDay": "thursday", "StartHour": 2, "EndWeekDay": "friday",
"EndHour": 8}]'

```

输出：

```

{
  "InstanceEventWindow": {
    "InstanceEventWindowId": "iew-0abcdef1234567890",
    "TimeRanges": [
      {
        "StartWeekDay": "monday",
        "StartHour": 2,
        "EndWeekDay": "wednesday",

```

```

        "EndHour": 8
      },
      {
        "StartWeekDay": "thursday",
        "StartHour": 2,
        "EndWeekDay": "friday",
        "EndHour": 8
      }
    ],
    "Name": "myEventWindowName",
    "AssociationTarget": {
      "InstanceIds": [
        "i-0abcdef1234567890",
        "i-0be35f9acb8ba01f0"
      ],
      "Tags": [],
      "DedicatedHostIds": []
    },
    "State": "creating",
    "Tags": [
      {
        "Key": "K1",
        "Value": "V1"
      }
    ]
  }
}

```

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中计划事件部分的[注意事项](#)。

### 示例 3：修改事件窗口的 cron 表达式

以下 `modify-instance-event-window` 示例修改了事件窗口的 cron 表达式。指定 `cron-expression` 参数以修改 Cron 表达式。您不能同时指定 `time-range` 参数。

```

aws ec2 modify-instance-event-window \
  --region us-east-1 \
  --instance-event-window-id iew-0abcdef1234567890 \
  --cron-expression "* 21-23 * * 2,3"

```

输出：

```
{
```

```
"InstanceEventWindow": {
  "InstanceEventWindowId": "iew-0abcdef1234567890",
  "Name": "myEventWindowName",
  "CronExpression": "* 21-23 * * 2,3",
  "AssociationTarget": {
    "InstanceIds": [
      "i-0abcdef1234567890",
      "i-0be35f9acb8ba01f0"
    ],
    "Tags": [],
    "DedicatedHostIds": []
  },
  "State": "creating",
  "Tags": [
    {
      "Key": "K1",
      "Value": "V1"
    }
  ]
}
```

有关事件窗口限制的信息，请参阅 Amazon EC2 用户指南中计划事件部分的[注意事项](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ModifyInstanceEventWindow](#)中的。

## modify-instance-maintenance-options

以下代码示例演示了如何使用 `modify-instance-maintenance-options`。

### AWS CLI

#### 示例 1：禁用实例的恢复行为

以下 `modify-instance-maintenance-options` 示例禁用正在运行或已停止的实例的简化自动恢复。

```
aws ec2 modify-instance-maintenance-options \
  --instance-id i-0abcdef1234567890 \
  --auto-recovery disabled
```

输出：

```
{
  "InstanceId": "i-0abcdef1234567890",
  "AutoRecovery": "disabled"
}
```

有关更多信息，请参阅《适用于 Linux [实例的 Amazon EC2 用户指南](#)》中的[恢复您的实例](#)。

示例 2：将实例的恢复行为设置为默认值

以下 `modify-instance-maintenance-options` 示例将自动恢复行为设置为默认值，这样可以简化支持的实例类型的自动恢复。

```
aws ec2 modify-instance-maintenance-options \
  --instance-id i-0abcdef1234567890 \
  --auto-recovery default
```

输出：

```
{
  "InstanceId": "i-0abcdef1234567890",
  "AutoRecovery": "default"
}
```

有关更多信息，请参阅《适用于 Linux [实例的 Amazon EC2 用户指南](#)》中的[恢复您的实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyInstanceMaintenanceOptions](#) 中的。

## `modify-instance-metadata-options`

以下代码示例演示了如何使用 `modify-instance-metadata-options`。

### AWS CLI

示例 1：启用 `imdsV2`

以下 `modify-instance-metadata-options` 示例配置了在指定实例上使用 `imdsV2`。

```
aws ec2 modify-instance-metadata-options \
  --instance-id i-1234567898abcdef0 \
  --http-tokens required \
```

```
--http-endpoint enabled
```

输出：

```
{
  "InstanceId": "i-1234567898abcdef0",
  "InstanceMetadataOptions": {
    "State": "pending",
    "HttpTokens": "required",
    "HttpPutResponseHopLimit": 1,
    "HttpEndpoint": "enabled"
  }
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的实例[元数据和用户数据](#)。

#### 示例 2：禁用实例元数据

以下 `modify-instance-metadata-options` 示例禁止在指定实例上使用所有版本的实例元数据。

```
aws ec2 modify-instance-metadata-options \
  --instance-id i-1234567898abcdef0 \
  --http-endpoint disabled
```

输出：

```
{
  "InstanceId": "i-1234567898abcdef0",
  "InstanceMetadataOptions": {
    "State": "pending",
    "HttpTokens": "required",
    "HttpPutResponseHopLimit": 1,
    "HttpEndpoint": "disabled"
  }
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的实例[元数据和用户数据](#)。

### 示例 3：为您的实例启用实例元数据 IPv6 终端节点

以下 `modify-instance-metadata-options` 示例向您展示如何为实例元数据服务打开 IPv6 终端节点。

```
aws ec2 modify-instance-metadata-options \  
  --instance-id i-1234567898abcdef0 \  
  --http-protocol-ipv6 enabled \  
  --http-endpoint enabled
```

输出：

```
{  
  "InstanceId": "i-1234567898abcdef0",  
  "InstanceMetadataOptions": {  
    "State": "pending",  
    "HttpTokens": "required",  
    "HttpPutResponseHopLimit": 1,  
    "HttpEndpoint": "enabled",  
    "HttpProtocolIpv6": "enabled"  
  }  
}
```

默认禁用 IPv6 端点。即使您已在仅 IPv6 子网中启动了实例，也是如此。IMDS 的 IPv6 端点只能在基于 Nitro 系统构建的实例上访问。有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的实例[元数据和用户数据](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyInstanceMetadataOptions](#) 中的。

## modify-instance-placement

以下代码示例演示了如何使用 `modify-instance-placement`。

### AWS CLI

#### 示例 1：移除实例与专用主机的关联性

以下 `modify-instance-placement` 示例删除了实例与专用主机的关联性，使其能够在您的账户中支持其实例类型的任何可用专用主机上启动。

```
aws ec2 modify-instance-placement \  
  --instance-id i-0e6ddf6187EXAMPLE \  
  --placement-type on-demand
```

```
--affinity default
```

输出：

```
{
  "Return": true
}
```

示例 2：在实例和指定的专用主机之间建立关联

以下modify-instance-placement示例在实例和专用主机之间建立了启动关系。该实例只能在指定的专用主机上运行。

```
aws ec2 modify-instance-placement \
  --instance-id i-0e6ddf6187EXAMPLE \
  --affinity host \
  --host-id i-0e6ddf6187EXAMPLE
```

输出：

```
{
  "Return": true
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[“修改实例租期和关联性”](#)。

示例 3：将实例移至置放群组

以下modify-instance-placement示例将实例移至置放群组，停止该实例，修改实例放置位置，然后重启该实例。

```
aws ec2 stop-instances \
  --instance-ids i-0123a456700123456

aws ec2 modify-instance-placement \
  --instance-id i-0123a456700123456 \
  --group-name MySpreadGroup

aws ec2 start-instances \
  --instance-ids i-0123a456700123456
```

有关更多信息，请参阅 Amazon Elastic Compute Cloud 用户指南中的[更改实例的置放群组](#)。

#### 示例 4：从置放群组中移除实例

以下 `modify-instance-placement` 示例通过停止实例、修改实例放置位置然后重启实例，将该实例从置放群组中移除。以下示例为置放群组名称指定了一个空字符串 ("")，以表示该实例不应位于置放群组中。

停止实例：

```
aws ec2 stop-instances \  
  --instance-ids i-0123a456700123456
```

修改放置位置 ( Windows 命令提示符、Linux 和 macOS )：

```
aws ec2 modify-instance-placement \  
  --instance-id i-0123a456700123456 \  
  --group-name ""
```

修改放置位置 (Windows PowerShell)：

```
aws ec2 modify-instance-placement `\  
  --instance-id i-0123a456700123456 `\  
  --group-name ""
```

重启实例：

```
aws ec2 start-instances \  
  --instance-ids i-0123a456700123456
```

输出：

```
{  
  "Return": true  
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的[“修改实例租期和关联性”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyInstancePlacement](#) 中的。



## modify-ipam-pool

以下代码示例演示了如何使用 modify-ipam-pool。

### AWS CLI

#### 修改 IPAM 池

以下 modify-ipam-pool 示例修改 IPAM 池。

(Linux) :

```
aws ec2 modify-ipam-pool \  
  --ipam-pool-id ipam-pool-0533048da7d823723 \  
  --add-allocation-resource-tags "Key=Owner,Value=Build Team" \  
  --clear-allocation-default-netmask-length \  
  --allocation-min-netmask-length 14
```

( 视窗 ) :

```
aws ec2 modify-ipam-pool ^\  
  --ipam-pool-id ipam-pool-0533048da7d823723 ^\  
  --add-allocation-resource-tags "Key=Owner,Value=Build Team" ^\  
  --clear-allocation-default-netmask-length ^\  
  --allocation-min-netmask-length 14
```

输出 :

```
{  
  "IpamPool": {  
    "OwnerId": "123456789012",  
    "IpamPoolId": "ipam-pool-0533048da7d823723",  
    "IpamPoolArn": "arn:aws:ec2::123456789012:ipam-pool/ipam-  
pool-0533048da7d823723",  
    "IpamScopeArn": "arn:aws:ec2::123456789012:ipam-scope/ipam-  
scope-02fc38cd4c48e7d38",  
    "IpamScopeType": "private",  
    "IpamArn": "arn:aws:ec2::123456789012:ipam/ipam-08440e7a3acde3908",  
    "IpamRegion": "us-east-1",  
    "Locale": "None",  
    "PoolDepth": 1,  
    "State": "modify-complete",
```

```
    "AutoImport": true,
    "AddressFamily": "ipv4",
    "AllocationMinNetmaskLength": 14,
    "AllocationMaxNetmaskLength": 26,
    "AllocationResourceTags": [
      {
        "Key": "Environment",
        "Value": "Preprod"
      },
      {
        "Key": "Owner",
        "Value": "Build Team"
      }
    ]
  }
}
```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的[编辑池](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ModifyIpamPool](#)中的。

## modify-ipam-resource-cidr

以下代码示例演示了如何使用 `modify-ipam-resource-cidr`。

### AWS CLI

#### 修改分配给资源的 CIDR

以下 `modify-ipam-resource-cidr` 示例修改了资源 CIDR。

(Linux) :

```
aws ec2 modify-ipam-resource-cidr \
  --current-ipam-scope-id ipam-scope-02fc38cd4c48e7d38 \
  --destination-ipam-scope-id ipam-scope-0da34c61fd189a141 \
  --resource-id vpc-010e1791024eb0af9 \
  --resource-cidr 10.0.1.0/24 \
  --resource-region us-east-1 \
  --monitored
```

( 视窗 ) :

```
aws ec2 modify-ipam-resource-cidr ^
--current-ipam-scope-id ipam-scope-02fc38cd4c48e7d38 ^
--destination-ipam-scope-id ipam-scope-0da34c61fd189a141 ^
--resource-id vpc-010e1791024eb0af9 ^
--resource-cidr 10.0.1.0/24 ^
--resource-region us-east-1 ^
--monitored
```

输出：

```
{
  "IpamResourceCidr": {
    "IpamId": "ipam-08440e7a3acde3908",
    "IpamScopeId": "ipam-scope-0da34c61fd189a141",
    "IpamPoolId": "ipam-pool-0533048da7d823723",
    "ResourceRegion": "us-east-1",
    "ResourceOwnerId": "123456789012",
    "ResourceId": "vpc-010e1791024eb0af9",
    "ResourceCidr": "10.0.1.0/24",
    "ResourceType": "vpc",
    "ResourceTags": [
      {
        "Key": "Environment",
        "Value": "Preprod"
      },
      {
        "Key": "Owner",
        "Value": "Build Team"
      }
    ],
    "IpUsage": 0.0,
    "ComplianceStatus": "noncompliant",
    "ManagementState": "managed",
    "OverlapStatus": "overlapping",
    "VpcId": "vpc-010e1791024eb0af9"
  }
}
```

有关移动资源的更多信息，请参阅 Amazon VPC IPAM 用户指南中的在范围之间移动资源 CID [R](#)。

有关更改[监控状态的更多信息](#)，请参阅 Amazon VPC IPAM 用户指南中的更改资源 CIDR 的监控状态。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyIpamResourceCidr](#)中的。

## modify-ipam-resource-discovery

以下代码示例演示了如何使用 `modify-ipam-resource-discovery`。

### AWS CLI

#### 修改资源发现的操作区域

在此示例中，您是 IPAM 委托的管理员，想要修改资源发现的操作区域。

要完成此请求，请执行以下操作：

您无法修改默认资源发现，并且必须是资源发现的所有者。您需要资源发现 ID，您可以使用它获得。[describe-ipam-resource-discoveries](#)

以下`modify-ipam-resource-discovery`示例修改了您 AWS 账户中的非默认资源发现。

```
aws ec2 modify-ipam-resource-discovery \
  --ipam-resource-discovery-id ipam-res-disco-0f4ef577a9f37a162 \
  --add-operating-regions RegionName='us-west-1' \
  --remove-operating-regions RegionName='us-east-2' \
  --region us-east-1
```

输出：

```
{
  "IpamResourceDiscovery": {
    "OwnerId": "149977607591",
    "IpamResourceDiscoveryId": "ipam-res-disco-0365d2977fc1672fe",
    "IpamResourceDiscoveryArn": "arn:aws:ec2::149977607591:ipam-resource-
discovery/ipam-res-disco-0365d2977fc1672fe",
    "IpamResourceDiscoveryRegion": "us-east-1",
    "Description": "Example",
    "OperatingRegions": [
      {
        "RegionName": "us-east-1"
      },
      {
        "RegionName": "us-west-1"
      }
    ]
  }
}
```

```
    ],  
    "IsDefault": false,  
    "State": "modify-in-progress"  
  }  
}
```

有关更多信息，请参阅《Amazon VPC IPAM 用户指南》中的[使用资源发现](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyIpamResourceDiscovery](#)中的。

## modify-ipam-scope

以下代码示例演示了如何使用 modify-ipam-scope。

### AWS CLI

#### 修改作用域的描述

在这种情况下，您是 IPAM 授权的管理员，想要修改 IPAM 作用域的描述。

要完成此请求，你需要范围 ID，你可以用它获得[describe-ipam-scopes](#)。

以下modify-ipam-scope示例更新了作用域的描述。

```
aws ec2 modify-ipam-scope \  
  --ipam-scope-id ipam-scope-0d3539a30b57dcdd1 \  
  --description example \  
  --region us-east-1
```

输出：

```
{  
  "IpamScope": {  
    "OwnerId": "320805250157",  
    "IpamScopeId": "ipam-scope-0d3539a30b57dcdd1",  
    "IpamScopeArn": "arn:aws:ec2::320805250157:ipam-scope/ipam-  
scope-0d3539a30b57dcdd1",  
    "IpamArn": "arn:aws:ec2::320805250157:ipam/ipam-005f921c17ebd5107",  
    "IpamRegion": "us-east-1",  
    "IpamScopeType": "public",  
    "IsDefault": true,  
    "Description": "example",
```

```
    "PoolCount": 1,  
    "State": "modify-in-progress"  
  }  
}
```

有关范围的更多信息，[请参阅 Amazon VPC IPAM 用户指南中的 IPAM 工作原理](#)。

- 有关 API 的详细信息，[请参阅 AWS CLI 命令参考 `ModifyIpamScope` 中的](#)。

## modify-ipam

以下代码示例演示了如何使用 `modify-ipam`。

### AWS CLI

#### 修改 IPAM

以下 `modify-ipam` 示例通过添加操作区域来修改 IPAM。

(Linux) :

```
aws ec2 modify-ipam \  
  --ipam-id ipam-08440e7a3acde3908 \  
  --add-operating-regions RegionName=us-west-2
```

( 视窗 ) :

```
aws ec2 modify-ipam ^  
  --ipam-id ipam-08440e7a3acde3908 ^  
  --add-operating-regions RegionName=us-west-2
```

输出 :

```
{  
  "Ipam": {  
    "OwnerId": "123456789012",  
    "IpamId": "ipam-08440e7a3acde3908",  
    "IpamArn": "arn:aws:ec2::123456789012:ipam/ipam-08440e7a3acde3908",  
    "IpamRegion": "us-east-1",  
    "PublicDefaultScopeId": "ipam-scope-0b9eed026396dbc16",  
    "PrivateDefaultScopeId": "ipam-scope-02fc38cd4c48e7d38",  
    "ScopeCount": 3,  
  }  
}
```

```
    "OperatingRegions": [  
      {  
        "RegionName": "us-east-1"  
      },  
      {  
        "RegionName": "us-east-2"  
      },  
      {  
        "RegionName": "us-west-1"  
      },  
      {  
        "RegionName": "us-west-2"  
      }  
    ],  
    "State": "modify-in-progress"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyIam](#)中的。

## modify-launch-template

以下代码示例演示了如何使用 modify-launch-template。

### AWS CLI

#### 更改默认启动模板版本

此示例将指定启动模板的版本 2 指定为默认版本。

#### 命令:

```
aws ec2 modify-launch-template --launch-template-id lt-0abcd290751193123 --default-version 2
```

#### 输出:

```
{  
  "LaunchTemplate": {  
    "LatestVersionNumber": 2,  
    "LaunchTemplateId": "lt-0abcd290751193123",  
    "LaunchTemplateName": "WebServers",
```

```
"DefaultVersionNumber": 2,  
"CreatedBy": "arn:aws:iam::123456789012:root",  
"CreateTime": "2017-12-01T13:35:46.000Z"  
}  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyLaunchTemplate](#)中的。

## modify-managed-prefix-list

以下代码示例演示了如何使用 `modify-managed-prefix-list`。

### AWS CLI

#### 修改前缀列表

以下`modify-managed-prefix-list`示例向指定的前缀列表中添加一个条目。

```
aws ec2 modify-managed-prefix-list \  
  --prefix-list-id pl-0123456abcabcabc1 \  
  --add-entries Cidr=10.1.0.0/16,Description=vpc-c \  
  --current-version 1
```

输出：

```
{  
  "PrefixList": {  
    "PrefixListId": "pl-0123456abcabcabc1",  
    "AddressFamily": "IPv4",  
    "State": "modify-in-progress",  
    "PrefixListArn": "arn:aws:ec2:us-west-2:123456789012:prefix-list/  
pl-0123456abcabcabc1",  
    "PrefixListName": "vpc-cidrs",  
    "MaxEntries": 10,  
    "Version": 1,  
    "OwnerId": "123456789012"  
  }  
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的[托管前缀列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyManagedPrefixList](#)中的。



## modify-network-interface-attribute

以下代码示例演示了如何使用 `modify-network-interface-attribute`。

### AWS CLI

#### 修改网络接口的连接属性

此示例命令修改指定网络接口的 `attachment` 属性。

命令:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --attachment AttachmentId=eni-attach-43348162,DeleteOnTermination=false
```

#### 修改网络接口的描述属性

此示例命令修改指定网络接口的 `description` 属性。

命令:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --description "My description"
```

#### 修改网络接口的 GroupSet 属性

此示例命令修改指定网络接口的 `groupSet` 属性。

命令:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --groups sg-903004f8 sg-1a2b3c4d
```

#### 修改网络接口的 sourceDestCheck 属性

此示例命令修改指定网络接口的 `sourceDestCheck` 属性。

命令:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --no-source-dest-check
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyNetworkInterfaceAttribute](#)中的。

## modify-private-dns-name-options

以下代码示例演示了如何使用 `modify-private-dns-name-options`。

### AWS CLI

#### 修改实例主机名的选项

以下 `modify-private-dns-name-options` 示例禁用了响应 DNS 查询的选项，例如带有 DNS A 记录的主机名。

```
aws ec2 modify-private-dns-name-options \  
  --instance-id i-1234567890abcdef0 \  
  --no-enable-resource-name-dns-a-record
```

输出：

```
{  
  "Return": true  
}
```

有关更多信息，请参阅 [Amazon EC2 用户指南中的 Amazon EC2 实例主机名类型](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyPrivateDnsNameOptions](#) 中的。

## modify-reserved-instances

以下代码示例演示了如何使用 `modify-reserved-instances`。

### AWS CLI

#### 修改预留实例

此示例命令将预留实例移动到同一地区的另一个可用区。

命令：

```
aws ec2 modify-reserved-instances --reserved-instances-ids b847fa93-e282-4f55-  
b59a-1342f5bd7c02 --target-configurations AvailabilityZone=us-west-1c,Platform=EC2-  
Classic,InstanceCount=10
```

输出：

```
{
  "ReservedInstancesModificationId": "rimod-d3ed4335-b1d3-4de6-ab31-0f13aaf46687"
}
```

## 修改预留实例的网络平台

此示例命令将 EC2-Classic 预留实例转换为 EC2-VPC。

命令:

```
aws ec2 modify-reserved-instances --reserved-instances-ids f127bd27-edb7-44c9-
a0eb-0d7e09259af0 --target-configurations AvailabilityZone=us-west-1c,Platform=EC2-
VPC,InstanceCount=5
```

输出 :

```
{
  "ReservedInstancesModificationId": "rimod-82fa9020-668f-4fb6-945d-61537009d291"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的修改您的预留实例。

## 修改预留实例的实例大小

此示例命令修改在 us-west-1c 中拥有 10 个 m1.small Linux/UNIX 实例的预留实例，这样 8 个 m1.small 实例变成 2 个 m1.large 实例，剩下的 2 个 m1.small 变成 1 个 m1.medium 实例，位于同一可用区。命令:

```
aws ec2 modify-reserved-instances --reserved-instances-ids
1ba8e2e3-3556-4264-949e-63ee671405a9 --target-configurations AvailabilityZone=us-
west-1c,Platform=EC2-Classic,InstanceCount=2,InstanceType=m1.large
AvailabilityZone=us-west-1c,Platform=EC2-
Classic,InstanceCount=1,InstanceType=m1.medium
```

输出 :

```
{
  "ReservedInstancesModificationId": "rimod-acc5f240-080d-4717-b3e3-1c6b11fa00b6"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的修改预留实例大小。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyReservedInstances](#)中的。

## modify-security-group-rules

以下代码示例演示了如何使用 modify-security-group-rules。

### AWS CLI

修改安全组规则以更新规则描述、IP 协议和 CidrIpv4 地址范围

以下modify-security-group-rules示例更新了指定安全组规则的描述、IP 协议和 IPV4 CIDR 范围。使用security-group-rules参数输入指定安全组规则的更新。 -1指定所有协议。

```
aws ec2 modify-security-group-rules \
  --group-id sg-1234567890abcdef0 \
  --security-group-rules SecurityGroupId=sg-
  abcdef01234567890,SecurityGroupRule='{Description=test,IpProtocol=-1,CidrIpv4=0.0.0.0/0}'
```

输出：

```
{
  "Return": true
}
```

有关安全组规则的更多信息，请参阅 Amazon EC2 用户指南中的[安全组规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifySecurityGroupRules](#)中的。

## modify-snapshot-attribute

以下代码示例演示了如何使用 modify-snapshot-attribute。

### AWS CLI

示例 1：修改快照属性

以下modify-snapshot-attribute示例更新了指定快照的createVolumePermission属性，删除了指定用户的卷权限。

```
aws ec2 modify-snapshot-attribute \
```

```
--snapshot-id snap-1234567890abcdef0 \  
--attribute createVolumePermission \  
--operation-type remove \  
--user-ids 123456789012
```

## 示例 2：公开快照

以下 `modify-snapshot-attribute` 示例将指定的快照设为公开。

```
aws ec2 modify-snapshot-attribute \  
  --snapshot-id snap-1234567890abcdef0 \  
  --attribute createVolumePermission \  
  --operation-type add \  
  --group-names all
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifySnapshotAttribute](#) 中的。

## `modify-snapshot-tier`

以下代码示例演示了如何使用 `modify-snapshot-tier`。

### AWS CLI

#### 示例 1：存档快照

以下 `modify-snapshot-tier` 示例存档了指定的快照。

```
aws ec2 modify-snapshot-tier \  
  --snapshot-id snap-01234567890abcdef \  
  --storage-tier archive
```

输出：

```
{  
  "SnapshotId": "snap-01234567890abcdef",  
  "TieringStartTime": "2021-09-15T16:44:37.574Z"  
}
```

响应参数 `TieringStartTime` 以 UTC 时间格式 (YYYY-MM-DDTHH:MM:SSZ) 表示归档过程的启动日期和时间。

有关快照存档的更多信息，请参阅 [Amazon EC2 用户指南中的存档 Amazon EBS 快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifySnapshotTier](#)中的。

## modify-spot-fleet-request

以下代码示例演示了如何使用 modify-spot-fleet-request。

### AWS CLI

#### 修改竞价型队列请求

此示例命令更新指定 Spot 队列请求的目标容量。

命令:

```
aws ec2 modify-spot-fleet-request --target-capacity 20 --spot-fleet-request-id
sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

输出:

```
{
  "Return": true
}
```

此示例命令减少了指定竞价型队列请求的目标容量，但不会因此终止任何竞价型实例。

命令:

```
aws ec2 modify-spot-fleet-request --target-capacity 10 --excess-capacity-
termination-policy NoTermination --spot-fleet-request-ids sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

输出:

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifySpotFleetRequest](#)中的。

## modify-subnet-attribute

以下代码示例演示了如何使用 `modify-subnet-attribute`。

### AWS CLI

#### 更改子网的公有 IPv4 寻址行为

此示例修改了 `subnet-1a2b3c4d`，以指定在该子网中启动的所有实例都分配一个公有 IPv4 地址。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --map-public-ip-on-launch
```

#### 更改子网的 IPv6 寻址行为

此示例修改了 `subnet-1a2b3c4d`，以指定向该子网启动的所有实例都分配了该子网范围内的 IPv6 地址。

命令:

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --assign-ipv6-address-on-creation
```

有关更多信息，请参阅《AWS 虚拟私有云用户指南》中的 VPC 中的 IP 寻址。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifySubnetAttribute](#) 中的。

## modify-traffic-mirror-filter-network-services

以下代码示例演示了如何使用 `modify-traffic-mirror-filter-network-services`。

### AWS CLI

#### 向流量镜像过滤器添加网络服务

以下 `modify-traffic-mirror-filter-network-services` 示例将 Amazon DNS 网络服务添加到指定的筛选条件中。

```
aws ec2 modify-traffic-mirror-filter-network-services \
```

```
--traffic-mirror-filter-id tmf-04812ff784EXAMPLE \  
--add-network-service amazon-dns
```

输出：

```
{  
  "TrafficMirrorFilter": {  
    "Tags": [  
      {  
        "Key": "Name",  
        "Value": "Production"  
      }  
    ],  
    "EgressFilterRules": [],  
    "NetworkServices": [  
      "amazon-dns"  
    ],  
    "TrafficMirrorFilterId": "tmf-04812ff784EXAMPLE",  
    "IngressFilterRules": [  
      {  
        "SourceCidrBlock": "0.0.0.0/0",  
        "RuleNumber": 1,  
        "DestinationCidrBlock": "0.0.0.0/0",  
        "Description": "TCP Rule",  
        "Protocol": 6,  
        "TrafficDirection": "ingress",  
        "TrafficMirrorFilterId": "tmf-04812ff784EXAMPLE",  
        "RuleAction": "accept",  
        "TrafficMirrorFilterRuleId": "tmf-04812ff784EXAMPLE"  
      }  
    ]  
  }  
}
```

有关更多信息，请参阅《[流量镜像指南](#)》中的[修改AWS 流量镜像过滤器网络服务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyTrafficMirrorFilterNetworkServices](#)中的。

## modify-traffic-mirror-filter-rule

以下代码示例演示了如何使用 modify-traffic-mirror-filter-rule。



## AWS CLI

### 修改流量镜像过滤器规则

以下modify-traffic-mirror-filter-rule示例修改了指定流量镜像过滤规则的描述。

```
aws ec2 modify-traffic-mirror-filter-rule \  
  --traffic-mirror-filter-rule-id tmfr-0ca76e0e08EXAMPLE \  
  --description "TCP Rule"
```

输出：

```
{  
  "TrafficMirrorFilterRule": {  
    "TrafficMirrorFilterRuleId": "tmfr-0ca76e0e08EXAMPLE",  
    "TrafficMirrorFilterId": "tmf-0293f26e86EXAMPLE",  
    "TrafficDirection": "ingress",  
    "RuleNumber": 100,  
    "RuleAction": "accept",  
    "Protocol": 6,  
    "DestinationCidrBlock": "10.0.0.0/24",  
    "SourceCidrBlock": "10.0.0.0/24",  
    "Description": "TCP Rule"  
  }  
}
```

有关更多信息，请参阅《[流量镜像指南](#)》中的[修改您的AWS 流量镜像过滤器规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyTrafficMirrorFilterRule](#)中的。

## modify-traffic-mirror-session

以下代码示例演示了如何使用 modify-traffic-mirror-session。

## AWS CLI

### 修改流量镜像会话

以下modify-traffic-mirror-session示例更改了流量镜像会话描述和要镜像的数据包数量。

```
aws ec2 modify-traffic-mirror-session \  
  --description "Change packet length" \  
  --traffic-mirror-session-id tms-08a33b1214EXAMPLE \  
  --packet-length 100
```

```
--remove-fields "packet-length"
```

输出：

```
{
  "TrafficMirrorSession": {
    "TrafficMirrorSessionId": "tms-08a33b1214EXAMPLE",
    "TrafficMirrorTargetId": "tmt-07f75d8feeEXAMPLE",
    "TrafficMirrorFilterId": "tmf-04812ff784EXAMPLE",
    "NetworkInterfaceId": "eni-070203f901EXAMPLE",
    "OwnerId": "111122223333",
    "SessionNumber": 1,
    "VirtualNetworkId": 7159709,
    "Description": "Change packet length",
    "Tags": []
  }
}
```

有关更多信息，请参阅《[流量镜像指南](#)》中的[修改您的流量镜像会话](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyTrafficMirrorSession](#)中的。

## modify-transit-gateway-prefix-list-reference

以下代码示例演示了如何使用 modify-transit-gateway-prefix-list-reference。

### AWS CLI

修改对前缀列表的引用

以下modify-transit-gateway-prefix-list-reference示例通过更改流量路由到的附件来修改指定路由表中的前缀列表引用。

```
aws ec2 modify-transit-gateway-prefix-list-reference \
  --transit-gateway-route-table-id tgw-rtb-0123456789abcd123 \
  --prefix-list-id pl-11111122222222333 \
  --transit-gateway-attachment-id tgw-attach-aabbccddaabbccaab
```

输出：

```
{
  "TransitGatewayPrefixListReference": {
```

```

    "TransitGatewayRouteTableId": "tgw-rtb-0123456789abcd123",
    "PrefixListId": "pl-1111112222222333",
    "PrefixListOwnerId": "123456789012",
    "State": "modifying",
    "Blackhole": false,
    "TransitGatewayAttachment": {
      "TransitGatewayAttachmentId": "tgw-attach-aabbccddaabbccaab",
      "ResourceType": "vpc",
      "ResourceId": "vpc-112233445566aabbcc"
    }
  }
}

```

有关更多信息，请参阅 [Transit Gateways 指南](#) 中的 [前缀列表参考](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [Modify Transit Gateway Prefix List Reference](#) 中的。

## modify-transit-gateway-vpc-attachment

以下代码示例演示了如何使用 `modify-transit-gateway-vpc-attachment`。

### AWS CLI

#### 修改传输网关 VPC 连接

以下 `modify-transit-gateway-vpc-attachment` 示例向指定的传输网关 VPC 连接添加子网。

```

aws ec2 modify-transit-gateway-vpc-attachment \
  --transit-gateway-attachment-id tgw-attach-09fbd47ddfEXAMPLE \
  --add-subnet-ids subnet-0e51f45802EXAMPLE

```

输出：

```

{
  "TransitGatewayVpcAttachment": {
    "TransitGatewayAttachmentId": "tgw-attach-09fbd47ddfEXAMPLE",
    "TransitGatewayId": "tgw-0560315ccfEXAMPLE",
    "VpcId": "vpc-5eccc927",
    "VpcOwnerId": "111122223333",
    "State": "modifying",
  }
}

```

```
    "SubnetIds": [
      "subnet-0e51f45802EXAMPLE",
      "subnet-1EXAMPLE"
    ],
    "CreationTime": "2019-08-08T16:47:38.000Z",
    "Options": {
      "DnsSupport": "enable",
      "Ipv6Support": "disable"
    }
  }
}
```

有关更多信息，请参阅 [《传输网关指南》中的连接到 VPC 的传输网关](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyTransitGatewayVpcAttachment](#) 中的。

## modify-transit-gateway

以下代码示例演示了如何使用 `modify-transit-gateway`。

### AWS CLI

#### 修改传输网关

以下 `modify-transit-gateway` 示例通过启用 ECMP 对 VPN 附件的支持来修改指定的传输网关。

```
aws ec2 modify-transit-gateway \
  --transit-gateway-id tgw-111111222222aaaaa \
  --options VpnEcmpSupport=enable
```

输出：

```
{
  "TransitGateway": {
    "TransitGatewayId": "tgw-111111222222aaaaa",
    "TransitGatewayArn": "64512",
    "State": "modifying",
    "OwnerId": "123456789012",
    "CreationTime": "2020-04-30T08:41:37.000Z",
    "Options": {
      "AmazonSideAsn": 64512,
```

```

        "AutoAcceptSharedAttachments": "disable",
        "DefaultRouteTableAssociation": "enable",
        "AssociationDefaultRouteTableId": "tgw-rtb-0123456789abcd123",
        "DefaultRouteTablePropagation": "enable",
        "PropagationDefaultRouteTableId": "tgw-rtb-0123456789abcd123",
        "VpnEcmpSupport": "enable",
        "DnsSupport": "enable"
    }
}
}

```

有关更多信息，请参阅《[传输网关指南](#)》中的中转网关。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyTransitGateway](#)中的。

## modify-verified-access-endpoint-policy

以下代码示例演示了如何使用 modify-verified-access-endpoint-policy。

### AWS CLI

为终端节点配置“已验证访问权限”策略

以下modify-verified-access-endpoint-policy示例将指定的已验证访问策略添加到指定的已验证访问终端节点。

```

aws ec2 modify-verified-access-endpoint-policy \
  --verified-access-endpoint-id vae-066fac616d4d546f2 \
  --policy-enabled \
  --policy-document file://policy.txt

```

policy.txt 的内容：

```

permit(principal,action,resource)
when {
  context.identity.groups.contains("finance") &&
  context.identity.email.verified == true
};

```

输出：

```
{
```

```

    "PolicyEnabled": true,
    "PolicyDocument": "permit(principal,action,resource)\nwhen
{\n    context.identity.groups.contains(\"finance\") &&\n
context.identity.email_verified == true\n};"
}

```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyVerifiedAccessEndpointPolicy](#) 中的。

## modify-verified-access-endpoint

以下代码示例演示了如何使用 modify-verified-access-endpoint。

### AWS CLI

#### 修改已验证访问终端节点的配置

以下 modify-verified-access-endpoint 示例将指定的描述添加到指定的已验证访问终端节点。

```

aws ec2 modify-verified-access-endpoint \
  --verified-access-endpoint-id vae-066fac616d4d546f2 \
  --description "Testing Verified Access"

```

输出：

```

{
  "VerifiedAccessEndpoint": {
    "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",
    "VerifiedAccessGroupId": "vagr-0dbe967baf14b7235",
    "VerifiedAccessEndpointId": "vae-066fac616d4d546f2",
    "ApplicationDomain": "example.com",
    "EndpointType": "network-interface",
    "AttachmentType": "vpc",
    "DomainCertificateArn": "arn:aws:acm:us-east-2:123456789012:certificate/
eb065ea0-26f9-4e75-a6ce-0a1a7EXAMPLE",
    "EndpointDomain": "my-ava-
app.edge-00c3372d53b1540bb.vai-0ce000c0b7643abea.prod.verified-access.us-
east-2.amazonaws.com",
    "SecurityGroupIds": [
      "sg-004915970c4c8f13a"
    ]
  }
}

```

```
    ],
    "NetworkInterfaceOptions": {
      "NetworkInterfaceId": "eni-0aec70418c8d87a0f",
      "Protocol": "https",
      "Port": 443
    },
    "Status": {
      "Code": "updating"
    },
    "Description": "Testing Verified Access",
    "CreationTime": "2023-08-25T20:54:43",
    "LastUpdatedTime": "2023-08-25T22:46:32"
  }
}
```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问终端节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyVerifiedAccessEndpoint](#) 中的。

## modify-verified-access-group-policy

以下代码示例演示了如何使用 `modify-verified-access-group-policy`。

### AWS CLI

为组配置“已验证访问权限”策略

以下 `modify-verified-access-group-policy` 示例将指定的“已验证访问权限”策略添加到指定的已验证访问权限组。

```
aws ec2 modify-verified-access-group-policy \
  --verified-access-group-id vagr-0dbe967baf14b7235 \
  --policy-enabled \
  --policy-document file://policy.txt
```

`policy.txt` 的内容：

```
permit(principal,action,resource)
when {
  context.identity.groups.contains("finance") &&
  context.identity.email.verified == true
};
```

输出：

```
{
  "PolicyEnabled": true,
  "PolicyDocument": "permit(principal,action,resource)\nwhen
{\n  context.identity.groups.contains(\"finance\") &&\n
context.identity.email_verified == true\n};"
}
```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyVerifiedAccessGroupPolicy](#) 中的。

## modify-verified-access-group

以下代码示例演示了如何使用 modify-verified-access-group。

### AWS CLI

修改已验证访问权限组的配置

以下 modify-verified-access-group 示例将指定的描述添加到指定的“已验证访问权限”组。

```
aws ec2 modify-verified-access-group \
  --verified-access-group-id vagr-0dbe967baf14b7235 \
  --description "Testing Verified Access"
```

输出：

```
{
  "VerifiedAccessGroup": {
    "VerifiedAccessGroupId": "vagr-0dbe967baf14b7235",
    "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",
    "Description": "Testing Verified Access",
    "Owner": "123456789012",
    "VerifiedAccessGroupArn": "arn:aws:ec2:us-east-2:123456789012:verified-
access-group/vagr-0dbe967baf14b7235",
    "CreationTime": "2023-08-25T19:55:19",
    "LastUpdatedTime": "2023-08-25T22:17:25"
  }
}
```



有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyVerifiedAccessGroup](#) 中的。

## modify-verified-access-instance-logging-configuration

以下代码示例演示了如何使用 `modify-verified-access-instance-logging-configuration`。

### AWS CLI

为已验证的访问权限实例启用日志记录

以下 `modify-verified-access-instance-logging-configuration` 示例为指定的 Verified Access 实例启用访问日志记录。日志将传送到指定的 Lo CloudWatch gs 日志组。

```
aws ec2 modify-verified-access-instance-logging-configuration \  
  --verified-access-instance-id vai-0ce000c0b7643abea \  
  --access-logs CloudWatchLogs={Enabled=true,LogGroup=my-log-group}
```

输出：

```
{  
  "LoggingConfiguration": {  
    "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",  
    "AccessLogs": {  
      "S3": {  
        "Enabled": false  
      },  
      "CloudWatchLogs": {  
        "Enabled": true,  
        "DeliveryStatus": {  
          "Code": "success"  
        },  
        "LogGroup": "my-log-group"  
      },  
      "KinesisDataFirehose": {  
        "Enabled": false  
      },  
      "LogVersion": "ocsf-1.0.0-rc.2",  
      "IncludeTrustContext": false  
    }  
  }  
}
```

```
}
```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问日志](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVerifiedAccessInstanceLoggingConfiguration](#)中的。

## modify-verified-access-instance

以下代码示例演示了如何使用 `modify-verified-access-instance`。

### AWS CLI

#### 修改已验证访问权限实例的配置

以下`modify-verified-access-instance`示例将指定的描述添加到指定的已验证访问权限实例。

```
aws ec2 modify-verified-access-instance \
  --verified-access-instance-id vai-0ce000c0b7643abea \
  --description "Testing Verified Access"
```

输出：

```
{
  "VerifiedAccessInstance": {
    "VerifiedAccessInstanceId": "vai-0ce000c0b7643abea",
    "Description": "Testing Verified Access",
    "VerifiedAccessTrustProviders": [
      {
        "VerifiedAccessTrustProviderId": "vatp-0bb32de759a3e19e7",
        "TrustProviderType": "user",
        "UserTrustProviderType": "iam-identity-center"
      }
    ],
    "CreationTime": "2023-08-25T18:27:56",
    "LastUpdatedTime": "2023-08-25T22:41:04"
  }
}
```

有关更多信息，请参阅《[已验证访问用户指南](#)》中的[AWS 已验证访问实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVerifiedAccessInstance](#)中的。

## modify-verified-access-trust-provider

以下代码示例演示了如何使用 `modify-verified-access-trust-provider`。

### AWS CLI

修改已验证访问信任提供商的配置

以下 `modify-verified-access-trust-provider` 示例将指定的描述添加到指定的已验证访问信任提供商。

```
aws ec2 modify-verified-access-trust-provider \  
  --verified-access-trust-provider-id vatp-0bb32de759a3e19e7 \  
  --description "Testing Verified Access"
```

输出：

```
{  
  "VerifiedAccessTrustProvider": {  
    "VerifiedAccessTrustProviderId": "vatp-0bb32de759a3e19e7",  
    "Description": "Testing Verified Access",  
    "TrustProviderType": "user",  
    "UserTrustProviderType": "iam-identity-center",  
    "PolicyReferenceName": "idc",  
    "CreationTime": "2023-08-25T19:00:38",  
    "LastUpdatedTime": "2023-08-25T19:18:21"  
  }  
}
```

有关更多信息，请参阅《已验证访问用户指南》中的“AWS 验证访问权限的[信任提供商](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyVerifiedAccessTrustProvider](#) 中的。

## modify-volume-attribute

以下代码示例演示了如何使用 `modify-volume-attribute`。

### AWS CLI

修改体积属性

此示例将 ID 为 `vol-1234567890abcdef0` 的卷的 `autoEnableIo` 属性设置 `vol-1234567890abcdef0` 为 `true`。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 modify-volume-attribute --volume-id vol-1234567890abcdef0 --auto-enable-io
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVolumeAttribute](#)中的。

## modify-volume

以下代码示例演示了如何使用 modify-volume。

### AWS CLI

示例 1：通过更改卷大小来修改卷

以下modify-volume示例将指定卷的大小更改为 150GB。

命令:

```
aws ec2 modify-volume --size 150 --volume-id vol-1234567890abcdef0
```

输出：

```
{
  "VolumeModification": {
    "TargetSize": 150,
    "TargetVolumeType": "io1",
    "ModificationState": "modifying",
    "VolumeId": " vol-1234567890abcdef0",
    "TargetIops": 100,
    "StartTime": "2019-05-17T11:27:19.000Z",
    "Progress": 0,
    "OriginalVolumeType": "io1",
    "OriginalIops": 100,
    "OriginalSize": 100
  }
}
```

示例 2：通过更改卷的类型、大小和 IOPS 值来修改卷

以下modify-volume示例将卷类型更改为预配置 IOPS 固态硬盘，将目标 IOPS 速率设置为 10000，并将卷大小设置为 350GB。

```
aws ec2 modify-volume \  
  --volume-type io1 \  
  --iops 10000 \  
  --size 350 \  
  --volume-id vol-1234567890abcdef0
```

输出：

```
{  
  "VolumeModification": {  
    "TargetSize": 350,  
    "TargetVolumeType": "io1",  
    "ModificationState": "modifying",  
    "VolumeId": "vol-0721c1a9d08c93bf6",  
    "TargetIops": 10000,  
    "StartTime": "2019-05-17T11:38:57.000Z",  
    "Progress": 0,  
    "OriginalVolumeType": "gp2",  
    "OriginalIops": 150,  
    "OriginalSize": 50  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVolume](#)中的。

## modify-vpc-attribute

以下代码示例演示了如何使用 modify-vpc-attribute。

### AWS CLI

#### 修改 enableDnsSupport 属性

此示例修改了该enableDnsSupport属性。此属性指示 VPC 是否已启用 DNS 解析。如果该属性为 true，则 Amazon DNS 服务器会将您实例的 DNS 主机名称解析为相应的 IP 地址，否则不会解析。如果命令成功，则不返回任何输出。

命令：

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-support "{\"Value  
\":false}"
```

## 修改 enableDnsHostnames 属性

此示例修改了该enableDnsHostnames属性。此属性表示在 VPC 中启动的实例是否获得 DNS 主机名。如果该属性为 true，则 VPC 内的实例可获得 DNS 主机名称，否则将无法获得。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-hostnames "{\"Value\":false}"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVpcAttribute](#)中的。

## modify-vpc-endpoint-connection-notification

以下代码示例演示了如何使用 modify-vpc-endpoint-connection-notification。

AWS CLI

### 修改端点连接通知

此示例更改了指定终端节点连接通知的 SNS 主题。

命令:

```
aws ec2 modify-vpc-endpoint-connection-notification --connection-notification-id vpce-nfn-008776de7e03f5abc --connection-events Accept Reject --connection-notification-arn arn:aws:sns:us-east-2:123456789012:mytopic
```

输出:

```
{
  "ReturnValue": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVpcEndpointConnectionNotification](#)中的。

## modify-vpc-endpoint-service-configuration

以下代码示例演示了如何使用 modify-vpc-endpoint-service-configuration。

## AWS CLI

### 修改终端节点服务配置

此示例更改了指定端点服务的接受要求。

命令:

```
aws ec2 modify-vpc-endpoint-service-configuration --service-id vpce-  
svc-09222513e6e77dc86 --no-acceptance-required
```

输出 :

```
{  
  "ReturnValue": true  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVpcEndpointServiceConfiguration](#)中的。

## modify-vpc-endpoint-service-payer-responsibility

以下代码示例演示了如何使用 `modify-vpc-endpoint-service-payer-responsibility`。

### AWS CLI

#### 修改付款人责任

以下`modify-vpc-endpoint-service-payer-responsibility`示例修改了指定终端节点服务的付款人责任。

```
aws ec2 modify-vpc-endpoint-service-payer-responsibility \  
  --service-id vpce-svc-071afff70666e61e0 \  
  --payer-responsibility ServiceOwner
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVpcEndpointServicePayerResponsibility](#)中的。

## modify-vpc-endpoint-service-permissions

以下代码示例演示了如何使用 `modify-vpc-endpoint-service-permissions`。

## AWS CLI

### 修改终端节点服务权限

此示例为 AWS 账户添加了连接到指定终端节点服务的权限。

命令:

```
aws ec2 modify-vpc-endpoint-service-permissions --service-id vpce-  
svc-03d5ebb7d9579a2b3 --add-allowed-principals '["arn:aws:iam::123456789012:root"]'
```

输出:

```
{  
  "ReturnValue": true  
}
```

此示例为特定 IAM 用户 (admin) 添加了连接到指定终端节点服务的权限。

命令:

```
aws ec2 modify-vpc-endpoint-service-permissions --service-id vpce-  
svc-03d5ebb7d9579a2b3 --add-allowed-principals '["arn:aws:iam::123456789012:user/  
admin"]'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVpcEndpointServicePermissions](#)中的。

## modify-vpc-endpoint

以下代码示例演示了如何使用 modify-vpc-endpoint。

## AWS CLI

### 修改网关终端节点

此示例vpce-1a2b3c4d通过将路由表rtb-aaa222bb与终端节点关联并重置策略文档来修改网关终端节点。

命令:

```
aws ec2 modify-vpc-endpoint --vpc-endpoint-id vpce-1a2b3c4d --add-route-table-ids  
rtb-aaa222bb --reset-policy
```



输出：

```
{
  "Return": true
}
```

### 修改接口终端节点

此示例通过向终端节点添加子网vpce-0fe5b17a0707d6fa5subnet-d6fcaa8d来修改接口终端节点。

命令：

```
aws ec2 modify-vpc-endpoint --vpc-endpoint-id vpce-0fe5b17a0707d6fa5 --add-subnet-id
subnet-d6fcaa8d
```

输出：

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVpcEndpoint](#)中的。

## modify-vpc-peering-connection-options

以下代码示例演示了如何使用 modify-vpc-peering-connection-options。

### AWS CLI

启用通过本地 ClassicLink 连接通过 VPC 对等连接进行通信

在此示例中，对于对等连接pcx-aaaabbbb，请求者 VPC 的所有者修改 VPC 对等连接选项，使本地 ClassicLink 连接能够与对等 VPC 通信。

命令：

```
aws ec2 modify-vpc-peering-connection-options --vpc-peering-
connection-id pcx-aaaabbbb --requester-peering-connection-options
AllowEgressFromLocalClassicLinkToRemoteVpc=true
```

输出：

```
{
  "RequesterPeeringConnectionOptions": {
    "AllowEgressFromLocalClassicLinkToRemoteVpc": true
  }
}
```

启用通过 VPC 对等连接从本地 VPC 到远程连接的通 ClassicLink 信

在此示例中，接受方 VPC 的所有者修改 VPC 对等连接选项，使本地 VPC 能够与对等 VPC 中的 ClassicLink 连接进行通信。

命令：

```
aws ec2 modify-vpc-peering-connection-options --vpc-peering-connection-id pcx-aaaabbbb --accepter-peering-connection-options AllowEgressFromLocalVpcToRemoteClassicLink=true
```

输出：

```
{
  "AcceptorPeeringConnectionOptions": {
    "AllowEgressFromLocalVpcToRemoteClassicLink": true
  }
}
```

为 VPC 对等连接启用 DNS 解析支持

在此示例中，请求者 VPC 的所有者修改了的 VPC 对等连接选项，pcx-aaaabbbb 以便当从对等 VPC 中的实例进行查询时，本地 VPC 能够将公有 DNS 主机名解析为私有 IP 地址。

命令：

```
aws ec2 modify-vpc-peering-connection-options --vpc-peering-connection-id pcx-aaaabbbb --requester-peering-connection-options AllowDnsResolutionFromRemoteVpc=true
```

输出：

```
{
  "RequesterPeeringConnectionOptions": {
    "AllowDnsResolutionFromRemoteVpc": true
  }
}
```

```
}  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVpcPeeringConnectionOptions](#)中的。

## modify-vpc-tenancy

以下代码示例演示了如何使用 `modify-vpc-tenancy`。

### AWS CLI

修改 VPC 的租约

此示例将 VPC `vpc-1a2b3c4d` 的租约修改为 `default`

命令:

```
aws ec2 modify-vpc-tenancy --vpc-id vpc-1a2b3c4d --instance-tenancy default
```

输出:

```
{  
  "Return": true  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVpcTenancy](#)中的。

## modify-vpn-connection-options

以下代码示例演示了如何使用 `modify-vpn-connection-options`。

### AWS CLI

修改您的 VPN 连接选项

以下`modify-vpn-connection-options`示例修改了指定 VPN 连接的客户网关端的本地 IPv4 CIDR。

```
aws ec2 modify-vpn-connection-options \  
  --vpn-connection-id vpn-1122334455aabbccd \  
  --local-ipv4-network-cidr 10.0.0.0/16
```

输出：

```
{
  "VpnConnections": [
    {
      "CustomerGatewayConfiguration": "...configuration information...",
      "CustomerGatewayId": "cgw-01234567abcde1234",
      "Category": "VPN",
      "State": "modifying",
      "Type": "ipsec.1",
      "VpnConnectionId": "vpn-1122334455aabbccd",
      "TransitGatewayId": "tgw-00112233445566aab",
      "Options": {
        "EnableAcceleration": false,
        "StaticRoutesOnly": true,
        "LocalIpv4NetworkCidr": "10.0.0.0/16",
        "RemoteIpv4NetworkCidr": "0.0.0.0/0",
        "TunnelInsideIpVersion": "ipv4"
      },
      "Routes": [],
      "Tags": [
        {
          "Key": "Name",
          "Value": "CanadaVPN"
        }
      ],
      "VgwTelemetry": [
        {
          "AcceptedRouteCount": 0,
          "LastStatusChange": "2020-07-29T10:35:11.000Z",
          "OutsideIpAddress": "203.0.113.3",
          "Status": "DOWN",
          "StatusMessage": ""
        },
        {
          "AcceptedRouteCount": 0,
          "LastStatusChange": "2020-09-02T09:09:33.000Z",
          "OutsideIpAddress": "203.0.113.5",
          "Status": "UP",
          "StatusMessage": ""
        }
      ]
    }
  ]
}
```

```
}
```

有关更多信息，请参阅《[站点到站点 VPN 用户指南](#)》中的“[修改站点到站点 VPN 连接选项](#)”。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVpnConnectionOptions](#)中的。

## modify-vpn-connection

以下代码示例演示了如何使用 modify-vpn-connection。

### AWS CLI

#### 修改 VPN 连接

以下modify-vpn-connection示例将 VPN 连接的目标网关更改vpn-12345678901234567为虚拟专用网关vgw-11223344556677889：

```
aws ec2 modify-vpn-connection \  
  --vpn-connection-id vpn-12345678901234567 \  
  --vpn-gateway-id vgw-11223344556677889
```

输出：

```
{  
  "VpnConnection": {  
    "CustomerGatewayConfiguration": "...configuration information...",  
    "CustomerGatewayId": "cgw-aabbccdee1122334",  
    "Category": "VPN",  
    "State": "modifying",  
    "Type": "ipsec.1",  
    "VpnConnectionId": "vpn-12345678901234567",  
    "VpnGatewayId": "vgw-11223344556677889",  
    "Options": {  
      "StaticRoutesOnly": false  
    },  
    "VgwTelemetry": [  
      {  
        "AcceptedRouteCount": 0,  
        "LastStatusChange": "2019-07-17T07:34:00.000Z",  
        "OutsideIpAddress": "18.210.3.222",  
        "Status": "DOWN",  
        "StatusMessage": "IPSEC IS DOWN"  
      }  
    ]  
  }  
}
```

```

    },
    {
      "AcceptedRouteCount": 0,
      "LastStatusChange": "2019-07-20T21:20:16.000Z",
      "OutsideIpAddress": "34.193.129.33",
      "Status": "DOWN",
      "StatusMessage": "IPSEC IS DOWN"
    }
  ]
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVpnConnection](#)中的。

## modify-vpn-tunnel-certificate

以下代码示例演示了如何使用 `modify-vpn-tunnel-certificate`。

### AWS CLI

#### 轮换 VPN 隧道证书

以下`modify-vpn-tunnel-certificate`示例轮换 VPN 连接的指定隧道的证书

```

aws ec2 modify-vpn-tunnel-certificate \
  --vpn-tunnel-outside-ip-address 203.0.113.17 \
  --vpn-connection-id vpn-12345678901234567

```

输出：

```

{
  "VpnConnection": {
    "CustomerGatewayConfiguration": "...configuration information...",
    "CustomerGatewayId": "cgw-aabbccdde1122334",
    "Category": "VPN",
    "State": "modifying",
    "Type": "ipsec.1",
    "VpnConnectionId": "vpn-12345678901234567",
    "VpnGatewayId": "vgw-11223344556677889",
    "Options": {
      "StaticRoutesOnly": false
    }
  },

```

```

    "VgwTelemetry": [
      {
        "AcceptedRouteCount": 0,
        "LastStatusChange": "2019-09-11T17:27:14.000Z",
        "OutsideIpAddress": "203.0.113.17",
        "Status": "DOWN",
        "StatusMessage": "IPSEC IS DOWN",
        "CertificateArn": "arn:aws:acm:us-east-1:123456789101:certificate/c544d8ce-20b8-4fff-98b0-example"
      },
      {
        "AcceptedRouteCount": 0,
        "LastStatusChange": "2019-09-11T17:26:47.000Z",
        "OutsideIpAddress": "203.0.114.18",
        "Status": "DOWN",
        "StatusMessage": "IPSEC IS DOWN",
        "CertificateArn": "arn:aws:acm:us-east-1:123456789101:certificate/5ab64566-761b-4ad3-b259-example"
      }
    ]
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVpnTunnelCertificate](#)中的。

## modify-vpn-tunnel-options

以下代码示例演示了如何使用 modify-vpn-tunnel-options。

### AWS CLI

#### 修改 VPN 连接的隧道选项

以下modify-vpn-tunnel-options示例更新了允许使用指定隧道和 VPN 连接的 Diffie-Hellman 组。

```

aws ec2 modify-vpn-tunnel-options \
  --vpn-connection-id vpn-12345678901234567 \
  --vpn-tunnel-outside-ip-address 203.0.113.17 \
  --tunnel-options Phase1DHGroupNumbers=[{Value=14},{Value=15},{Value=16},
{Value=17},{Value=18}],Phase2DHGroupNumbers=[{Value=14},{Value=15},{Value=16},
{Value=17},{Value=18}]

```

输出：

```
{
  "VpnConnection": {
    "CustomerGatewayConfiguration": "...configuration information...",
    "CustomerGatewayId": "cgw-aabbccdde1122334",
    "Category": "VPN",
    "State": "available",
    "Type": "ipsec.1",
    "VpnConnectionId": "vpn-12345678901234567",
    "VpnGatewayId": "vgw-11223344556677889",
    "Options": {
      "StaticRoutesOnly": false,
      "TunnelOptions": [
        {
          "OutsideIpAddress": "203.0.113.17",
          "Phase1DHGroupNumbers": [
            {
              "Value": 14
            },
            {
              "Value": 15
            },
            {
              "Value": 16
            },
            {
              "Value": 17
            },
            {
              "Value": 18
            }
          ],
          "Phase2DHGroupNumbers": [
            {
              "Value": 14
            },
            {
              "Value": 15
            },
            {
              "Value": 16
            }
          ]
        }
      ]
    }
  }
}
```



```

        "Value": 17
      },
      {
        "Value": 18
      }
    ]
  },
  {
    "OutsideIpAddress": "203.0.114.19"
  }
]
},
"VgwTelemetry": [
  {
    "AcceptedRouteCount": 0,
    "LastStatusChange": "2019-09-10T21:56:54.000Z",
    "OutsideIpAddress": "203.0.113.17",
    "Status": "DOWN",
    "StatusMessage": "IPSEC IS DOWN"
  },
  {
    "AcceptedRouteCount": 0,
    "LastStatusChange": "2019-09-10T21:56:43.000Z",
    "OutsideIpAddress": "203.0.114.19",
    "Status": "DOWN",
    "StatusMessage": "IPSEC IS DOWN"
  }
]
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyVpnTunnelOptions](#)中的。

## monitor-instances

以下代码示例演示了如何使用 monitor-instances。

### AWS CLI

启用对实例的详细监控

本示例命令启用对指定实例的详细监控。

命令:

```
aws ec2 monitor-instances --instance-ids i-1234567890abcdef0
```

输出:

```
{
  "InstanceMonitorings": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "Monitoring": {
        "State": "pending"
      }
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[MonitorInstances](#)中的。

## move-address-to-vpc

以下代码示例演示了如何使用 `move-address-to-vpc`。

AWS CLI

将地址移至 EC2-VPC

此示例将弹性 IP 地址 54.123.4.56 移至 EC2-VPC 平台。

命令:

```
aws ec2 move-address-to-vpc --public-ip 54.123.4.56
```

输出:

```
{
  "Status": "MoveInProgress"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[MoveAddressToVpc](#)中的。

## move-byoip-cidr-to-ipam

以下代码示例演示了如何使用 `move-byoip-cidr-to-ipam`。

### AWS CLI

将 BYOIP CIDR 传输到 IPAM

以下 `move-byoip-cidr-to-ipam` 示例将 BYOIP CIDR 传输到 IPAM。

(Linux) :

```
aws ec2 move-byoip-cidr-to-ipam \  
  --region us-west-2 \  
  --ipam-pool-id ipam-pool-0a03d430ca3f5c035 \  
  --ipam-pool-owner 111111111111 \  
  --cidr 130.137.249.0/24
```

( 视窗 ) :

```
aws ec2 move-byoip-cidr-to-ipam ^\  
  --region us-west-2 ^\  
  --ipam-pool-id ipam-pool-0a03d430ca3f5c035 ^\  
  --ipam-pool-owner 111111111111 ^\  
  --cidr 130.137.249.0/24
```

输出 :

```
{  
  "ByoipCidr": {  
    "Cidr": "130.137.249.0/24",  
    "State": "pending-transfer"  
  }  
}
```

有关更多信息，请参阅 [Amazon VPC IPAM 用户指南中的教程：将现有 BYOIP IPv4 CIDR 传输到 IPAM](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [MoveByoipCidrToIpam](#) 中的。

## network-insights-access-scope

以下代码示例演示了如何使用 `network-insights-access-scope`。

## AWS CLI

### 创建网络见解访问范围

以下 `create-network-insights-access-scope` 示例在您的 AWS 账户中创建网络见解访问范围。

```
aws ec2 create-network-insights-access-scope \  
  --cli-input-json file://access-scope-file.json
```

`access-scope-file.json` 的内容：

```
{  
  {  
    "MatchPaths": [  
      {  
        "Source": {  
          "ResourceStatement": {  
            "Resources": [  
              "vpc-abcd12e3"  
            ]  
          }  
        }  
      ],  
      "ExcludePaths": [  
        {  
          "Source": {  
            "ResourceStatement": {  
              "ResourceTypes": [  
                "AWS::EC2::InternetGateway"  
              ]  
            }  
          }  
        }  
      ]  
    }  
  }  
}
```

输出：

```
{
```

```

"NetworkInsightsAccessScopeAnalysisId": "nisa-123456789111"
}{
  "NetworkInsightsAccessScope": {
    "NetworkInsightsAccessScopeId": "nis-123456789222",
    "NetworkInsightsAccessScopeArn": "arn:aws:ec2:us-
east-1:123456789222:network-insights-access-scope/nis-123456789222",
    "CreateDate": "2022-01-25T19:20:28.796000+00:00",
    "UpdateDate": "2022-01-25T19:20:28.797000+00:00"
  },
  "NetworkInsightsAccessScopeContent": {
    "NetworkInsightsAccessScopeId": "nis-04c0c0fbca737c404",
    "MatchPaths": [
      {
        "Source": {
          "ResourceStatement": {
            "Resources": [
              "vpc-abcd12e3"
            ]
          }
        }
      }
    ],
    "ExcludePaths": [
      {
        "Source": {
          "ResourceStatement": {
            "ResourceTypes": [
              "AWS::EC2::InternetGateway"
            ]
          }
        }
      }
    ]
  }
}

```

有关更多信息，请参阅《[网络访问分析器指南](#)》中的“[使用 AWS CLI 开始使用网络访问分析器](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [NetworkInsightsAccessScope](#) 中的。

## provision-byoip-cidr

以下代码示例演示了如何使用 `provision-byoip-cidr`。

## AWS CLI

### 配置地址范围

以下 `provision-byoip-cidr` 示例预置了一个公有 IP 地址范围以供使用 AWS。

```
aws ec2 provision-byoip-cidr \  
  --cidr 203.0.113.25/24 \  
  --cidr-authorization-context Message="$text_message",Signature="$signed_message"
```

输出：

```
{  
  "ByoipCidr": {  
    "Cidr": "203.0.113.25/24",  
    "State": "pending-provision"  
  }  
}
```

有关为授权上下文创建消息字符串的更多信息，请参阅 Amazon EC2 用户指南中的 [自带 IP 地址](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ProvisionByoipCidr](#) 中的。

### `provision-ipam-pool-cidr`

以下代码示例演示了如何使用 `provision-ipam-pool-cidr`。

## AWS CLI

### 向 IPAM 池配置 CIDR

以下 `provision-ipam-pool-cidr` 示例为 IPAM 池配置 CIDR。

(Linux)：

```
aws ec2 provision-ipam-pool-cidr \  
  --ipam-pool-id ipam-pool-0533048da7d823723 \  
  --cidr 10.0.0.0/24
```

( 视窗 )：

```
aws ec2 provision-ipam-pool-cidr ^
```

```
--ipam-pool-id ipam-pool-0533048da7d823723 ^  
--cidr 10.0.0.0/24
```

输出：

```
{  
  "IpamPoolCidr": {  
    "Cidr": "10.0.0.0/24",  
    "State": "pending-provision"  
  }  
}
```

有关更多信息，请参阅 Amazon VPC IPAM 用户指南中的向池配置 CID [R](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ProvisionIpamPoolCidr](#) 中的。

## purchase-host-reservation

以下代码示例演示了如何使用 purchase-host-reservation。

### AWS CLI

#### 购买专用主机预留

此示例在您的账户中为指定的专用主机购买指定的专用主机预留服务。

命令：

```
aws ec2 purchase-host-reservation --offering-id hro-03f707bf363b6b324 --host-id-set  
h-013abcd2a00cbd123
```

输出：

```
{  
  "TotalHourlyPrice": "1.499",  
  "Purchase": [  
    {  
      "HourlyPrice": "1.499",  
      "InstanceFamily": "m4",  
      "PaymentOption": "NoUpfront",  
      "HostIdSet": [  
        "h-013abcd2a00cbd123"  
      ]  
    }  
  ]  
}
```

```
    ],  
    "HostReservationId": "hr-0d418a3a4ffc669ae",  
    "UpfrontPrice": "0.000",  
    "Duration": 31536000  
  }  
],  
"TotalUpfrontPrice": "0.000"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PurchaseHostReservation](#)中的。

## **purchase-reserved-instances-offering**

以下代码示例演示了如何使用 `purchase-reserved-instances-offering`。

### AWS CLI

#### 购买预留实例产品

此示例命令说明了购买预留实例产品的情况，并指定了产品 ID 和实例数量。

命令:

```
aws ec2 purchase-reserved-instances-offering --reserved-instances-offering-id  
ec06327e-dd07-46ee-9398-75b5fexample --instance-count 3
```

输出:

```
{  
  "ReservedInstancesId": "af9f760e-6f91-4559-85f7-4980eexample"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PurchaseReservedInstancesOffering](#)中的。

## **purchase-scheduled-instances**

以下代码示例演示了如何使用 `purchase-scheduled-instances`。

### AWS CLI

#### 购买计划实例



此示例购买了计划实例。

命令:

```
aws ec2 purchase-scheduled-instances --purchase-requests file://purchase-request.json
```

购买请求.json :

```
[
  {
    "PurchaseToken": "eyJ2IjoiMSIsInMiOjEsImMiOi...",
    "InstanceCount": 1
  }
]
```

输出 :

```
{
  "ScheduledInstanceSet": [
    {
      "AvailabilityZone": "us-west-2b",
      "ScheduledInstanceId": "sci-1234-1234-1234-1234-123456789012",
      "HourlyPrice": "0.095",
      "CreateDate": "2016-01-25T21:43:38.612Z",
      "Recurrence": {
        "OccurrenceDaySet": [
          1
        ],
        "Interval": 1,
        "Frequency": "Weekly",
        "OccurrenceRelativeToEnd": false,
        "OccurrenceUnit": ""
      },
      "Platform": "Linux/UNIX",
      "TermEndDate": "2017-01-31T09:00:00Z",
      "InstanceCount": 1,
      "SlotDurationInHours": 32,
      "TermStartDate": "2016-01-31T09:00:00Z",
      "NetworkPlatform": "EC2-VPC",
      "TotalScheduledInstanceHours": 1696,
      "NextSlotStartTime": "2016-01-31T09:00:00Z",
    }
  ]
}
```

```
        "InstanceType": "c4.large"  
    }  
]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PurchaseScheduledInstances](#)中的。

## reboot-instances

以下代码示例演示了如何使用 `reboot-instances`。

### AWS CLI

#### 重启 Amazon EC2 实例

本示例将重启指定的实例。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 reboot-instances --instance-ids i-1234567890abcdef5
```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的“重启实例”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RebootInstances](#)中的。

## register-image

以下代码示例演示了如何使用 `register-image`。

### AWS CLI

#### 示例 1：使用清单文件注册 AMI

以下 `register-image` 示例在 Amazon S3 中使用指定的清单文件注册一个 AMI。

```
aws ec2 register-image \  
  --name my-image \  
  --image-location my-s3-bucket/myimage/image.manifest.xml
```

输出：

```
{
  "ImageId": "ami-1234567890EXAMPLE"
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[亚马逊机器映像 \(AMI\)](#)。

#### 示例 2：使用根设备的快照注册 AMI

以下 `register-image` 示例使用 EBS 根卷的指定快照将 AMI 注册为设备 `/dev/xvda`。块储存设备映射还包括一个空的 100 GiB EBS 卷作为设备 `/dev/xvdf`。

```
aws ec2 register-image \
  --name my-image \
  --root-device-name /dev/xvda \
  --block-device-mappings DeviceName=/dev/
xvda,Ebs={SnapshotId=snap-0db2cf683925d191f} DeviceName=/dev/
xvdf,Ebs={VolumeSize=100}
```

输出：

```
{
  "ImageId": "ami-1a2b3c4d5eEXAMPLE"
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[亚马逊机器映像 \(AMI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterImage](#) 中的。

## register-instance-event-notification-attributes

以下代码示例演示了如何使用 `register-instance-event-notification-attributes`。

### AWS CLI

#### 示例 1：在事件通知中包含所有标签

以下 `register-instance-event-notification-attributes` 示例包括事件通知中的所有标签。

```
aws ec2 register-instance-event-notification-attributes \
```

```
--instance-tag-attribute IncludeAllTagsOfInstance=true
```

输出：

```
{
  "InstanceTagAttribute": {
    "InstanceTagKeys": [],
    "IncludeAllTagsOfInstance": true
  }
}
```

有关更多信息，请参阅适用于 [Linux 实例的 Amazon 弹性计算云用户指南中的实例计划事件](#)。

示例 2：在事件通知中包含特定标签

以下 `register-instance-event-notification-attributes` 示例在事件通知中包含指定的标签。如果 `IncludeAllTagsOfInstance` 是，则无法指定标签 `true`。

```
aws ec2 register-instance-event-notification-attributes \
  --instance-tag-attribute InstanceTagKeys="tag-key1","tag-key2"
```

输出：

```
{
  "InstanceTagAttribute": {
    "InstanceTagKeys": [
      "tag-key1",
      "tag-key2"
    ],
    "IncludeAllTagsOfInstance": false
  }
}
```

有关更多信息，请参阅适用于 [Linux 实例的 Amazon 弹性计算云用户指南中的实例计划事件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterInstanceEventNotificationAttributes](#) 中的。

## register-transit-gateway-multicase-group-sources

以下代码示例演示了如何使用 `register-transit-gateway-multicase-group-sources`。

## AWS CLI

向传输网关组播组注册源。

以下`register-transit-gateway-multicast-group-sources`示例将指定的网络接口组源注册到多播组。

```
aws ec2 register-transit-gateway-multicast-group-sources \
  --transit-gateway-multicast-domain-id tgw-mcast-domain-0c4905cef79d6e597 \
  --group-ip-address 224.0.1.0 \
  --network-interface-ids eni-07f290fc3c090cbae
```

输出：

```
{
  "RegisteredMulticastGroupSources": {
    "TransitGatewayMulticastDomainId": "tgw-mcast-domain-0c4905cef79d6e597",
    "RegisteredNetworkInterfaceIds": [
      "eni-07f290fc3c090cbae"
    ],
    "GroupIpAddress": "224.0.1.0"
  }
}
```

有关更多信息，请参阅 T AWS Transit Gateways 用户指南中的向组播组[注册源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterTransitGatewayMulticastGroupSources](#)中的。

## **register-transit-gateway-multicast-group-members**

以下代码示例演示了如何使用 `register-transit-gateway-multicast-group-members`。

### AWS CLI

查看有关传输网关组播域关联的信息

以下`register-transit-gateway-multicast-group-members`示例返回指定多播域的关联。

```
aws ec2 register-transit-gateway-multicast-group-members \
```

```
--transit-gateway-multicast-domain-id tgw-mcast-domain-0c4905cef79d6e597 \  
--group-ip-address 224.0.1.0 \  
--network-interface-ids eni-0e246d32695012e81
```

输出：

```
{  
  "RegisteredMulticastGroupMembers": {  
    "TransitGatewayMulticastDomainId": "tgw-mcast-domain-0c4905cef79d6e597",  
    "RegisteredNetworkInterfaceIds": [  
      "eni-0e246d32695012e81"  
    ],  
    "GroupIpAddress": "224.0.1.0"  
  }  
}
```

有关更多信息，请参阅 Transit Gateways 用户指南中的[管理多播域](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterTransitGatewayMulticastGroupMembers](#) 中的。

## register-transit-gateway-multicast-group-sources

以下代码示例演示了如何使用 register-transit-gateway-multicast-group-sources。

AWS CLI

向传输网关组播组注册源。

以下 register-transit-gateway-multicast-group-sources 示例将指定的网络接口组源注册到多播组。

```
aws ec2 register-transit-gateway-multicast-group-sources \  
  --transit-gateway-multicast-domain-id tgw-mcast-domain-0c4905cef79d6e597 \  
  --group-ip-address 224.0.1.0 \  
  --network-interface-ids eni-07f290fc3c090cbae
```

输出：

```
{  
  "RegisteredMulticastGroupSources": {
```

```

    "TransitGatewayMulticastDomainId": "tgw-mcast-domain-0c4905cef79d6e597",
    "RegisteredNetworkInterfaceIds": [
      "eni-07f290fc3c090cbae"
    ],
    "GroupIpAddress": "224.0.1.0"
  }
}

```

有关更多信息，请参阅《传输网关指南》中的[管理多播域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterTransitGatewayMulticastGroupSources](#)中的。

## reject-transit-gateway-peering-attachment

以下代码示例演示了如何使用 reject-transit-gateway-peering-attachment。

### AWS CLI

拒绝传输网关对等连接

以下reject-transit-gateway-peering-attachment示例拒绝了指定的传输网关对等连接请求。该--region参数指定接受者中转网关所在的区域。

```

aws ec2 reject-transit-gateway-peering-attachment \
  --transit-gateway-attachment-id tgw-attach-4455667788aabbccd \
  --region us-east-2

```

输出：

```

{
  "TransitGatewayPeeringAttachment": {
    "TransitGatewayAttachmentId": "tgw-attach-4455667788aabbccd",
    "RequesterTgwInfo": {
      "TransitGatewayId": "tgw-123abc05e04123abc",
      "OwnerId": "123456789012",
      "Region": "us-west-2"
    },
    "AcceptorTgwInfo": {
      "TransitGatewayId": "tgw-11223344aabbcc112",
      "OwnerId": "123456789012",
      "Region": "us-east-2"
    }
  }
}

```

```
    },
    "State": "rejecting",
    "CreationTime": "2019-12-09T11:50:31.000Z"
  }
}
```

有关更多信息，请参阅《[公交网关指南](#)》中的 [Transit Gateway 对等连接附件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RejectTransitGatewayPeeringAttachment](#) 中的。

## reject-transit-gateway-vpc-attachment

以下代码示例演示了如何使用 reject-transit-gateway-vpc-attachment。

### AWS CLI

拒绝传输网关 VPC 连接

以下 reject-transit-gateway-vpc-attachment 示例拒绝指定的传输网关 VPC 连接。

```
aws ec2 reject-transit-gateway-vpc-attachment \
  --transit-gateway-attachment-id tgw-attach-0a34fe6b4fEXAMPLE
```

输出：

```
{
  "TransitGatewayVpcAttachment": {
    "TransitGatewayAttachmentId": "tgw-attach-0a34fe6b4fEXAMPLE",
    "TransitGatewayId": "tgw-0262a0e521EXAMPLE",
    "VpcId": "vpc-07e8ffd50fEXAMPLE",
    "VpcOwnerId": "111122223333",
    "State": "pending",
    "SubnetIds": [
      "subnet-0752213d59EXAMPLE"
    ],
    "CreationTime": "2019-07-10T17:33:46.000Z",
    "Options": {
      "DnsSupport": "enable",
      "Ipv6Support": "disable"
    }
  }
}
```



```
}
```

有关更多信息，请参阅《[传输网关指南](#)》中的[连接到 VPC](#) 的传输网关。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RejectTransitGatewayVpcAttachment](#)中的。

## reject-transit-gateway-vpc-attachments

以下代码示例演示了如何使用 reject-transit-gateway-vpc-attachments。

### AWS CLI

拒绝传输网关 VPC 连接

以下reject-transit-gateway-vpc-attachment示例拒绝指定的传输网关 VPC 连接。

```
aws ec2 reject-transit-gateway-vpc-attachment \  
  --transit-gateway-attachment-id tgw-attach-0a34fe6b4fEXAMPLE
```

输出：

```
{  
  "TransitGatewayVpcAttachment": {  
    "TransitGatewayAttachmentId": "tgw-attach-0a34fe6b4fEXAMPLE",  
    "TransitGatewayId": "tgw-0262a0e521EXAMPLE",  
    "VpcId": "vpc-07e8ffd50fEXAMPLE",  
    "VpcOwnerId": "111122223333",  
    "State": "pending",  
    "SubnetIds": [  
      "subnet-0752213d59EXAMPLE"  
    ],  
    "CreationTime": "2019-07-10T17:33:46.000Z",  
    "Options": {  
      "DnsSupport": "enable",  
      "Ipv6Support": "disable"  
    }  
  }  
}
```

有关更多信息，请参阅《[传输网关指南](#)》中的[连接到 VPC](#) 的传输网关。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RejectTransitGatewayVpcAttachments](#)中的。

## reject-vpc-endpoint-connections

以下代码示例演示了如何使用 `reject-vpc-endpoint-connections`。

### AWS CLI

拒绝接口终端节点连接请求

此示例拒绝指定终端节点服务的指定端点连接请求。

命令:

```
aws ec2 reject-vpc-endpoint-connections --service-id vpce-svc-03d5ebb7d9579a2b3 --vpc-endpoint-ids vpce-0c1308d7312217abc
```

输出:

```
{
  "Unsuccessful": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RejectVpcEndpointConnections](#)中的。

## reject-vpc-peering-connection

以下代码示例演示了如何使用 `reject-vpc-peering-connection`。

### AWS CLI

拒绝 VPC 对等连接

此示例拒绝指定的 VPC 对等连接请求。

命令:

```
aws ec2 reject-vpc-peering-connection --vpc-peering-connection-id pcx-1a2b3c4d
```

输出:

```
{
```

```
"Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RejectVpcPeeringConnection](#)中的。

## release-address

以下代码示例演示了如何使用 `release-address`。

### AWS CLI

为 EC2-Classical 释放弹性 IP 地址

本示例将释放一个弹性 IP 地址，供 EC2-Classical 中的实例使用。如果命令成功，则不返回任何输出。

命令：

```
aws ec2 release-address --public-ip 198.51.100.0
```

为 EC2-VPC 释放弹性 IP 地址

本示例将释放一个弹性 IP 地址，供 VPC 中的实例使用。如果命令成功，则不返回任何输出。

命令：

```
aws ec2 release-address --allocation-id eipalloc-64d5890a
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReleaseAddress](#)中的。

## release-hosts

以下代码示例演示了如何使用 `release-hosts`。

### AWS CLI

从您的账户中释放专用主机

从您的账户中释放专用主机。必须先停止或终止主机上的实例，然后才能释放主机。

命令:

```
aws ec2 release-hosts --host-id=h-0029d6e3cacf1b3da
```

输出:

```
{
  "Successful": [
    "h-0029d6e3cacf1b3da"
  ],
  "Unsuccessful": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReleaseHosts](#)中的。

## release-ipam-pool-allocation

以下代码示例演示了如何使用 `release-ipam-pool-allocation`。

### AWS CLI

#### 发放 IPAM 池分配

在此示例中，您是 IPAM 委托的管理员，他尝试删除 IPAM 池，但收到错误消息，提示在池有分配时您无法删除该池。您正在使用此命令来释放池分配。

请注意以下几点：

此命令只能用于自定义分配。要在不删除资源的情况下移除资源分配，请使用将其监控状态设置为 `false` [modify-ipam-resource-cidr](#)。要完成此请求，您需要有 IPAM 池 ID，您可以使用它获取 [describe-ipam-pools](#)您还需要分配 ID，您可以通过它获得[get-ipam-pool-allocations](#)。如果您不想逐个删除分配，则可以在删除 IPAM 池之前使用“删除 IPAM 池`--cascade option`时”自动释放池中的所有分配。运行此命令之前有许多先决条件。有关更多信息，请参阅 Amazon VPC IPAM 用户指南`--region`中的[释放分配](#)。运行此命令的位置必须是分配所在的 IPAM 池的语言环境。

以下`release-ipam-pool-allocation`示例发布 IPAM 池分配。

```
aws ec2 release-ipam-pool-allocation \
  --ipam-pool-id ipam-pool-07bdd12d7c94e4693 \
```

```
--cidr 10.0.0.0/23 \  
--ipam-pool-allocation-id ipam-pool-alloc-0e66a1f730da54791b99465b79e7d1e89 \  
--region us-west-1
```

输出：

```
{  
  "Success": true  
}
```

一旦您释放了分配，您可能需要运行[delete-ipam-pool](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReleaseIpamPoolAllocation](#)中的。

## replace-iam-instance-profile-association

以下代码示例演示了如何使用 `replace-iam-instance-profile-association`。

### AWS CLI

替换实例的 IAM 实例配置文件

本示例将关联 `iip-assoc-060bae234aac2e7fa` 表示的 IAM 实例配置文件，替换为名为 `AdminRole` 的 IAM 实例配置文件。

```
aws ec2 replace-iam-instance-profile-association \  
  --iam-instance-profile Name=AdminRole \  
  --association-id iip-assoc-060bae234aac2e7fa
```

输出：

```
{  
  "IamInstanceProfileAssociation": {  
    "InstanceId": "i-087711ddaf98f9489",  
    "State": "associating",  
    "AssociationId": "iip-assoc-0b215292fab192820",  
    "IamInstanceProfile": {  
      "Id": "AIPAJLNLDX3AMYZNWYYAY",  
      "Arn": "arn:aws:iam::123456789012:instance-profile/AdminRole"  
    }  
  }  
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReplaceIamInstanceProfileAssociation](#)中的。

## replace-network-acl-association

以下代码示例演示了如何使用 `replace-network-acl-association`。

### AWS CLI

替换与子网关联的网络 ACL

此示例将指定的网络 ACL 与指定网络 ACL 关联的子网相关联。

命令:

```
aws ec2 replace-network-acl-association --association-id aclassoc-e5b95c8c --  
network-acl-id acl-5fb85d36
```

输出:

```
{  
  "NewAssociationId": "aclassoc-3999875b"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReplaceNetworkAclAssociation](#)中的。

## replace-network-acl-entry

以下代码示例演示了如何使用 `replace-network-acl-entry`。

### AWS CLI

替换网络 ACL 条目

此示例替换了指定网络 ACL 的条目。新规则 100 允许通过 UDP 端口 53 (DNS) 从 203.0.113.12/24 进入任何关联子网的入口流量。

命令:

```
aws ec2 replace-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-number 100 --protocol udp --port-range From=53,To=53 --cidr-block 203.0.113.12/24 --rule-action allow
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReplaceNetworkAclEntry](#)中的。

## replace-route-table-association

以下代码示例演示了如何使用 `replace-route-table-association`。

### AWS CLI

替换与子网关联的路由表

此示例将指定的路由表与指定路由表关联的子网相关联。

命令:

```
aws ec2 replace-route-table-association --association-id rtbassoc-781d0d1a --route-table-id rtb-22574640
```

输出 :

```
{
  "NewAssociationId": "rtbassoc-3a1f0f58"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReplaceRouteTableAssociation](#)中的。

## replace-route

以下代码示例演示了如何使用 `replace-route`。

### AWS CLI

替换路线

此示例替换了指定路由表中的指定路由。新路由匹配指定的 CIDR，并将流量发送到指定的虚拟专用网关。如果命令成功，则不返回任何输出。

**命令:**

```
aws ec2 replace-route --route-table-id rtb-22574640 --destination-cidr-block 10.0.0.0/16 --gateway-id vgw-9a4cacf3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReplaceRoute](#)中的。

**replace-transit-gateway-route**

以下代码示例演示了如何使用 `replace-transit-gateway-route`。

## AWS CLI

替换指定公交网关路由表中的指定路由

以下`replace-transit-gateway-route`示例替换了指定公交网关路由表中的路由。

```
aws ec2 replace-transit-gateway-route \
  --destination-cidr-block 10.0.2.0/24 \
  --transit-gateway-attachment-id tgw-attach-09b52ccdb5EXAMPLE \
  --transit-gateway-route-table-id tgw-rtb-0a823edbdeEXAMPLE
```

**输出:**

```
{
  "Route": {
    "DestinationCidrBlock": "10.0.2.0/24",
    "TransitGatewayAttachments": [
      {
        "ResourceId": "vpc-4EXAMPLE",
        "TransitGatewayAttachmentId": "tgw-attach-09b52ccdb5EXAMPLE",
        "ResourceType": "vpc"
      }
    ],
    "Type": "static",
    "State": "active"
  }
}
```

有关更多信息，请参阅《[公交网关指南](#)》中的 [Transit Gateway 路由表](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReplaceTransitGatewayRoute](#)中的。

## report-instance-status

以下代码示例演示了如何使用 `report-instance-status`。

### AWS CLI

报告实例的状态反馈

此示例命令报告指定实例的状态反馈。

命令:

```
aws ec2 report-instance-status --instances i-1234567890abcdef0 --status impaired --reason-codes unresponsive
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReportInstanceStatus](#)中的。

## request-spot-fleet

以下代码示例演示了如何使用 `request-spot-fleet`。

### AWS CLI

请求子网中价格最低的 Spot 队列

此示例命令创建一个 Spot 队列请求，其中包含两个仅因子网而异的启动规格。竞价型队列以最低价格启动指定子网中的实例。如果实例在默认 VPC 中启动，则默认情况下它们会收到一个公有 IP 地址。如果在非默认 VPC 中启动实例，则默认情况下它们不会收到一个公有 IP 地址。

请注意，您不能在竞价型队列请求中指定来自同一可用区的不同子网。

命令:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

config.json :

```
{
```

```
"SpotPrice": "0.04",
"TargetCapacity": 2,
"IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
"LaunchSpecifications": [
  {
    "ImageId": "ami-1a2b3c4d",
    "KeyName": "my-key-pair",
    "SecurityGroups": [
      {
        "GroupId": "sg-1a2b3c4d"
      }
    ],
    "InstanceType": "m3.medium",
    "SubnetId": "subnet-1a2b3c4d, subnet-3c4d5e6f",
    "IamInstanceProfile": {
      "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
    }
  }
]
}
```

输出：

```
{
  "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
}
```

请求可用区内价格最低的 Spot 队列

此示例命令创建具有两个启动规格的 Spot 队列请求，这两个启动规格仅因可用区而异。竞价型队列以最低价格在指定可用区启动实例。如果您的账户仅支持 EC2-VPC，则 Amazon EC2 会在可用区的默认子网中启动竞价型实例。如果您的账户支持 EC2-Classic，则 Amazon EC2 会在可用区启动 EC2-Classic 中的实例。

命令：

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

config.json：

```
{
```

```

"SpotPrice": "0.04",
"TargetCapacity": 2,
"IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
"LaunchSpecifications": [
  {
    "ImageId": "ami-1a2b3c4d",
    "KeyName": "my-key-pair",
    "SecurityGroups": [
      {
        "GroupId": "sg-1a2b3c4d"
      }
    ],
    "InstanceType": "m3.medium",
    "Placement": {
      "AvailabilityZone": "us-west-2a, us-west-2b"
    },
    "IamInstanceProfile": {
      "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
    }
  }
]
}

```

在子网中启动竞价型实例并为其分配公有 IP 地址

此示例命令为在非默认 VPC 中启动的实例分配公有地址。请注意，在指定网络接口时，必须使用该网络接口包括子网 ID 和安全组 ID。

命令:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

config.json :

```

{
  "SpotPrice": "0.04",
  "TargetCapacity": 2,
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
  "LaunchSpecifications": [
    {
      "ImageId": "ami-1a2b3c4d",
      "KeyName": "my-key-pair",

```

```

    "InstanceType": "m3.medium",
    "NetworkInterfaces": [
      {
        "DeviceIndex": 0,
        "SubnetId": "subnet-1a2b3c4d",
        "Groups": [ "sg-1a2b3c4d" ],
        "AssociatePublicIpAddress": true
      }
    ],
    "IamInstanceProfile": {
      "Arn": "arn:aws:iam::880185128111:instance-profile/my-iam-role"
    }
  }
]
}

```

### 使用多元化分配策略申请 Spot 队列

此示例命令创建一个 Spot 队列请求，该请求使用多元化分配策略启动 30 个实例。启动规格因实例类型而异。竞价型队列按启动规格分配实例，因此每种类型有 10 个实例。

命令:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

config.json :

```

{
  "SpotPrice": "0.70",
  "TargetCapacity": 30,
  "AllocationStrategy": "diversified",
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
  "LaunchSpecifications": [
    {
      "ImageId": "ami-1a2b3c4d",
      "InstanceType": "c4.2xlarge",
      "SubnetId": "subnet-1a2b3c4d"
    },
    {
      "ImageId": "ami-1a2b3c4d",
      "InstanceType": "m3.2xlarge",
      "SubnetId": "subnet-1a2b3c4d"
    }
  ],
}

```

```
{
  "ImageId": "ami-1a2b3c4d",
  "InstanceType": "r3.2xlarge",
  "SubnetId": "subnet-1a2b3c4d"
}
]
```

有关更多信息，请参阅 Amazon 弹性计算云用户指南中的竞价型队列请求。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RequestSpotFleet](#) 中的。

## request-spot-instances

以下代码示例演示了如何使用 request-spot-instances。

### AWS CLI

#### 请求竞价型实例

此示例命令为指定可用区中的五个实例创建一次性竞价型实例请求。如果您的账户仅支持 EC2-VPC，则 Amazon EC2 会在指定可用区的默认子网中启动实例。如果您的账户支持 EC2-Classic，则 Amazon EC2 会在指定的可用区域中启动 EC2-Classic 中的实例。

#### 命令：

```
aws ec2 request-spot-instances --spot-price "0.03" --instance-count 5 --type "one-time" --launch-specification file://specification.json
```

#### 规格.json：

```
{
  "ImageId": "ami-1a2b3c4d",
  "KeyName": "my-key-pair",
  "SecurityGroupIds": [ "sg-1a2b3c4d" ],
  "InstanceType": "m3.medium",
  "Placement": {
    "AvailabilityZone": "us-west-2a"
  },
  "IamInstanceProfile": {
    "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
  }
}
```

```
}
```

输出：

```
{
  "SpotInstanceRequests": [
    {
      "Status": {
        "UpdateTime": "2014-03-25T20:54:21.000Z",
        "Code": "pending-evaluation",
        "Message": "Your Spot request has been submitted for review, and is
pending evaluation."
      },
      "ProductDescription": "Linux/UNIX",
      "SpotInstanceRequestId": "sir-df6f405d",
      "State": "open",
      "LaunchSpecification": {
        "Placement": {
          "AvailabilityZone": "us-west-2a"
        },
        "ImageId": "ami-1a2b3c4d",
        "KeyName": "my-key-pair",
        "SecurityGroups": [
          {
            "GroupName": "my-security-group",
            "GroupId": "sg-1a2b3c4d"
          }
        ],
        "Monitoring": {
          "Enabled": false
        },
        "IamInstanceProfile": {
          "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
        },
        "InstanceType": "m3.medium"
      },
      "Type": "one-time",
      "CreateTime": "2014-03-25T20:54:20.000Z",
      "SpotPrice": "0.050000"
    },
    ...
  ]
}
```

此示例命令为指定子网中的五个实例创建一次性竞价型实例请求。Amazon EC2 会在指定子网中启动实例。如果 VPC 是非默认 VPC，则默认情况下实例不会收到公有 IP 地址。

命令:

```
aws ec2 request-spot-instances --spot-price "0.050" --instance-count 5 --type "one-time" --launch-specification file://specification.json
```

规格.json :

```
{
  "ImageId": "ami-1a2b3c4d",
  "SecurityGroupIds": [ "sg-1a2b3c4d" ],
  "InstanceType": "m3.medium",
  "SubnetId": "subnet-1a2b3c4d",
  "IamInstanceProfile": {
    "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
  }
}
```

输出 :

```
{
  "SpotInstanceRequests": [
    {
      "Status": {
        "UpdateTime": "2014-03-25T22:21:58.000Z",
        "Code": "pending-evaluation",
        "Message": "Your Spot request has been submitted for review, and is pending evaluation."
      },
      "ProductDescription": "Linux/UNIX",
      "SpotInstanceRequestId": "sir-df6f405d",
      "State": "open",
      "LaunchSpecification": {
        "Placement": {
          "AvailabilityZone": "us-west-2a"
        }
      },
      "ImageId": "ami-1a2b3c4d",
      "SecurityGroups": [
        {
          "GroupName": "my-security-group",
```

```

        "GroupID": "sg-1a2b3c4d"
      }
    ]
    "SubnetId": "subnet-1a2b3c4d",
    "Monitoring": {
      "Enabled": false
    },
    "IamInstanceProfile": {
      "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
    },
    "InstanceType": "m3.medium",
  },
  "Type": "one-time",
  "CreateTime": "2014-03-25T22:21:58.000Z",
  "SpotPrice": "0.050000"
},
...
]
}

```

此示例为您在非默认 VPC 中启动的竞价型实例分配公有 IP 地址。请注意，在指定网络接口时，必须使用该网络接口包括子网 ID 和安全组 ID。

#### 命令:

```
aws ec2 request-spot-instances --spot-price "0.050" --instance-count 1 --type "one-time" --launch-specification file://specification.json
```

#### 规格.json :

```

{
  "ImageId": "ami-1a2b3c4d",
  "KeyName": "my-key-pair",
  "InstanceType": "m3.medium",
  "NetworkInterfaces": [
    {
      "DeviceIndex": 0,
      "SubnetId": "subnet-1a2b3c4d",
      "Groups": [ "sg-1a2b3c4d" ],
      "AssociatePublicIpAddress": true
    }
  ],

```



```
"IamInstanceProfile": {
  "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RequestSpotInstances](#)中的。

## reset-address-attribute

以下代码示例演示了如何使用 `reset-address-attribute`。

### AWS CLI

重置与弹性 IP 地址关联的域名属性

以下`reset-address-attribute`示例重置弹性 IP 地址的域名属性。

Linux :

```
aws ec2 reset-address-attribute \
  --allocation-id eipalloc-abcdef01234567890 \
  --attribute domain-name
```

Windows:

```
aws ec2 reset-address-attribute ^
  --allocation-id eipalloc-abcdef01234567890 ^
  --attribute domain-name
```

输出 :

```
{
  "Addresses": [
    {
      "PublicIp": "192.0.2.0",
      "AllocationId": "eipalloc-abcdef01234567890",
      "PtrRecord": "example.com."
      "PtrRecordUpdate": {
        "Value": "example.net.",
        "Status": "PENDING"
      }
    }
  ]
}
```

```
]
}
```

要监控待处理的更改，请参阅《AWS CLI 命令参考》[describe-addresses-attribute](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ResetAddressAttribute](#)中的。

## reset-ebs-default-kms-key-id

以下代码示例演示了如何使用 `reset-ebs-default-kms-key-id`。

### AWS CLI

#### 重置 EBS 加密的默认 CMK

以下`reset-ebs-default-kms-key-id`示例为您在当前区域的 AWS 账户重置 EBS 加密的默认 CMK。

```
aws ec2 reset-ebs-default-kms-key-id
```

输出：

```
{
  "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/8c5b2c63-b9bc-45a3-
a87a-5513eEXAMPLE"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ResetEbsDefaultKmsKeyId](#)中的。

## reset-fpga-image-attribute

以下代码示例演示了如何使用 `reset-fpga-image-attribute`。

### AWS CLI

#### 重置 Amazon FPGA 镜像的属性

此示例重置指定 AFI 的加载权限。

命令：

```
aws ec2 reset-fpga-image-attribute --fpga-image-id afi-0d123e123bfc85abc --attribute
loadPermission
```

输出：

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ResetFpgaImageAttribute](#)中的。

## reset-image-attribute

以下代码示例演示了如何使用 `reset-image-attribute`。

AWS CLI

重置 `launchPermission` 属性

此示例将指定 AMI 的 `launchPermission` 属性重置为其默认值。默认情况下，AMI 是私有的。如果命令成功，则不返回任何输出。

命令：

```
aws ec2 reset-image-attribute --image-id ami-5731123e --attribute launchPermission
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ResetImageAttribute](#)中的。

## reset-instance-attribute

以下代码示例演示了如何使用 `reset-instance-attribute`。

AWS CLI

重置 `sourceDestCheck` 属性

此示例重置了指定实例的 `sourceDestCheck` 属性。该实例必须位于 VPC 中。如果命令成功，则不返回任何输出。

命令：

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --attribute
sourceDestCheck
```

### 重置内核属性

此示例重置了指定实例的kernel属性。该实例必须处于 stopped 状态。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --attribute
kernel
```

### 重置 ramdisk 属性

此示例重置了指定实例的ramdisk属性。该实例必须处于 stopped 状态。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --attribute
ramdisk
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ResetInstanceAttribute](#)中的。

## reset-network-interface-attribute

以下代码示例演示了如何使用 reset-network-interface-attribute。

### AWS CLI

#### 重置网络接口属性

以下reset-network-interface-attribute示例将源/目标检查属性的值重置为 true

```
aws ec2 reset-network-interface-attribute \
  --network-interface-id eni-686ea200 \
  --source-dest-check
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ResetNetworkInterfaceAttribute](#)中的。

## reset-snapshot-attribute

以下代码示例演示了如何使用 `reset-snapshot-attribute`。

### AWS CLI

#### 重置快照属性

此示例重置了快照 `snap-1234567890abcdef0` 的创建卷权限。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 reset-snapshot-attribute --snapshot-id snap-1234567890abcdef0 --attribute createVolumePermission
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ResetSnapshotAttribute](#)中的。

## restore-address-to-classic

以下代码示例演示了如何使用 `restore-address-to-classic`。

### AWS CLI

#### 将地址恢复到 EC2-Classic

此示例将弹性 IP 地址 `198.51.100.0` 恢复到 EC2-Classic 平台。

命令:

```
aws ec2 restore-address-to-classic --public-ip 198.51.100.0
```

输出:

```
{
  "Status": "MoveInProgress",
  "PublicIp": "198.51.100.0"
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RestoreAddressToClassic](#)中的。

## restore-image-from-recycle-bin

以下代码示例演示了如何使用 `restore-image-from-recycle-bin`。

### AWS CLI

从回收站恢复图像

以下`restore-image-from-recycle-bin`示例从回收站中恢复 AMI `ami-0111222333444abcd`。

```
aws ec2 restore-image-from-recycle-bin \  
  --image-id ami-0111222333444abcd
```

输出：

```
{  
  "Return": true  
}
```

有关更多信息，请参阅《[亚马逊弹性计算云用户指南](#)》中的[从回收站中恢复 AM I](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RestoreImageFromRecycleBin](#)中的。

## restore-managed-prefix-list-version

以下代码示例演示了如何使用 `restore-managed-prefix-list-version`。

### AWS CLI

us-west-2\*\*用于恢复前缀列表版本\*\*

以下内容`restore-managed-prefix-list-version`恢复了指定前缀列表的版本 1 中的条目。

```
aws ec2 restore-managed-prefix-list-version \  
  --prefix-list-id pl-0123456abcabc1 \  
  --version 1
```

```
--current-version 2 \  
--previous-version 1
```

输出：

```
{  
  "PrefixList": {  
    "PrefixListId": "pl-0123456abcabcabc1",  
    "AddressFamily": "IPv4",  
    "State": "restore-in-progress",  
    "PrefixListArn": "arn:aws:ec2:us-west-2:123456789012:prefix-list/  
pl-0123456abcabcabc1",  
    "PrefixListName": "vpc-cidrs",  
    "MaxEntries": 10,  
    "Version": 2,  
    "OwnerId": "123456789012"  
  }  
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的[托管前缀列表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RestoreManagedPrefixListVersion](#)中的。

## restore-snapshot-from-recycle-bin

以下代码示例演示了如何使用 `restore-snapshot-from-recycle-bin`。

### AWS CLI

#### 从回收站恢复快照

以下 `restore-snapshot-from-recycle-bin` 示例从回收站恢复快照。从回收站还原快照时，该快照立即可供使用，回收站会将其删除。还原快照后，您可以像使用账户中任何其它快照一样使用它。

```
aws ec2 restore-snapshot-from-recycle-bin \  
--snapshot-id snap-01234567890abcdef
```

此命令不生成任何输出。

有关 Amazon EBS 回收站的更多信息，请参阅 Amazon EC2 用户指南中的[从回收站恢复快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RestoreSnapshotFromRecycleBin](#)中的。

## restore-snapshot-tier

以下代码示例演示了如何使用 `restore-snapshot-tier`。

### AWS CLI

#### 示例 1：永久恢复已存档的快照

以下`restore-snapshot-tier`示例永久恢复指定的快照。指定`--snapshot-id`并包括`permanent-restore`选项。

```
aws ec2 restore-snapshot-tier \  
  --snapshot-id snap-01234567890abcdef \  
  --permanent-restore
```

输出：

```
{  
  "SnapshotId": "snap-01234567890abcdef",  
  "IsPermanentRestore": true  
}
```

有关快照存档的更多信息，请参阅亚马逊 EC2 用户指南中的存档亚马逊 EBS 快照 < [https://docs.aws.amazon.com/AWS\\_ec2/latest/UserGuide/snapshot-archive.html](https://docs.aws.amazon.com/AWS_ec2/latest/UserGuide/snapshot-archive.html) >。

#### 示例 2：临时恢复已存档的快照

以下`restore-snapshot-tier`示例暂时恢复指定的快照。忽略 `--permanent-restore` 选项。指定`--snapshot-id`，对于`temporary-restore-days`，指定恢复快照的天数。`temporary-restore-days`必须以天为单位指定。允许的范围为1到180。如果没有指定一个值，默认为 1 天。

```
aws ec2 restore-snapshot-tier \  
  --snapshot-id snap-01234567890abcdef \  
  --temporary-restore-days 5
```

输出：



```
{
  "SnapshotId": "snap-01234567890abcdef",
  "RestoreDuration": 5,
  "IsPermanentRestore": false
}
```

有关快照存档的更多信息，请参阅亚马逊 EC2 用户指南中的存档亚马逊 EBS 快照 < <https://docs.aws.amazon.com/AWS ec2/latest/ UserGuide /snapshot-archive.html> >。

### 示例 3：修改还原周期

以下 `restore-snapshot-tier` 示例将指定快照的还原周期更改为 10 天。

```
aws ec2 restore-snapshot-tier \
  --snapshot-id snap-01234567890abcdef
  --temporary-restore-days 10
```

输出：

```
{
  "SnapshotId": "snap-01234567890abcdef",
  "RestoreDuration": 10,
  "IsPermanentRestore": false
}
```

有关快照存档的更多信息，请参阅亚马逊 EC2 用户指南中的存档亚马逊 EBS 快照 < <https://docs.aws.amazon.com/AWS ec2/latest/ UserGuide /snapshot-archive.html> >。

### 示例 4：修改还原类型

以下 `restore-snapshot-tier` 示例将指定快照的还原类型从临时更改为永久。

```
aws ec2 restore-snapshot-tier \
  --snapshot-id snap-01234567890abcdef
  --permanent-restore
```

输出：

```
{
  "SnapshotId": "snap-01234567890abcdef",
```

```
"IsPermanentRestore": true
}
```

有关快照存档的更多信息，请参阅亚马逊 EC2 用户指南中的存档亚马逊 EBS 快照 < [https://docs.aws.amazon.com/AWS\\_ec2/latest/UserGuide/snapshot-archive.html](https://docs.aws.amazon.com/AWS_ec2/latest/UserGuide/snapshot-archive.html) >。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RestoreSnapshotTier](#)中的。

## revoke-client-vpn-ingress

以下代码示例演示了如何使用 `revoke-client-vpn-ingress`。

### AWS CLI

撤销 Client VPN 终端节点的授权规则

以下`revoke-client-vpn-ingress`示例撤消了所有群组的互联网访问规则 (`0.0.0.0/0`)。

```
aws ec2 revoke-client-vpn-ingress \
  --client-vpn-endpoint-id cvpn-endpoint-123456789123abcde \
  --target-network-cidr 0.0.0.0/0 --revoke-all-groups
```

输出：

```
{
  "Status": {
    "Code": "revoking"
  }
}
```

有关更多信息，请参阅《AWS Client VPN 管理员指南》中的[授权规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RevokeClientVpnIngress](#)中的。

## revoke-security-group-egress

以下代码示例演示了如何使用 `revoke-security-group-egress`。

### AWS CLI

示例 1：删除允许出站流量到达特定地址范围的规则

以下`revoke-security-group-egress`示例命令删除了授予对 TCP 端口 80 上指定地址范围的访问权限的规则。

```
aws ec2 revoke-security-group-egress \  
  --group-id sg-026c12253ce15eff7 \  
  --ip-permissions  
  [{IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges=[{CidrIp=10.0.0.0/16}]}
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 用户指南中的[安全组](#)。

示例 2：删除允许向特定安全组发送出站流量的规则

以下`revoke-security-group-egress`示例命令删除了在 TCP 端口 80 上授予对指定安全组的访问权限的规则。

```
aws ec2 revoke-security-group-egress \  
  --group-id sg-026c12253ce15eff7 \  
  --ip-permissions '[{"IpProtocol": "tcp", "FromPort": 443, "ToPort":  
  443, "UserIdGroupPairs": [{"GroupId": "sg-06df23a01ff2df86d"}]}'
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 用户指南中的[安全组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RevokeSecurityGroupEgress](#)中的。

## **revoke-security-group-ingress**

以下代码示例演示了如何使用 `revoke-security-group-ingress`。

### AWS CLI

示例 1：从安全组中移除规则

以下`revoke-security-group-ingress`示例从默认 VPC 的指定安全组中删除 `203.0.113.0/24` 地址范围的 TCP 端口 22 访问权限。

```
aws ec2 revoke-security-group-ingress \  
  --group-name mySecurityGroup
```

```
--protocol tcp \  
--port 22 \  
--cidr 203.0.113.0/24
```

如果成功执行此命令，则不会产生任何输出。

有关更多信息，请参阅 Amazon EC2 用户指南中的[安全组](#)。

#### 示例 2：使用 IP 权限集删除规则

以下 `revoke-security-group-ingress` 示例使用 `ip-permissions` 参数删除允许 ICMP 消息的入站规则 `Destination Unreachable: Fragmentation Needed and Don't Fragment was Set` (类型 3, 代码 4)。

```
aws ec2 revoke-security-group-ingress \  
  --group-id sg-026c12253ce15eff7 \  
  --ip-permissions  
  IpProtocol=icmp,FromPort=3,ToPort=4,IpRanges=[{CidrIp=0.0.0.0/0}]
```

如果成功执行此命令，则不会产生任何输出。

有关更多信息，请参阅 Amazon EC2 用户指南中的[安全组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RevokeSecurityGroupIngress](#) 中的。

## run-instances

以下代码示例演示了如何使用 `run-instances`。

### AWS CLI

#### 示例 1：将实例启动到默认子网

以下 `run-instances` 示例将类型 `t2.micro` 的单个实例启动到当前区域的默认子网中，并将其与该区域默认 VPC 的默认子网相关联。如果不打算使用 SSH (Linux) 或 RDP (Windows) 连接到实例，则密钥对是可选的。

```
aws ec2 run-instances \  
  --image-id ami-0abcdef1234567890 \  
  --instance-type t2.micro \  
  --key-name MyKeyPair
```

输出：

```
{
  "Instances": [
    {
      "AmiLaunchIndex": 0,
      "ImageId": "ami-0abcdef1234567890",
      "InstanceId": "i-1231231230abcdef0",
      "InstanceType": "t2.micro",
      "KeyName": "MyKeyPair",
      "LaunchTime": "2018-05-10T08:05:20.000Z",
      "Monitoring": {
        "State": "disabled"
      },
      "Placement": {
        "AvailabilityZone": "us-east-2a",
        "GroupName": "",
        "Tenancy": "default"
      },
      "PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",
      "PrivateIpAddress": "10.0.0.157",
      "ProductCodes": [],
      "PublicDnsName": "",
      "State": {
        "Code": 0,
        "Name": "pending"
      },
      "StateTransitionReason": "",
      "SubnetId": "subnet-04a636d18e83cfac",
      "VpcId": "vpc-1234567890abcdef0",
      "Architecture": "x86_64",
      "BlockDeviceMappings": [],
      "ClientToken": "",
      "EbsOptimized": false,
      "Hypervisor": "xen",
      "NetworkInterfaces": [
        {
          "Attachment": {
            "AttachTime": "2018-05-10T08:05:20.000Z",
            "AttachmentId": "eni-attach-0e325c07e928a0405",
            "DeleteOnTermination": true,
            "DeviceIndex": 0,
            "Status": "attaching"
          },

```

```
    "Description": "",
    "Groups": [
      {
        "GroupName": "MySecurityGroup",
        "GroupId": "sg-0598c7d356eba48d7"
      }
    ],
    "Ipv6Addresses": [],
    "MacAddress": "0a:ab:58:e0:67:e2",
    "NetworkInterfaceId": "eni-0c0a29997760baee7",
    "OwnerId": "123456789012",
    "PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",
    "PrivateIpAddress": "10.0.0.157",
    "PrivateIpAddresses": [
      {
        "Primary": true,
        "PrivateDnsName": "ip-10-0-0-157.us-
east-2.compute.internal",
        "PrivateIpAddress": "10.0.0.157"
      }
    ],
    "SourceDestCheck": true,
    "Status": "in-use",
    "SubnetId": "subnet-04a636d18e83cfacb",
    "VpcId": "vpc-1234567890abcdef0",
    "InterfaceType": "interface"
  }
],
"RootDeviceName": "/dev/xvda",
"RootDeviceType": "ebs",
"SecurityGroups": [
  {
    "GroupName": "MySecurityGroup",
    "GroupId": "sg-0598c7d356eba48d7"
  }
],
"SourceDestCheck": true,
"StateReason": {
  "Code": "pending",
  "Message": "pending"
},
"Tags": [],
"VirtualizationType": "hvm",
"CpuOptions": {
```

```
        "CoreCount": 1,
        "ThreadsPerCore": 1
    },
    "CapacityReservationSpecification": {
        "CapacityReservationPreference": "open"
    },
    "MetadataOptions": {
        "State": "pending",
        "HttpTokens": "optional",
        "HttpPutResponseHopLimit": 1,
        "HttpEndpoint": "enabled"
    }
}
],
"OwnerId": "123456789012",
"ReservationId": "r-02a3f596d91211712"
}
```

示例 2：将实例启动到非默认子网，并添加一个公有 IP 地址

以下 `run-instances` 示例为要启动到非默认子网的实例请求一个公有 IP 地址。实例与指定的安全组相关联。

```
aws ec2 run-instances \
  --image-id ami-0abcdef1234567890 \
  --instance-type t2.micro \
  --subnet-id subnet-08fc749671b2d077c \
  --security-group-ids sg-0b0384b66d7d692f9 \
  --associate-public-ip-address \
  --key-name MyKeyPair
```

有关 `run-instances` 的输出示例，请参阅示例 1。

示例 3：启动具有附加卷的实例

以下 `run-instances` 示例使用 `mapping.json` 中指定的块设备映射，在启动时对附加卷进行附加。块设备映射可以指定 EBS 卷、实例存储卷，也可以同时指定 EBS 卷和实例存储卷。

```
aws ec2 run-instances \
  --image-id ami-0abcdef1234567890 \
  --instance-type t2.micro \
  --subnet-id subnet-08fc749671b2d077c \
  --security-group-ids sg-0b0384b66d7d692f9 \
```

```
--key-name MyKeyPair \  
--block-device-mappings file://mapping.json
```

mapping.json 的内容。本示例添加了 /dev/sdh，这是一个大小为 100GiB 的空 EBS 卷。

```
[  
  {  
    "DeviceName": "/dev/sdh",  
    "Ebs": {  
      "VolumeSize": 100  
    }  
  }  
]
```

mapping.json 的内容。本示例添加了 ephemeral1，作为实例存储卷。

```
[  
  {  
    "DeviceName": "/dev/sdc",  
    "VirtualName": "ephemeral1"  
  }  
]
```

有关 run-instances 的输出示例，请参阅示例 1。

有关块设备映射的更多信息，请参阅《Amazon EC2 用户指南》中的[块设备映射](#)。

示例 4：启动实例并在创建时添加标签

以下 run-instances 示例向实例中添加了一个键为 webserver、值为 production 的标签。该命令还向创建的任何 EBS 卷 (此示例中为根卷) 应用键为 cost-center、值为 cc123 的标签。

```
aws ec2 run-instances \  
  --image-id ami-0abcdef1234567890 \  
  --instance-type t2.micro \  
  --count 1 \  
  --subnet-id subnet-08fc749671b2d077c \  
  --key-name MyKeyPair \  
  --security-group-ids sg-0b0384b66d7d692f9 \  
  --tag-specifications  
'ResourceType=instance,Tags=[{Key=webserver,Value=production}]'  
'ResourceType=volume,Tags=[{Key=cost-center,Value=cc123}]'
```



有关 `run-instances` 的输出示例，请参阅示例 1。

#### 示例 5：启动包含用户数据的实例

以下 `run-instances` 示例在名为 `my_script.txt` 的文件中传递用户数据，该文件包含实例的配置脚本。该脚本在启动时运行。

```
aws ec2 run-instances \  
  --image-id ami-0abcdef1234567890 \  
  --instance-type t2.micro \  
  --count 1 \  
  --subnet-id subnet-08fc749671b2d077c \  
  --key-name MyKeyPair \  
  --security-group-ids sg-0b0384b66d7d692f9 \  
  --user-data file://my_script.txt
```

有关 `run-instances` 的输出示例，请参阅示例 1。

有关实例用户数据的更多信息，请参阅《Amazon EC2 用户指南》中的[使用实例用户数据](#)。

#### 示例 6：启动可突增性能实例

以下 `run-instances` 示例启动带有 `unlimited` 服务抵扣金选项的 `t2.micro` 实例。当启动 T2 实例时，如果未指定 `--credit-specification`，则默认为 `standard` 服务抵扣金选项。当启动 T3 实例时，默认为 `unlimited` 服务抵扣金选项。

```
aws ec2 run-instances \  
  --image-id ami-0abcdef1234567890 \  
  --instance-type t2.micro \  
  --count 1 \  
  --subnet-id subnet-08fc749671b2d077c \  
  --key-name MyKeyPair \  
  --security-group-ids sg-0b0384b66d7d692f9 \  
  --credit-specification CpuCredits=unlimited
```

有关 `run-instances` 的输出示例，请参阅示例 1。

有关可突增性能实例的更多信息，请参阅《Amazon EC2 用户指南》中的[可突增性能实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RunInstances](#)中的。

## run-scheduled-instances

以下代码示例演示了如何使用 run-scheduled-instances。

### AWS CLI

#### 启动计划实例

此示例在 VPC 中启动指定的计划实例。

命令:

```
aws ec2 run-scheduled-instances --scheduled-instance-id
sci-1234-1234-1234-1234-123456789012 --instance-count 1 --launch-specification
file://launch-specification.json
```

启动规范.json :

```
{
  "ImageId": "ami-12345678",
  "KeyName": "my-key-pair",
  "InstanceType": "c4.large",
  "NetworkInterfaces": [
    {
      "DeviceIndex": 0,
      "SubnetId": "subnet-12345678",
      "AssociatePublicIpAddress": true,
      "Groups": ["sg-12345678"]
    }
  ],
  "IamInstanceProfile": {
    "Name": "my-iam-role"
  }
}
```

输出 :

```
{
  "InstanceIdSet": [
    "i-1234567890abcdef0"
  ]
}
```

此示例在 EC2-Classic 中启动指定的计划实例。

命令:

```
aws ec2 run-scheduled-instances --scheduled-instance-id
sci-1234-1234-1234-1234-123456789012 --instance-count 1 --launch-specification
file://launch-specification.json
```

启动规范.json :

```
{
  "ImageId": "ami-12345678",
  "KeyName": "my-key-pair",
  "SecurityGroupIds": ["sg-12345678"],
  "InstanceType": "c4.large",
  "Placement": {
    "AvailabilityZone": "us-west-2b"
  }
  "IamInstanceProfile": {
    "Name": "my-iam-role"
  }
}
```

输出 :

```
{
  "InstanceIdSet": [
    "i-1234567890abcdef0"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RunScheduledInstances](#)中的。

## search-local-gateway-routes

以下代码示例演示了如何使用 search-local-gateway-routes。

### AWS CLI

在本地网关路由表中搜索路由

以下search-local-gateway-routes示例在指定的本地网关路由表中搜索静态路由。

```
aws ec2 search-local-gateway-routes \  
  --local-gateway-route-table-id lgw-rtb-059615ef7dEXAMPLE \  
  --filters "Name=type,Values=static"
```

输出：

```
{  
  "Route": {  
    "DestinationCidrBlock": "0.0.0.0/0",  
    "LocalGatewayVirtualInterfaceGroupId": "lgw-vif-grp-07145b276bEXAMPLE",  
    "Type": "static",  
    "State": "deleted",  
    "LocalGatewayRouteTableId": "lgw-rtb-059615ef7EXAMPLE"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SearchLocalGatewayRoutes](#)中的。

## search-transit-gateway-multicast-groups

以下代码示例演示了如何使用 search-transit-gateway-multicast-groups。

### AWS CLI

搜索一个或多个公交网关组播组并返回组成员资格信息

以下search-transit-gateway-multicast-groups示例返回指定多播组的组成员资格。

```
aws ec2 search-transit-gateway-multicast-groups \  
  --transit-gateway-multicast-domain-id tgw-mcast-domain-000fb24d04EXAMPLE
```

输出：

```
{  
  "MulticastGroups": [  
    {  
      "GroupIpAddress": "224.0.1.0",  
      "TransitGatewayAttachmentId": "tgw-attach-0372e72386EXAMPLE",  
      "SubnetId": "subnet-0187aff814EXAMPLE",  
      "ResourceId": "vpc-0065acced4EXAMPLE",  
      "ResourceType": "vpc",  
    }  
  ]  
}
```

```

        "NetworkInterfaceId": "eni-03847706f6EXAMPLE",
        "GroupMember": false,
        "GroupSource": true,
        "SourceType": "static"
    }
]
}

```

有关更多信息，请参阅《传输网关指南》中的[管理多播组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SearchTransitGatewayMulticastGroups](#)中的。

## search-transit-gateway-routes

以下代码示例演示了如何使用 search-transit-gateway-routes。

### AWS CLI

在指定的公网网关路由表中搜索路由

以下search-transit-gateway-routes示例返回指定路由表static中所有类型的路由。

```

aws ec2 search-transit-gateway-routes \
  --transit-gateway-route-table-id tgw-rtb-0a823edbdeEXAMPLE \
  --filters "Name=type,Values=static"

```

输出：

```

{
  "Routes": [
    {
      "DestinationCidrBlock": "10.0.2.0/24",
      "TransitGatewayAttachments": [
        {
          "ResourceId": "vpc-4EXAMPLE",
          "TransitGatewayAttachmentId": "tgw-attach-09b52ccdb5EXAMPLE",
          "ResourceType": "vpc"
        }
      ],
      "Type": "static",
      "State": "active"
    },
    {

```

```
    "DestinationCidrBlock": "10.1.0.0/24",
    "TransitGatewayAttachments": [
      {
        "ResourceId": "vpc-4EXAMPLE",
        "TransitGatewayAttachmentId": "tgw-attach-09b52ccdb5EXAMPLE",
        "ResourceType": "vpc"
      }
    ],
    "Type": "static",
    "State": "active"
  }
],
"AdditionalRoutesAvailable": false
}
```

有关更多信息，请参阅《[公网网关指南](#)》中的 [Transit Gateway 路由表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SearchTransitGatewayRoutes](#)中的。

## send-diagnostic-interrupt

以下代码示例演示了如何使用 send-diagnostic-interrupt。

### AWS CLI

#### 发送诊断中断

以下send-diagnostic-interrupt示例向指定实例发送诊断中断。

```
aws ec2 send-diagnostic-interrupt \
  --instance-id i-1234567890abcdef0
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SendDiagnosticInterrupt](#)中的。

## start-instances

以下代码示例演示了如何使用 start-instances。

### AWS CLI

#### 启动 Amazon EC2 实例

本示例启动指定的受 Amazon EBS 支持的实例。

命令:

```
aws ec2 start-instances --instance-ids i-1234567890abcdef0
```

输出:

```
{
  "StartingInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "CurrentState": {
        "Code": 0,
        "Name": "pending"
      },
      "PreviousState": {
        "Code": 80,
        "Name": "stopped"
      }
    }
  ]
}
```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的“停止和启动实例”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartInstances](#)中的。

## start-network-insights-access-scope-analysis

以下代码示例演示了如何使用 start-network-insights-access-scope-analysis。

AWS CLI

开始网络见解访问范围分析

以下start-network-insights-access-scope-analysis示例在您的 AWS 账户中启动范围分析。

```
aws ec2 start-network-insights-access-scope-analysis \
  --region us-east-1 \
  --network-insights-access-scope-id nis-123456789111
```

输出：

```
{
  "NetworkInsightsAccessScopeAnalysis": {
    "NetworkInsightsAccessScopeAnalysisId": "nisa-123456789222",
    "NetworkInsightsAccessScopeAnalysisArn": "arn:aws:ec2:us-
east-1:123456789012:network-insights-access-scope-analysis/nisa-123456789222",
    "NetworkInsightsAccessScopeId": "nis-123456789111",
    "Status": "running",
    "StartDate": "2022-01-26T00:47:06.814000+00:00"
  }
}
```

有关更多信息，请参阅《[网络访问分析器指南](#)》中的“[使用 AWS CLI 开始使用网络访问分析器](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartNetworkInsightsAccessScopeAnalysis](#) 中的。

## start-network-insights-analysis

以下代码示例演示了如何使用 start-network-insights-analysis。

### AWS CLI

#### 分析路径

以下 start-network-insights-analysis 示例分析了源和目标之间的路径。要查看路径分析的结果，请使用 describe-network-insights-analyses 命令。

```
aws ec2 start-network-insights-analysis \
  --network-insights-path-id nip-0b26f224f1d131fa8
```

输出：

```
{
  "NetworkInsightsAnalysis": {
    "NetworkInsightsAnalysisId": "nia-02207aa13eb480c7a",
    "NetworkInsightsAnalysisArn": "arn:aws:ec2:us-east-1:123456789012:network-
insights-analysis/nia-02207aa13eb480c7a",
    "NetworkInsightsPathId": "nip-0b26f224f1d131fa8",
    "StartDate": "2021-01-20T22:58:37.495Z",
  }
}
```



```
    "Status": "running"
  }
}
```

有关更多信息，请参阅 [《Reachability Analyzer 指南》](#) 中的 [AWS CLI 使用入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartNetworkInsightsAnalysis](#) 中的。

## start-vpc-endpoint-service-private-dns-verification

以下代码示例演示了如何使用 `start-vpc-endpoint-service-private-dns-verification`。

### AWS CLI

启动 DNS 验证流程

以下 `start-vpc-endpoint-service-private-dns-verification` 示例启动指定终端节点服务的 DNS 验证过程。

```
aws ec2 start-vpc-endpoint-service-private-dns-verification \
  --service-id vpce-svc-071afff70666e61e0
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS PrivateLink 用户指南》中的 [“管理 DNS 名称”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartVpcEndpointServicePrivateDnsVerification](#) 中的。

## stop-instances

以下代码示例演示了如何使用 `stop-instances`。

### AWS CLI

示例 1：停止 Amazon EC2 实例

以下 `stop-instances` 示例将停止指定的受 Amazon EBS 支持的实例。

```
aws ec2 stop-instances \
  --instance-ids i-1234567890abcdef0
```

输出：

```
{
  "StoppingInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "CurrentState": {
        "Code": 64,
        "Name": "stopping"
      },
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的[停止和启动实例](#)。

示例 2：使 Amazon EC2 实例进入休眠状态

以下 `stop-instances` 示例将让受 Amazon EBS 支持的实例进入休眠，前提是实例已启用休眠并且满足休眠先决条件。实例进入休眠状态后，实例将停止。

```
aws ec2 stop-instances \
  --instance-ids i-1234567890abcdef0 \
  --hibernate
```

输出：

```
{
  "StoppingInstances": [
    {
      "CurrentState": {
        "Code": 64,
        "Name": "stopping"
      },
      "InstanceId": "i-1234567890abcdef0",
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
```

```
    }  
  }  
]  
}
```

有关更多信息，请参阅《Amazon Elastic Cloud Compute 用户指南》中的[休眠按需 Linux 实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopInstances](#)中的。

## terminate-client-vpn-connections

以下代码示例演示了如何使用 terminate-client-vpn-connections。

### AWS CLI

终止与 Client VPN 端点的连接

以下terminate-client-vpn-connections示例终止与 Client VPN 端点的指定连接。

```
aws ec2 terminate-client-vpn-connections \  
  --client-vpn-endpoint-id vpn-endpoint-123456789123abcde \  
  --connection-id cvpn-connection-04edd76f5201e0cb8
```

输出：

```
{  
  "ClientVpnEndpointId": "vpn-endpoint-123456789123abcde",  
  "ConnectionStatuses": [  
    {  
      "ConnectionId": "cvpn-connection-04edd76f5201e0cb8",  
      "PreviousStatus": {  
        "Code": "active"  
      },  
      "CurrentStatus": {  
        "Code": "terminating"  
      }  
    }  
  ]  
}
```

有关更多信息，请参阅《Client VPN 管理员指南》中的“AWS 客户端[连接](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TerminateClientVpnConnections](#)中的。

## terminate-instances

以下代码示例演示了如何使用 terminate-instances。

### AWS CLI

终止 Amazon EC2 实例

本示例将终止指定的实例。

命令:

```
aws ec2 terminate-instances --instance-ids i-1234567890abcdef0
```

输出:

```
{
  "TerminatingInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "CurrentState": {
        "Code": 32,
        "Name": "shutting-down"
      },
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
```

有关更多信息，请参阅《AWS 命令行界面用户指南》中的“使用 Amazon EC2 实例”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TerminateInstances](#)中的。

## unassign-ipv6-addresses

以下代码示例演示了如何使用 unassign-ipv6-addresses。

## AWS CLI

从网络接口取消分配 IPv6 地址

此示例从指定的网络接口取消分配指定的 IPv6 地址。

命令:

```
aws ec2 unassign-ipv6-addresses --ipv6-addresses
2001:db8:1234:1a00:3304:8879:34cf:4071 --network-interface-id eni-23c49b68
```

输出 :

```
{
  "NetworkInterfaceId": "eni-23c49b68",
  "UnassignedIpv6Addresses": [
    "2001:db8:1234:1a00:3304:8879:34cf:4071"
  ]
}
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [UnassignIpv6Addresses](#)。

## unassign-private-ip-addresses

以下代码示例演示了如何使用 unassign-private-ip-addresses。

## AWS CLI

从网络接口取消分配辅助私有 IP 地址

此示例取消指定网络接口的指定私有 IP 地址的分配。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 unassign-private-ip-addresses --network-interface-id eni-e5aa89a3 --private-
ip-addresses 10.0.0.82
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UnassignPrivateIpAddresses](#)中的。

## unassign-private-nat-gateway-address

以下代码示例演示了如何使用 unassign-private-nat-gateway-address。

## AWS CLI

### 取消分配私有 NAT 网关的私有 IP 地址

以下`unassign-private-nat-gateway-address`示例取消指定私有 NAT 网关的指定 IP 地址的分配。

```
aws ec2 unassign-private-nat-gateway-address \
  --nat-gateway-id nat-1234567890abcdef0 \
  --private-ip-addresses 10.0.20.197
```

输出：

```
{
  "NatGatewayId": "nat-0ee3edd182361f662",
  "NatGatewayAddresses": [
    {
      "NetworkInterfaceId": "eni-0065a61b324d1897a",
      "PrivateIp": "10.0.20.197",
      "IsPrimary": false,
      "Status": "unassigning"
    }
  ]
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 [NAT 网关](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UnassignPrivateNatGatewayAddress](#)中的。

## **unmonitor-instances**

以下代码示例演示了如何使用 `unmonitor-instances`。

### AWS CLI

#### 禁用对实例的详细监控

本示例命令禁用对指定实例的详细监控。

命令：

```
aws ec2 unmonitor-instances --instance-ids i-1234567890abcdef0
```

输出：

```
{
  "InstanceMonitorings": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "Monitoring": {
        "State": "disabling"
      }
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UnmonitorInstances](#)中的。

## update-security-group-rule-descriptions-egress

以下代码示例演示了如何使用 update-security-group-rule-descriptions-egress。

### AWS CLI

#### 更新出站安全组规则的描述

以下update-security-group-rule-descriptions-egress示例更新了指定端口和 IPv4 地址范围的安全组规则的描述。描述“Outbound HTTP access to server 2”取代了该规则的所有现有描述。

```
aws ec2 update-security-group-rule-descriptions-egress \
  --group-id sg-02f0d35a850ba727f \
  --ip-permissions
  IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges=[{CidrIp=203.0.113.0/24,Description="Outbound
  HTTP access to server 2"}]
```

输出：

```
{
  "Return": true
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[安全组规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateSecurityGroupRuleDescriptionsEgress](#)中的。

## update-security-group-rule-descriptions-ingress

以下代码示例演示了如何使用 `update-security-group-rule-descriptions-ingress`。

### AWS CLI

#### 示例 1：使用 CIDR 源更新入站安全组规则的描述

以下 `update-security-group-rule-descriptions-ingress` 示例更新了指定端口和 IPv4 地址范围的安全组规则的描述。描述“SSH access from ABC office”取代了该规则的所有现有描述。

```
aws ec2 update-security-group-rule-descriptions-ingress \  
  --group-id sg-02f0d35a850ba727f \  
  --ip-permissions  
  IpProtocol=tcp,FromPort=22,ToPort=22,IpRanges='[{"CidrIp=203.0.113.0/16,Description="SSH  
access from corpnet"}]'
```

输出：

```
{  
  "Return": true  
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[安全组规则](#)。

#### 示例 2：使用前缀列表源更新入站安全组规则的描述

以下 `update-security-group-rule-descriptions-ingress` 示例更新了指定端口和前缀列表的安全组规则的描述。描述“SSH access from ABC office”取代了该规则的所有现有描述。

```
aws ec2 update-security-group-rule-descriptions-ingress \  
  --group-id sg-02f0d35a850ba727f \  
  --ip-permissions  
  IpProtocol=tcp,FromPort=22,ToPort=22,PrefixListIds='[{"PrefixListId=pl-12345678,Description=  
access from corpnet"}]'
```



输出：

```
{
  "Return": true
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[安全组规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateSecurityGroupRuleDescriptionsIngress](#)中的。

## withdraw-byoip-cidr

以下代码示例演示了如何使用 withdraw-byoip-cidr。

### AWS CLI

停止发布地址范围的广告

以下withdraw-byoip-cidr示例停止发布指定地址范围。

```
aws ec2 withdraw-byoip-cidr
  --cidr 203.0.113.25/24
```

输出：

```
{
  "ByoipCidr": {
    "Cidr": "203.0.113.25/24",
    "StatusMessage": "ipv4pool-ec2-1234567890abcdef0",
    "State": "advertised"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[WithdrawByoipCidr](#)中的。

## 使用 Amazon EC2 Instance Connect 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon EC2 Instance Connect 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## send-ssh-public-key

以下代码示例演示了如何使用 send-ssh-public-key。

AWS CLI

向实例发送 SSH 公钥

以下 send-ssh-public-key 示例将指定的 SSH 公钥发送到指定的实例。该密钥用于对指定用户进行身份验证。

```
aws ec2-instance-connect send-ssh-public-key \  
  --instance-id i-1234567890abcdef0 \  
  --instance-os-user ec2-user \  
  --availability-zone us-east-2b \  
  --ssh-public-key file:///path/my-rsa-key.pub
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SendSshPublicKey](#) 中的。

## 使用 Amazon ECR 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon ECR 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## batch-check-layer-availability

以下代码示例演示了如何使用 batch-check-layer-availability。

AWS CLI

检查图层的可用性

以下batch-check-layer-availability示例检查cluster-autoscaler存储库中包含摘要的图

层sha256:6171c7451a50945f8ddd72f7732cc04d7a0d1f48138a426b2e64387fdeb834ed的可用性。

```
aws ecr batch-check-layer-availability \  
  --repository-name cluster-autoscaler \  
  --layer-digests  
  sha256:6171c7451a50945f8ddd72f7732cc04d7a0d1f48138a426b2e64387fdeb834ed
```

输出：

```
{  
  "layers": [  
    {  
      "layerDigest":  
      "sha256:6171c7451a50945f8ddd72f7732cc04d7a0d1f48138a426b2e64387fdeb834ed",  
      "layerAvailability": "AVAILABLE",  
      "layerSize": 2777,  
      "mediaType": "application/vnd.docker.container.image.v1+json"  
    }  
  ],  
  "failures": []
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchCheckLayerAvailability](#)中的。

## batch-delete-image

以下代码示例演示了如何使用 batch-delete-image。

### AWS CLI

#### 示例 1：删除图像

以下batch-delete-image示例删除了账户默认注册表precise中指定存储库中带有标签的图像。

```
aws ecr batch-delete-image \  
  --repository-name ubuntu \  
  --image-ids imageTag=precise
```

输出：

```
{  
  "failures": [],  
  "imageIds": [  
    {  
      "imageTag": "precise",  
      "imageDigest":  
      "sha256:19665f1e6d1e504117a1743c0a3d3753086354a38375961f2e665416ef4b1b2f"  
    }  
  ]  
}
```

#### 示例 2：删除多张图片

以下batch-delete-image示例删除指定存储库team1中所有用prod和标记的图像。

```
aws ecr batch-delete-image \  
  --repository-name MyRepository \  
  --image-ids imageTag=prod imageTag=team1
```

输出：

```
{
  "imageIds": [
    {
      "imageDigest": "sha256:123456789012",
      "imageTag": "prod"
    },
    {
      "imageDigest": "sha256:567890121234",
      "imageTag": "team1"
    }
  ],
  "failures": []
}
```

有关更多信息，请参阅 Amazon ECR 用户指南中的[删除图片](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[BatchDeleteImage](#)中的。

## batch-get-image

以下代码示例演示了如何使用 batch-get-image。

### AWS CLI

#### 示例 1：获取图片

以下 batch-get-image 示例在账户的默认注册表 v1.13.6 中名为的存储库 cluster-autoscaler 中获取带有标签的图像。

```
aws ecr batch-get-image \
  --repository-name cluster-autoscaler \
  --image-ids imageTag=v1.13.6
```

输出：

```
{
  "images": [
    {
      "registryId": "012345678910",
      "repositoryName": "cluster-autoscaler",
      "imageId": {
```

```

        "imageDigest":
        "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
        "imageTag": "v1.13.6"
    },
    "imageManifest": "{\n  \"schemaVersion\": 2,\n
  \"mediaType\": \"application/vnd.docker.distribution.manifest.v2+json
  \",\n  \"config\": {\n    \"mediaType\": \"application/
  vnd.docker.container.image.v1+json\",,\n    \"size\": 2777,\n    \"digest
  \": \"sha256:6171c7451a50945f8ddd72f7732cc04d7a0d1f48138a426b2e64387fdeb834ed
  \"\n  },\n  \"layers\": [\n    {\n      \"mediaType
  \": \"application/vnd.docker.image.rootfs.diff.tar.gzip
  \",,\n      \"size\": 17743696,\n      \"digest\":
  \"sha256:39fafc05754f195f134ca11ecdb1c9a691ab0848c697fffefb5a85f900caaf6e1\"\n
    },\n    {\n      \"mediaType\": \"application/
  vnd.docker.image.rootfs.diff.tar.gzip\",,\n      \"size\": 2565026,\n
    \"digest\":
  \"sha256:8c8a779d3a537b767ae1091fe6e00c2590afd16767aa6096d1b318d75494819f
  \"\n    },\n    {\n      \"mediaType\": \"application/
  vnd.docker.image.rootfs.diff.tar.gzip\",,\n      \"size\": 28005981,\n
    \"digest\":
  \"sha256:c44ba47496991c9982ee493b47fd25c252caabf2b4ae7dd679c9a27b6a3c8fb7\"\n
    },\n    {\n      \"mediaType\": \"application/
  vnd.docker.image.rootfs.diff.tar.gzip\",,\n      \"size\": 775,\n      \"digest
  \": \"sha256:e2c388b44226544363ca007be7b896bcce1baebea04da23cbd165eac30be650f\"\n
    }\n  ]\n}"
  ],
  "failures": []
}

```

## 示例 2：获取多张图片

以下batch-get-image示例显示了指定存储库team1中使用prod和标记的所有图像的详细信息。

```

aws ecr batch-get-image \
  --repository-name MyRepository \
  --image-ids imageTag=prod imageTag=team1

```

输出：

```

{
  "images": [
    {

```

```
    "registryId": "123456789012",
    "repositoryName": "MyRepository",
    "imageId": {
      "imageDigest": "sha256:123456789012",
      "imageTag": "prod"
    },
    "imageManifest": "manifestExample1"
  },
  {
    "registryId": "567890121234",
    "repositoryName": "MyRepository",
    "imageId": {
      "imageDigest": "sha256:123456789012",
      "imageTag": "team1"
    },
    "imageManifest": "manifestExample2"
  }
],
"failures": []
}
```

有关更多信息，请参阅 Amazon ECR 用户指南中的[图片](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[BatchGetImage](#)中的。

## complete-layer-upload

以下代码示例演示了如何使用 complete-layer-upload。

### AWS CLI

要完成图像图层上传

以下 complete-layer-upload 示例完成了向 layer-test 存储库的图像层上传。

```
aws ecr complete-layer-upload \
  --repository-name layer-test \
  --upload-id 6cb64b8a-9378-0e33-2ab1-b780fab8a9e9 \
  --layer-digests 6cb64b8a-9378-0e33-2ab1-
b780fab8a9e9:48074e6d3a68b39aad8ccc002cdad912d4148c0f92b3729323e
```

输出：

```
{
  "uploadId": "6cb64b8a-9378-0e33-2ab1-b780fab8a9e9",
  "layerDigest":
    "sha256:9a77f85878aa1906f2020a0ecdf7a7e962d57e882250acd773383224b3fe9a02",
  "repositoryName": "layer-test",
  "registryId": "130757420319"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CompleteLayerUpload](#)中的。

## create-repository

以下代码示例演示了如何使用 create-repository。

### AWS CLI

#### 示例 1：创建存储库

以下create-repository示例在账户的默认注册表中的指定命名空间内创建一个存储库。

```
aws ecr create-repository \
  --repository-name project-a/nginx-web-app
```

输出：

```
{
  "repository": {
    "registryId": "123456789012",
    "repositoryName": "sample-repo",
    "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-a/nginx-web-app"
  }
}
```

有关更多信息，请参阅 Amazon ECR 用户指南中的[创建存储库](#)。

#### 示例 2：创建配置了图像标签不可变性的存储库

以下create-repository示例在账户的默认注册表中创建了一个为标签不可变性配置的存储库。

```
aws ecr create-repository \
```



```
--repository-name sample-repo \  
--image-tag-mutability IMMUTABLE
```

输出：

```
{  
  "repository": {  
    "registryId": "123456789012",  
    "repositoryName": "sample-repo",  
    "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/sample-  
repo",  
    "imageTagMutability": "IMMUTABLE"  
  }  
}
```

有关更多信息，请参阅 Amazon ECR 用户指南中的[图像标签可变性](#)。

示例 3：创建配置了扫描配置的存储库

以下 `create-repository` 示例在账户的默认注册表中创建了一个配置为在映像推送时执行漏洞扫描的存储库。

```
aws ecr create-repository \  
  --repository-name sample-repo \  
  --image-scanning-configuration scanOnPush=true
```

输出：

```
{  
  "repository": {  
    "registryId": "123456789012",  
    "repositoryName": "sample-repo",  
    "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/sample-  
repo",  
    "imageScanningConfiguration": {  
      "scanOnPush": true  
    }  
  }  
}
```

有关更多信息，请参阅 Amazon ECR 用户指南中的[图像扫描](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRepository](#)中的。

## delete-lifecycle-policy

以下代码示例演示了如何使用 delete-lifecycle-policy。

### AWS CLI

#### 删除存储库的生命周期策略

以下delete-lifecycle-policy示例删除了hello-world存储库的生命周期策略。

```
aws ecr delete-lifecycle-policy \  
  --repository-name hello-world
```

输出：

```
{  
  "registryId": "012345678910",  
  "repositoryName": "hello-world",  
  "lifecyclePolicyText": "{\"rules\": [{\"rulePriority\": 1, \"description\": \"Remove  
untagged images.\", \"selection\": {\"tagStatus\": \"untagged\", \"countType\":  
\"sinceImagePushed\", \"countUnit\": \"days\", \"countNumber\": 10}, \"action\": {\"type  
\": \"expire\"}}]}\",  
  "lastEvaluatedAt": 0.0  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLifecyclePolicy](#)中的。

## delete-repository-policy

以下代码示例演示了如何使用 delete-repository-policy。

### AWS CLI

#### 删除存储库的存储库策略

以下delete-repository-policy示例删除了存储库的cluster-autoscaler存储库策略。

```
aws ecr delete-repository-policy \  
  --repository-name cluster-autoscaler
```

输出：

```
{
  "registryId": "012345678910",
  "repositoryName": "cluster-autoscaler",
  "policyText": "{\n  \"Version\" : \"2008-10-17\",\n  \"Statement\" : [ {\n    \"Sid\" : \"allow public pull\",\n    \"Effect\" : \"Allow\",\n    \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchCheckLayerAvailability\", \"ecr:BatchGetImage\", \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRepositoryPolicy](#)中的。

## delete-repository

以下代码示例演示了如何使用 delete-repository。

### AWS CLI

#### 删除存储库

以下delete-repository示例命令强制删除帐户默认注册表中的指定存储库。如果存储库包含图像，则必须使用该--force标志。

```
aws ecr delete-repository \
  --repository-name ubuntu \
  --force
```

输出：

```
{
  "repository": {
    "registryId": "123456789012",
    "repositoryName": "ubuntu",
    "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/ubuntu"
  }
}
```

有关更多信息，请参阅 Amazon ECR 用户指南中的[删除存储库](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRepository](#)中的。

## describe-image-scan-findings

以下代码示例演示了如何使用 `describe-image-scan-findings`。

### AWS CLI

#### 描述图像的扫描结果

以下 `describe-image-scan-findings` 示例使用账户默认注册表中指定存储库中的图像摘要返回图像扫描结果。

```
aws ecr describe-image-scan-findings \  
  --repository-name sample-repo \  
  --image-id  
  imageDigest=sha256:74b2c688c700ec95a93e478cdb959737c148df3fbf5ea706abe0318726e885e6
```

输出：

```
{  
  "imageScanFindings": {  
    "findings": [  
      {  
        "name": "CVE-2019-5188",  
        "description": "A code execution vulnerability exists in the directory  
rehashing functionality of E2fsprogs e2fsck 1.45.4. A specially crafted ext4  
directory can cause an out-of-bounds write on the stack, resulting in code  
execution. An attacker can corrupt a partition to trigger this vulnerability.",  
        "uri": "http://people.ubuntu.com/~ubuntu-security/cve/CVE-2019-5188",  
        "severity": "MEDIUM",  
        "attributes": [  
          {  
            "key": "package_version",  
            "value": "1.44.1-1ubuntu1.1"  
          },  
          {  
            "key": "package_name",  
            "value": "e2fsprogs"  
          },  
          {  
            "key": "CVSS2_VECTOR",  
            "value": "AV:L/AC:L/Au:N/C:P/I:P/A:P"  
          },  
          {
```

```

        "key": "CVSS2_SCORE",
        "value": "4.6"
      }
    ]
  },
  "imageScanCompletedAt": 1579839105.0,
  "vulnerabilitySourceUpdatedAt": 1579811117.0,
  "findingSeverityCounts": {
    "MEDIUM": 1
  }
},
"registryId": "123456789012",
"repositoryName": "sample-repo",
"imageId": {
  "imageDigest":
"sha256:74b2c688c700ec95a93e478cdb959737c148df3fbf5ea706abe0318726e885e6"
},
"imageScanStatus": {
  "status": "COMPLETE",
  "description": "The scan was completed successfully."
}
}

```

有关更多信息，请参阅 Amazon ECR 用户指南中的[图像扫描](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeImageScanFindings](#) 中的。

## describe-images

以下代码示例演示了如何使用 describe-images。

### AWS CLI

#### 描述存储库中的图像

以下 describe-images 示例显示了 cluster-autoscaler 存储库中带有标签 v1.13.6 的图像的信息。

```

aws ecr describe-images \
  --repository-name cluster-autoscaler \
  --image-ids imageTag=v1.13.6

```

输出：

```
{
  "imageDetails": [
    {
      "registryId": "012345678910",
      "repositoryName": "cluster-autoscaler",
      "imageDigest":
"sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
      "imageTags": [
        "v1.13.6"
      ],
      "imageSizeInBytes": 48318255,
      "imagePushedAt": 1565128275.0
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeImages](#)中的。

## describe-repositories

以下代码示例演示了如何使用 describe-repositories。

### AWS CLI

在注册表中描述存储库

此示例将描述账户默认注册表中的存储库。

命令：

```
aws ecr describe-repositories
```

输出：

```
{
  "repositories": [
    {
      "registryId": "012345678910",
      "repositoryName": "ubuntu",
```

```
        "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/ubuntu"
      },
      {
        "registryId": "012345678910",
        "repositoryName": "test",
        "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/test"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeRepositories](#)中的。

## get-authorization-token

以下代码示例演示了如何使用 `get-authorization-token`。

### AWS CLI

获取默认注册表的授权令牌

以下`get-authorization-token`示例命令获取默认注册表的授权令牌。

```
aws ecr get-authorization-token
```

输出：

```
{
  "authorizationData": [
    {
      "authorizationToken": "QVdT0kN...",
      "expiresAt": 1448875853.241,
      "proxyEndpoint": "https://123456789012.dkr.ecr.us-west-2.amazonaws.com"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAuthorizationToken](#)中的。

## get-download-url-for-layer

以下代码示例演示了如何使用 `get-download-url-for-layer`。

## AWS CLI

### 获取图层的下载 URL

以下`get-download-url-for-layer`示例显示`cluster-autoscaler`存储库中包含摘要`sha256:6171c7451a50945f8ddd72f7732cc04d7a0d1f48138a426b2e64387fdeb834ed`的图层的下载 URL。

```
aws ecr get-download-url-for-layer \
  --repository-name cluster-autoscaler \
  --layer-digest
sha256:6171c7451a50945f8ddd72f7732cc04d7a0d1f48138a426b2e64387fdeb834ed
```

输出：

```
{
  "downloadUrl": "https://prod-us-west-2-starport-layer-bucket.s3.us-
west-2.amazonaws.com/e501-012345678910-9cb60dc0-7284-5643-3987-
da6dac0465f0/04620aac-66a5-4167-8232-55ee7ef6d565?X-Amz-Algorithm=AWS4-HMAC-
SHA256&X-Amz-Date=20190814T220617Z&X-Amz-SignedHeaders=host&X-Amz-Expires=3600&X-
Amz-Credential=AKIA32P3D2JDNMVAJLGF%2F20190814%2Fus-west-2%2Fs3%2Faws4_request&X-
Amz-Signature=9161345894947a1672467a0da7a1550f2f7157318312fe4941b59976239c3337",
  "layerDigest":
  "sha256:6171c7451a50945f8ddd72f7732cc04d7a0d1f48138a426b2e64387fdeb834ed"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDownloadUrlForLayer](#)中的。

## get-lifecycle-policy-preview

以下代码示例演示了如何使用 `get-lifecycle-policy-preview`。

### AWS CLI

#### 检索生命周期策略预览的详细信息

以下`get-lifecycle-policy-preview`示例检索账户的默认注册表中指定存储库的生命周期策略预览结果。

命令：

```
aws ecr get-lifecycle-policy-preview \
```



```
--repository-name "project-a/amazon-ecs-sample"
```

输出：

```
{
  "registryId": "012345678910",
  "repositoryName": "project-a/amazon-ecs-sample",
  "lifecyclePolicyText": "{\n  \"rules\": [\n    {\n      \"rulePriority\": 1,\n      \"description\": \"Expire images older than 14 days\",\n      \"selection\": {\n        \"tagStatus\": \"untagged\",\n        \"countType\": \"sinceImagePushed\",\n        \"countUnit\": \"days\",\n        \"countNumber\": 14\n      },\n      \"action\": {\n        \"type\": \"expire\"\n      }\n    }\n  ]\n}",
  "status": "COMPLETE",
  "previewResults": [],
  "summary": {
    "expiringImageTotalCount": 0
  }
}
```

有关更多信息，请参阅 Amazon ECR 用户指南中的[生命周期策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetLifecyclePolicyPreview](#)中的。

## get-lifecycle-policy

以下代码示例演示了如何使用 get-lifecycle-policy。

### AWS CLI

#### 检索生命周期策略

以下 get-lifecycle-policy 示例在账户的默认注册表中显示指定存储库的生命周期策略的详细信息。

```
aws ecr get-lifecycle-policy \
  --repository-name "project-a/amazon-ecs-sample"
```

输出：

```
{
```

```
    "registryId": "123456789012",
    "repositoryName": "project-a/amazon-ecs-sample",
    "lifecyclePolicyText": "{\"rules\": [{\"rulePriority\": 1, \"description\": \"Expire images older than 14 days\", \"selection\": {\"tagStatus\": \"untagged\", \"countType\": \"sinceImagePushed\", \"countUnit\": \"days\", \"countNumber\": 14}, \"action\": {\"type\": \"expire\"}}]}",
    "lastEvaluatedAt": 1504295007.0
  }
```

有关更多信息，请参阅 Amazon ECR 用户指南中的[生命周期策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetLifecyclePolicy](#)中的。

## get-login-password

以下代码示例演示了如何使用 get-login-password。

### AWS CLI

检索密码以向注册表进行身份验证

下面 get-login-password 显示了一个密码，您可以将该密码与您选择的容器客户端一起使用，对您的 IAM 委托人有权访问的任何 Amazon ECR 注册表进行身份验证。

```
aws ecr get-login-password
```

输出：

```
<password>
```

要与 Docker CLI 一起使用，请将命令的输出通过管道 get-login-password 传输到该 docker login 命令。在检索密码时，请确保您指定的区域与 Amazon ECR 注册表所在的区域相同。

```
aws ecr get-login-password \
  --region <region> \
| docker login \
  --username AWS \
  --password-stdin <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

有关更多信息，请参阅 Amazon ECR 用户指南中的[注册身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLoginPassword](#)中的。

## get-login

以下代码示例演示了如何使用 get-login。

### AWS CLI

将 Docker 登录命令检索到您的默认注册表

此示例打印了一个命令，您可以使用该命令登录默认 Amazon ECR 注册表。

命令:

```
aws ecr get-login
```

输出:

```
docker login -u AWS -p <password> -e none https://  
<aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

登录其他账户的注册表

此示例打印了一个或多个命令，您可以使用这些命令登录与其他账户关联的 Amazon ECR 注册表。

命令:

```
aws ecr get-login --registry-ids 012345678910 023456789012
```

输出:

```
docker login -u <username> -p <token-1> -e none <endpoint-1>  
docker login -u <username> -p <token-2> -e none <endpoint-2>
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLogin](#)中的。

## get-repository-policy

以下代码示例演示了如何使用 get-repository-policy。

## AWS CLI

### 检索存储库的存储库策略

以下`get-repository-policy`示例显示了有关存储库策略的`cluster-autoscaler`详细信息。

```
aws ecr get-repository-policy \  
  --repository-name cluster-autoscaler
```

输出：

```
{  
  "registryId": "012345678910",  
  "repositoryName": "cluster-autoscaler",  
  "policyText": "{\n  \"Version\" : \"2008-10-17\",\n  \"Statement\" : [ {\n    \"Sid\" : \"allow public pull\",\n    \"Effect\" : \"Allow\",\n    \"Principal\" :  
    \"*\",\n    \"Action\" : [ \"ecr:BatchCheckLayerAvailability\", \"ecr:BatchGetImage  
\", \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRepositoryPolicy](#)中的。

### initiate-layer-upload

以下代码示例演示了如何使用 `initiate-layer-upload`。

## AWS CLI

### 启动图像图层上传

以下`initiate-layer-upload`示例启动图像层上传到`layer-test`存储库。

```
aws ecr initiate-layer-upload \  
  --repository-name layer-test
```

输出：

```
{  
  "partSize": 10485760,
```

```
"uploadId": "6cb64b8a-9378-0e33-2ab1-b780fab8a9e9"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[InitiateLayerUpload](#)中的。

## list-images

以下代码示例演示了如何使用 `list-images`。

### AWS CLI

列出存储库的映像

以下 `list-images` 示例将显示 `cluster-autoscaler` 存储库的映像列表。

```
aws ecr list-images \
  --repository-name cluster-autoscaler
```

输出：

```
{
  "imageIds": [
    {
      "imageDigest":
"sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
      "imageTag": "v1.13.8"
    },
    {
      "imageDigest":
"sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
      "imageTag": "v1.13.7"
    },
    {
      "imageDigest":
"sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
      "imageTag": "v1.13.6"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListImages](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出存储库的标签

以下 `list-tags-for-resource` 示例显示了与 `hello-world` 存储库关联的标签列表。

```
aws ecr list-tags-for-resource \  
  --resource-arn arn:aws:ecr:us-west-2:012345678910:repository/hello-world
```

输出：

```
{  
  "tags": [  
    {  
      "Key": "Stage",  
      "Value": "Integ"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## put-image-scanning-configuration

以下代码示例演示了如何使用 `put-image-scanning-configuration`。

### AWS CLI

更新存储库的图像扫描配置

以下 `put-image-scanning-configuration` 示例更新了指定存储库的图像扫描配置。

```
aws ecr put-image-scanning-configuration \  
  --repository-name sample-repo \  
  --image-scanning-configuration scanOnPush=true
```

输出：

```
{
  "registryId": "012345678910",
  "repositoryName": "sample-repo",
  "imageScanningConfiguration": {
    "scanOnPush": true
  }
}
```

有关更多信息，请参阅 Amazon ECR 用户指南中的[图像扫描](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutImageScanningConfiguration](#)中的。

## put-image-tag-mutability

以下代码示例演示了如何使用 put-image-tag-mutability。

### AWS CLI

更新存储库的图像标签可变性设置

以下put-image-tag-mutability示例为标签不可变性配置了指定的存储库。这样可以防止存储库中的所有图像标签被覆盖。

```
aws ecr put-image-tag-mutability \
  --repository-name hello-repository \
  --image-tag-mutability IMMUTABLE
```

输出：

```
{
  "registryId": "012345678910",
  "repositoryName": "sample-repo",
  "imageTagMutability": "IMMUTABLE"
}
```

有关更多信息，请参阅 Amazon ECR 用户指南中的[图像标签可变性](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutImageTagMutability](#)中的。

## put-image

以下代码示例演示了如何使用 put-image。

## AWS CLI

### 使用其清单重新标记图像

以下put-image示例使用现有图像清单在hello-world存储库中创建一个新标签。

```
aws ecr put-image \  
  --repository-name hello-world \  
  --image-tag 2019.08 \  
  --image-manifest file://hello-world.manifest.json
```

hello-world.manifest.json 的内容：

```
{  
  "schemaVersion": 2,  
  "mediaType": "application/vnd.docker.distribution.manifest.v2+json",  
  "config": {  
    "mediaType": "application/vnd.docker.container.image.v1+json",  
    "size": 5695,  
    "digest":  
    "sha256:cea5fe7701b7db3dd1c372f3cea6f43cdda444fcc488f530829145e426d8b980"  
  },  
  "layers": [  
    {  
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",  
      "size": 39096921,  
      "digest":  
      "sha256:d8868e50ac4c7104d2200d42f432b661b2da8c1e417ccfae217e6a1e04bb9295"  
    },  
    {  
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",  
      "size": 57938,  
      "digest":  
      "sha256:83251ac64627fc331584f6c498b3aba5badc01574e2c70b2499af3af16630eed"  
    },  
    {  
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",  
      "size": 423,  
      "digest":  
      "sha256:589bba2f1b36ae56f0152c246e2541c5aa604b058febfcf2be32e9a304fec610"  
    },  
    {  
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
```



```
        "size": 680,
        "digest":
"sha256:d62ecaceda3964b735cdd2af613d6bb136a52c1da0838b2ff4b4dab4212bcb1c"
    },
    {
        "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
        "size": 162,
        "digest":
"sha256:6d93b41cfc6bf0d2522b7cf61588de4cd045065b36c52bd3aec2ba0622b2b22b"
    },
    {
        "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
        "size": 28268840,
        "digest":
"sha256:6986b4d4c07932c680b3587f2eac8b0e013568c003cc23b04044628a5c5e599f"
    },
    {
        "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
        "size": 35369152,
        "digest":
"sha256:8c5ec60f10102dc8da0649d866c7c2f706e459d0bdc25c83ad2de86f4996c276"
    },
    {
        "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
        "size": 155,
        "digest":
"sha256:cde50b1c594539c5f67cbede9aef95c9ae321ccfb857f7b251b45b84198adc85"
    },
    {
        "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
        "size": 28737,
        "digest":
"sha256:2e102807ab72a73fc9abf53e8c50e421bdc337a0a8afcb242176edeec65977e4"
    },
    {
        "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
        "size": 190,
        "digest":
"sha256:fc379bbd5ed37808772bef016553a297356c59b8f134659e6ee4ecb563c2f5a7"
    },
    {
        "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
        "size": 28748,
```

```

    "digest":
      "sha256:021db240dfccf5a1aff19507d17c0177e5888e518acf295b52204b1825e8b7ee"
    }
  ]
}

```

输出：

```

{
  "image": {
    "registryId": "130757420319",
    "repositoryName": "hello-world",
    "imageId": {
      "imageDigest":
        "sha256:8ece96b74f87652876199d83bd107d0435a196133af383ac54cb82b6cc5283ae",
      "imageTag": "2019.08"
    },
    "imageManifest": "{\n  \"schemaVersion\": 2,\n  \"mediaType
\n: \"application/vnd.docker.distribution.manifest.v2+json
\n,\n  \"config\": {\n    \"mediaType\": \"application/
vnd.docker.container.image.v1+json\",\n    \"size\": 5695,\n    \"digest\":
\n  \"sha256:cea5fe7701b7db3dd1c372f3cea6f43cdda444fcc488f530829145e426d8b980\"\n
  },\n  \"layers\": [\n    {\n      \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n      \"size\": 39096921,\n      \"digest
\n: \"sha256:d8868e50ac4c7104d2200d42f432b661b2da8c1e417ccfae217e6a1e04bb9295\"\n
    },\n    {\n      \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n      \"size\": 57938,\n      \"digest
\n: \"sha256:83251ac64627fc331584f6c498b3aba5badc01574e2c70b2499af3af16630eed
\n\n  },\n    {\n      \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n      \"size\": 423,\n      \"digest\":
\n  \"sha256:589bba2f1b36ae56f0152c246e2541c5aa604b058febfcf2be32e9a304fec610\"\n
    },\n    {\n      \"mediaType\": \"application/vnd.docker.image.rootfs.diff.tar.gzip\",\n
\n      \"size\": 680,\n      \"digest\":
\n  \"sha256:d62ecaceda3964b735cdd2af613d6bb136a52c1da0838b2ff4b4dab4212bcb1c
\n\n  },\n    {\n      \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n      \"size\": 162,\n      \"digest
\n: \"sha256:6d93b41cfc6bf0d2522b7cf61588de4cd045065b36c52bd3aec2ba0622b2b22b
\n\n  },\n    {\n      \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n      \"size\": 28268840,\n      \"digest
\n: \"sha256:6986b4d4c07932c680b3587f2eac8b0e013568c003cc23b04044628a5c5e599f
\n\n  },\n    {\n      \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n      \"size\": 35369152,\n      \"digest
\n: \"sha256:8c5ec60f10102dc8da0649d866c7c2f706e459d0bdc25c83ad2de86f4996c276\"\n
\n  }
}

```

```

    },\n    {\n        \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\", \n        \"size\": 155, \n        \"digest\":
\"sha256:cde50b1c594539c5f67cbede9aef95c9ae321ccfb857f7b251b45b84198adc85\" \n    },
\n    {\n        \"mediaType\": \"application/vnd.docker.image.rootfs.diff.tar.gzip\",
\n        \"size\": 28737, \n        \"digest\":
\"sha256:2e102807ab72a73fc9abf53e8c50e421bdc337a0a8afcb242176edeec65977e4\" \n    },
\n    {\n        \"mediaType\": \"application/vnd.docker.image.rootfs.diff.tar.gzip\",
\n        \"size\": 190, \n        \"digest\":
\"sha256:fc379bbd5ed37808772bef016553a297356c59b8f134659e6ee4ecb563c2f5a7\" \n    },
\n    {\n        \"mediaType\": \"application/vnd.docker.image.rootfs.diff.tar.gzip\",
\n        \"size\": 28748, \n        \"digest\":
\"sha256:021db240dfccf5a1aff19507d17c0177e5888e518acf295b52204b1825e8b7ee\" \n
    } \n ] \n }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutImage](#)中的。

## put-lifecycle-policy

以下代码示例演示了如何使用 `put-lifecycle-policy`。

### AWS CLI

#### 创建生命周期策略

以下`put-lifecycle-policy`示例在账户的默认注册表中为指定存储库创建生命周期策略。

```

aws ecr put-lifecycle-policy \
  --repository-name "project-a/amazon-ecs-sample" \
  --lifecycle-policy-text "file://policy.json"

```

`policy.json` 的内容：

```

{
  "rules": [
    {
      "rulePriority": 1,
      "description": "Expire images older than 14 days",
      "selection": {
        "tagStatus": "untagged",
        "countType": "sinceImagePushed",
        "countUnit": "days",

```

```

        "countNumber": 14
      },
      "action": {
        "type": "expire"
      }
    }
  ]
}

```

输出：

```

{
  "registryId": "<aws_account_id>",
  "repositoryName": "project-a/amazon-ecs-sample",
  "lifecyclePolicyText": "{\"rules\": [{\"rulePriority\": 1, \"description\": \"Expire images older than 14 days\", \"selection\": {\"tagStatus\": \"untagged\", \"countType\": \"sinceImagePushed\", \"countUnit\": \"days\", \"countNumber\": 14}, \"action\": {\"type\": \"expire\"}}]}"
}

```

有关更多信息，请参阅 Amazon ECR 用户指南中的[生命周期策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutLifecyclePolicy](#) 中的。

## set-repository-policy

以下代码示例演示了如何使用 set-repository-policy。

### AWS CLI

为存储库设置存储库策略

以下 set-repository-policy 示例将文件中包含的存储库策略附加到 cluster-autoscaler 存储库。

```

aws ecr set-repository-policy \
  --repository-name cluster-autoscaler \
  --policy-text file://my-policy.json

```

my-policy.json 的内容：

```

{

```

```

"Version" : "2008-10-17",
"Statement" : [
  {
    "Sid" : "allow public pull",
    "Effect" : "Allow",
    "Principal" : "*",
    "Action" : [
      "ecr:BatchCheckLayerAvailability",
      "ecr:BatchGetImage",
      "ecr:GetDownloadUrlForLayer"
    ]
  }
]
}

```

输出：

```

{
  "registryId": "012345678910",
  "repositoryName": "cluster-autoscaler",
  "policyText": "{\n  \"Version\" : \"2008-10-17\",\n  \"Statement\" : [ {\n\n    \"Sid\" : \"allow public pull\",\n    \"Effect\" : \"Allow\",\n    \"Principal\" :\n    \"*\",\n    \"Action\" : [ \"ecr:BatchCheckLayerAvailability\", \"ecr:BatchGetImage\n    \", \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetRepositoryPolicy](#)中的。

## start-image-scan

以下代码示例演示了如何使用 start-image-scan。

### AWS CLI

#### 启动图像漏洞扫描

以下start-image-scan示例启动图像扫描，并由指定存储库中的图像摘要指定。

```

aws ecr start-image-scan \
  --repository-name sample-repo \
  --image-id
imageDigest=sha256:74b2c688c700ec95a93e478cdb959737c148df3fbf5ea706abe0318726e885e6

```

输出：

```
{
  "registryId": "012345678910",
  "repositoryName": "sample-repo",
  "imageId": {
    "imageDigest":
"sha256:74b2c688c700ec95a93e478cdb959737c148df3fbf5ea706abe0318726e885e6"
  },
  "imageScanStatus": {
    "status": "IN_PROGRESS"
  }
}
```

有关更多信息，请参阅 Amazon ECR 用户指南中的[图像扫描](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[StartImageScan](#)中的。

## start-lifecycle-policy-preview

以下代码示例演示了如何使用 start-lifecycle-policy-preview。

### AWS CLI

#### 创建生命周期策略预览

以下 start-lifecycle-policy-preview 示例创建了由指定存储库的 JSON 文件定义的生命周期策略预览。

```
aws ecr start-lifecycle-policy-preview \
  --repository-name "project-a/amazon-ecs-sample" \
  --lifecycle-policy-text "file://policy.json"
```

policy.json 的内容：

```
{
  "rules": [
    {
      "rulePriority": 1,
      "description": "Expire images older than 14 days",
      "selection": {
        "tagStatus": "untagged",
```

```

        "countType": "sinceImagePushed",
        "countUnit": "days",
        "countNumber": 14
    },
    "action": {
        "type": "expire"
    }
}
]
}

```

输出：

```

{
  "registryId": "012345678910",
  "repositoryName": "project-a/amazon-ecs-sample",
  "lifecyclePolicyText": "{\n  \\"rules\":" [ \n      {\n
  \\"rulePriority\":" 1,\n      \\"description\":" \\"Expire images older than 14
  days\"," \n      \\"selection\":" {\n          \\"tagStatus\":" \\"untagged\","
  \n          \\"countType\":" \\"sinceImagePushed\"," \n          \\"countUnit
  \":" \\"days\"," \n          \\"countNumber\":" 14\n      }, \n
  \\"action\":" {\n          \\"type\":" \\"expire\\"\n      } \n      } \n
  ] \n } \n",
  "status": "IN_PROGRESS"
}

```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartLifecyclePolicyPreview](#) 中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为存储库添加标签

以下 tag-resource 示例在 hello-world 存储库 Integ 上设置了一个带有键 Stage 和值的标签。

```

aws ecr tag-resource \
  --resource-arn arn:aws:ecr:us-west-2:012345678910:repository/hello-world \
  --tags Key=Stage,Value=Integ

```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

取消存储库的标签

以下untag-resource示例Stage从hello-world存储库中删除带有密钥的标签。

```
aws ecr untag-resource \  
  --resource-arn arn:aws:ecr:us-west-2:012345678910:repository/hello-world \  
  --tag-keys Stage
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## upload-layer-part

以下代码示例演示了如何使用 upload-layer-part。

### AWS CLI

上传图层分段

以下内容upload-layer-part将图像层部分上传到layer-test存储库。

```
aws ecr upload-layer-part \  
  --repository-name layer-test \  
  --upload-id 6cb64b8a-9378-0e33-2ab1-b780fab8a9e9 \  
  --part-first-byte 0 \  
  --part-last-byte 8323314 \  
  --layer-part-blob file:///var/lib/docker/image/overlay2/layerdb/sha256/  
ff986b10a018b48074e6d3a68b39aad8ccc002cdad912d4148c0f92b3729323e/layer.b64
```

输出：

```
{
```



```
"uploadId": "6cb64b8a-9378-0e33-2ab1-b780fab8a9e9",
"registryId": "012345678910",
"lastByteReceived": 8323314,
"repositoryName": "layer-test"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UploadLayerPart](#)中的。

## 使用 Amazon ECS 示例 AWS CLI

以下代码示例向您展示了如何在 Amazon ECS 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-capacity-provider**

以下代码示例演示了如何使用 create-capacity-provider。

AWS CLI

创建容量提供商

以下 create-capacity-provider 示例创建了一个容量提供商，该容量提供者使用名为 myASG 的 Auto Scaling 组，启用了托管扩展和托管终止保护。此配置用于 Amazon ECS 集群自动扩展。

```
aws ecs create-capacity-provider \
  --name "MyCapacityProvider" \
  --auto-scaling-group-provider "autoScalingGroupArn=arn:aws:autoscaling:us-
east-1:123456789012:autoScalingGroup:57ffcb94-11f0-4d6d-
```

```
bf60-3bac5EXAMPLE:autoScalingGroupName/  
MyASG,managedScaling={status=ENABLED,targetCapacity=100},managedTerminationProtection=ENABLED
```

输出：

```
{  
  "capacityProvider": {  
    "capacityProviderArn": "arn:aws:ecs:us-east-1:123456789012:capacity-provider/  
MyCapacityProvider",  
    "name": "MyCapacityProvider",  
    "status": "ACTIVE",  
    "autoScalingGroupProvider": {  
      "autoScalingGroupArn": "arn:aws:autoscaling:us-  
east-1:132456789012:autoScalingGroup:57ffcb94-11f0-4d6d-  
bf60-3bac5EXAMPLE:autoScalingGroupName/MyASG",  
      "managedScaling": {  
        "status": "ENABLED",  
        "targetCapacity": 100,  
        "minimumScalingStepSize": 1,  
        "maximumScalingStepSize": 10000,  
        "instanceWarmupPeriod": 300  
      },  
      "managedTerminationProtection": "ENABLED"  
    },  
    "tags": []  
  }  
}
```

有关更多信息，请参阅 [《亚马逊 ECS 开发人员指南》](#) 中的 [Amazon ECS 集群自动扩展](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateCapacityProvider](#) 中的。

## create-cluster

以下代码示例演示了如何使用 create-cluster。

### AWS CLI

#### 示例 1：创建新集群

以下 create-cluster 示例将创建一个集群。

```
aws ecs create-cluster \  
  --cluster-name MyCluster
```

输出：

```
{
  "cluster": {
    "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
    "clusterName": "MyCluster",
    "status": "ACTIVE",
    "registeredContainerInstancesCount": 0,
    "pendingTasksCount": 0,
    "runningTasksCount": 0,
    "activeServicesCount": 0,
    "statistics": [],
    "tags": []
  }
}
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[创建集群](#)。

#### 示例 2：使用容量提供程序创建新集群

以下 `create-cluster` 示例将创建一个集群并将两个现有容量提供程序与该集群相关联。使用 `create-capacity-provider` 命令创建容量提供程序。指定默认容量提供程序策略是可选的，但建议您这样做。在此示例中，我们创建一个名为 `MyCluster` 的集群，并将 `MyCapacityProvider1` 和 `MyCapacityProvider2` 容量提供程序与其相关联。指定默认容量提供程序策略，将任务平均分散到两个容量提供程序。

```
aws ecs create-cluster-name--cluster-name--c default-capacity-provider-strategy ap
MyCapacityProvider acityProvider 1 2 MyCluster — capacityProvider= MyCapacityProvider
1、weight=1 capacityProvider= MyCapacityProvider 2、w MyCapacityProvider
```

输出：

```
{
  "cluster": {
    "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
    "clusterName": "MyCluster",
    "status": "PROVISIONING",
    "registeredContainerInstancesCount": 0,
    "pendingTasksCount": 0,
    "runningTasksCount": 0,
    "activeServicesCount": 0,
    "statistics": [],
    "settings": [
```

```

    {
      "name": "containerInsights",
      "value": "enabled"
    }
  ],
  "capacityProviders": [
    "MyCapacityProvider1",
    "MyCapacityProvider2"
  ],
  "defaultCapacityProviderStrategy": [
    {
      "capacityProvider": "MyCapacityProvider1",
      "weight": 1,
      "base": 0
    },
    {
      "capacityProvider": "MyCapacityProvider2",
      "weight": 1,
      "base": 0
    }
  ],
  "attachments": [
    {
      "id": "0fb0c8f4-6edd-4de1-9b09-17e470ee1918",
      "type": "asp",
      "status": "PRECREATED",
      "details": [
        {
          "name": "capacityProviderName",
          "value": "MyCapacityProvider1"
        },
        {
          "name": "scalingPlanName",
          "value": "ECManagedAutoScalingPlan-a1b2c3d4-5678-90ab-cdef-
EXAMPLE111111"
        }
      ]
    },
    {
      "id": "ae592060-2382-4663-9476-b015c685593c",
      "type": "asp",
      "status": "PRECREATED",
      "details": [
        {

```

```

        "name": "capacityProviderName",
        "value": "MyCapacityProvider2"
      },
      {
        "name": "scalingPlanName",
        "value": "ECSManagedAutoScalingPlan-a1b2c3d4-5678-90ab-cdef-
EXAMPLE222222"
      }
    ]
  },
  "attachmentsStatus": "UPDATE_IN_PROGRESS"
}
}

```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[集群容量提供程序](#)。

### 示例 3：创建带有多个标签的新集群

以下 `create-cluster` 示例将创建一个带有多个标签的集群。有关使用速记语法添加标签的更多信息，请参阅 CLI AWS 用户指南中的在[AWS 命令行界面中使用速记语法](#)。

```

aws ecs create-cluster \
  --cluster-name MyCluster \
  --tags key=key1,value=value1 key=key2,value=value2 key=key3,value=value3

```

输出：

```

{
  "cluster": {
    "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
    "clusterName": "MyCluster",
    "status": "ACTIVE",
    "registeredContainerInstancesCount": 0,
    "pendingTasksCount": 0,
    "runningTasksCount": 0,
    "activeServicesCount": 0,
    "statistics": [],
    "tags": [
      {
        "key": "key1",
        "value": "value1"
      }
    ],
  },
}

```

```

    {
      "key": "key2",
      "value": "value2"
    },
    {
      "key": "key3",
      "value": "value3"
    }
  ]
}
}

```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[创建集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCluster](#)中的。

## create-service

以下代码示例演示了如何使用 create-service。

### AWS CLI

#### 示例 1：使用 Fargate 任务创建服务

以下 create-service 示例演示了如何使用 Fargate 任务创建服务。

```

aws ecs create-service \
  --cluster MyCluster \
  --service-name MyService \
  --task-definition sample-fargate:1 \
  --desired-count 2 \
  --launch-type FARGATE \
  --platform-version LATEST \
  --network-configuration
  "awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp
  \
  --tags key=key1,value=value1 key=key2,value=value2 key=key3,value=value3

```

输出：

```

{
  "service": {

```

```
    "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/MyCluster/MyService",
    "serviceName": "MyService",
    "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
    "loadBalancers": [],
    "serviceRegistries": [],
    "status": "ACTIVE",
    "desiredCount": 2,
    "runningCount": 0,
    "pendingCount": 0,
    "launchType": "FARGATE",
    "platformVersion": "LATEST",
    "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/sample-fargate:1",
    "deploymentConfiguration": {
      "maximumPercent": 200,
      "minimumHealthyPercent": 100
    },
    "deployments": [
      {
        "id": "ecs-svc/1234567890123456789",
        "status": "PRIMARY",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/sample-fargate:1",
        "desiredCount": 2,
        "pendingCount": 0,
        "runningCount": 0,
        "createdAt": 1557119253.821,
        "updatedAt": 1557119253.821,
        "launchType": "FARGATE",
        "platformVersion": "1.3.0",
        "networkConfiguration": {
          "awsvpcConfiguration": {
            "subnets": [
              "subnet-12344321"
            ],
            "securityGroups": [
              "sg-12344321"
            ],
            "assignPublicIp": "ENABLED"
          }
        }
      }
    ]
  ],
```

```
    "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
    "events": [],
    "createdAt": 1557119253.821,
    "placementConstraints": [],
    "placementStrategy": [],
    "networkConfiguration": {
      "awsvpcConfiguration": {
        "subnets": [
          "subnet-12344321"
        ],
        "securityGroups": [
          "sg-12344321"
        ],
        "assignPublicIp": "ENABLED"
      }
    },
    "schedulingStrategy": "REPLICA",
    "tags": [
      {
        "key": "key1",
        "value": "value1"
      },
      {
        "key": "key2",
        "value": "value2"
      },
      {
        "key": "key3",
        "value": "value3"
      }
    ],
    "enableECSTags": false,
    "propagateTags": "NONE"
  }
}
```

## 示例 2：使用 EC2 启动类型创建服务

以下 `create-service` 示例演示如何通过使用 EC2 启动类型的任务创建名为 `ecs-simple-service` 的服务。该服务使用 `sleep360` 任务定义并维护任务的 1 个实例化。

```
aws ecs create-service \
```



```
--cluster MyCluster \  
--service-name ecs-simple-service \  
--task-definition sleep360:2 \  
--desired-count 1
```

输出：

```
{  
  "service": {  
    "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/MyCluster/ecs-  
simple-service",  
    "serviceName": "ecs-simple-service",  
    "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",  
    "loadBalancers": [],  
    "serviceRegistries": [],  
    "status": "ACTIVE",  
    "desiredCount": 1,  
    "runningCount": 0,  
    "pendingCount": 0,  
    "launchType": "EC2",  
    "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/  
sleep360:2",  
    "deploymentConfiguration": {  
      "maximumPercent": 200,  
      "minimumHealthyPercent": 100  
    },  
    "deployments": [  
      {  
        "id": "ecs-svc/1234567890123456789",  
        "status": "PRIMARY",  
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-  
definition/sleep360:2",  
        "desiredCount": 1,  
        "pendingCount": 0,  
        "runningCount": 0,  
        "createdAt": 1557206498.798,  
        "updatedAt": 1557206498.798,  
        "launchType": "EC2"  
      }  
    ],  
    "events": [],  
    "createdAt": 1557206498.798,  
    "placementConstraints": [],
```

```
    "placementStrategy": [],
    "schedulingStrategy": "REPLICA",
    "enableECSTags": false,
    "propagateTags": "NONE"
  }
}
```

### 示例 3：创建使用外部部署控制器的服务

以下 `create-service` 示例将创建使用外部部署控制器的服务。

```
aws ecs create-service \
  --cluster MyCluster \
  --service-name MyService \
  --deployment-controller type=EXTERNAL \
  --desired-count 1
```

输出：

```
{
  "service": {
    "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/MyCluster/MyService",
    "serviceName": "MyService",
    "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
    "loadBalancers": [],
    "serviceRegistries": [],
    "status": "ACTIVE",
    "desiredCount": 1,
    "runningCount": 0,
    "pendingCount": 0,
    "launchType": "EC2",
    "deploymentConfiguration": {
      "maximumPercent": 200,
      "minimumHealthyPercent": 100
    },
    "taskSets": [],
    "deployments": [],
    "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/ecs.amazonaws.com/AWSServiceRoleForECS",
    "events": [],
    "createdAt": 1557128207.101,
    "placementConstraints": [],
```

```

    "placementStrategy": [],
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
      "type": "EXTERNAL"
    },
    "enableECSManagedTags": false,
    "propagateTags": "NONE"
  }
}

```

#### 示例 4：在负载均衡器后面创建新服务

以下 `create-service` 示例将显示如何创建位于负载均衡器之后的服务。您必须将负载均衡器配置为您的容器实例所在的同一区域。此示例使用 `--cli-input-json` 选项和名为 `ecs-simple-service-elb.json` 的 JSON 输入文件，该文件包含以下内容：

```

{
  "serviceName": "ecs-simple-service-elb",
  "taskDefinition": "ecs-demo",
  "loadBalancers": [
    {
      "loadBalancerName": "EC2Contai-EcsElast-123456789012",
      "containerName": "simple-demo",
      "containerPort": 80
    }
  ],
  "desiredCount": 10,
  "role": "ecsServiceRole"
}

```

#### 命令:

```

aws ecs create-service \
  --cluster MyCluster \
  --service-name ecs-simple-service-elb \
  --cli-input-json file://ecs-simple-service-elb.json

```

#### 输出：

```

{
  "service": {

```

```
    "status": "ACTIVE",
    "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/ecs-
demo:1",
    "pendingCount": 0,
    "loadBalancers": [
      {
        "containerName": "ecs-demo",
        "containerPort": 80,
        "loadBalancerName": "EC2Contai-EcsElast-123456789012"
      }
    ],
    "roleArn": "arn:aws:iam::123456789012:role/ecsServiceRole",
    "desiredCount": 10,
    "serviceName": "ecs-simple-service-elb",
    "clusterArn": "arn:aws:ecs:<us-west-2:123456789012:cluster/MyCluster",
    "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/ecs-simple-
service-elb",
    "deployments": [
      {
        "status": "PRIMARY",
        "pendingCount": 0,
        "createdAt": 1428100239.123,
        "desiredCount": 10,
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/ecs-demo:1",
        "updatedAt": 1428100239.123,
        "id": "ecs-svc/1234567890123456789",
        "runningCount": 0
      }
    ],
    "events": [],
    "runningCount": 0
  }
}
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[创建服务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateService](#)中的。

## create-task-set

以下代码示例演示了如何使用 create-task-set。

## AWS CLI

### 创建任务集

以下create-task-set示例在使用外部部署控制器的服务中创建任务集。

```
aws ecs create-task-set \  
  --cluster MyCluster \  
  --service MyService \  
  --task-definition MyTaskDefinition:2 \  
  --network-configuration  
  "awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321]}"
```

输出：

```
{  
  "taskSet": {  
    "id": "ecs-svc/1234567890123456789",  
    "taskSetArn": "arn:aws:ecs:us-west-2:123456789012:task-set/MyCluster/  
MyService/ecs-svc/1234567890123456789",  
    "status": "ACTIVE",  
    "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/  
MyTaskDefinition:2",  
    "computedDesiredCount": 0,  
    "pendingCount": 0,  
    "runningCount": 0,  
    "createdAt": 1557128360.711,  
    "updatedAt": 1557128360.711,  
    "launchType": "EC2",  
    "networkConfiguration": {  
      "awsvpcConfiguration": {  
        "subnets": [  
          "subnet-12344321"  
        ],  
        "securityGroups": [  
          "sg-12344321"  
        ],  
        "assignPublicIp": "DISABLED"  
      }  
    },  
    "loadBalancers": [],  
    "serviceRegistries": [],  
    "scale": {
```

```
        "value": 0.0,  
        "unit": "PERCENT"  
    },  
    "stabilityStatus": "STABILIZING",  
    "stabilityStatusAt": 1557128360.711  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTaskSet](#)中的。

## delete-account-setting

以下代码示例演示了如何使用 delete-account-setting。

### AWS CLI

删除特定 IAM 用户或 IAM 角色的账户设置

以下示例delete-account-setting删除了特定 IAM 用户或 IAM 角色的账户设置。

```
aws ecs delete-account-setting \  
  --name serviceLongArnFormat \  
  --principal-arn arn:aws:iam::123456789012:user/MyUser
```

输出：

```
{  
  "setting": {  
    "name": "serviceLongArnFormat",  
    "value": "enabled",  
    "principalArn": "arn:aws:iam::123456789012:user/MyUser"  
  }  
}
```

有关更多信息，请参阅 [Amazon ECS 开发人员指南中的亚马逊资源名称 \(ARN\) 和 ID](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAccountSetting](#)中的。

## delete-attributes

以下代码示例演示了如何使用 delete-attributes。

## AWS CLI

从 Amazon ECS 资源中删除一个或多个自定义属性

以下内容stack从容器实例中delete-attributes删除名称为的属性。

```
aws ecs delete-attributes \  
  --attributes name=stack,targetId=arn:aws:ecs:us-west-2:130757420319:container-  
instance/1c3be8ed-df30-47b4-8f1e-6e68ebd01f34
```

输出：

```
{  
  "attributes": [  
    {  
      "name": "stack",  
      "targetId": "arn:aws:ecs:us-west-2:130757420319:container-  
instance/1c3be8ed-df30-47b4-8f1e-6e68ebd01f34",  
      "value": "production"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAttributes](#)中的。

## delete-capacity-provider

以下代码示例演示了如何使用 delete-capacity-provider。

### AWS CLI

示例 1：使用亚马逊资源名称 (ARN) 删除容量提供商

以下delete-capacity-provider示例通过指定容量提供商的 Amazon 资源名称 (ARN) 来删除容量提供商。可以使用命令检索 ARN 以及容量提供商删除的describe-capacity-providers状态。

```
aws ecs delete-capacity-provider \  
  --capacity-provider arn:aws:ecs:us-west-2:123456789012:capacity-provider/  
ExampleCapacityProvider
```

输出：

```
{
  "capacityProvider": {
    "capacityProviderArn": "arn:aws:ecs:us-west-2:123456789012:capacity-provider/ExampleCapacityProvider",
    "name": "ExampleCapacityProvider",
    "status": "ACTIVE",
    "autoScalingGroupProvider": {
      "autoScalingGroupArn": "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111:autoScalingGroupName/MyAutoScalingGroup",
      "managedScaling": {
        "status": "ENABLED",
        "targetCapacity": 100,
        "minimumScalingStepSize": 1,
        "maximumScalingStepSize": 10000
      },
      "managedTerminationProtection": "DISABLED"
    },
    "updateStatus": "DELETE_IN_PROGRESS",
    "tags": []
  }
}
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[集群容量提供程序](#)。

示例 2：使用名称删除容量提供商

以下delete-capacity-provider示例通过指定容量提供商的短名称来删除容量提供商。可以使用describe-capacity-providers命令检索容量提供商删除的短名称和状态。

```
aws ecs delete-capacity-provider \
  --capacity-provider ExampleCapacityProvider
```

输出：

```
{
  "capacityProvider": {
    "capacityProviderArn": "arn:aws:ecs:us-west-2:123456789012:capacity-provider/ExampleCapacityProvider",
    "name": "ExampleCapacityProvider",
```



```
    "status": "ACTIVE",
    "autoScalingGroupProvider": {
      "autoScalingGroupArn": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111:autoScalingGroupName/MyAutoScalingGroup",
      "managedScaling": {
        "status": "ENABLED",
        "targetCapacity": 100,
        "minimumScalingStepSize": 1,
        "maximumScalingStepSize": 10000
      },
      "managedTerminationProtection": "DISABLED"
    },
    "updateStatus": "DELETE_IN_PROGRESS",
    "tags": []
  }
}
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[集群容量提供程序](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCapacityProvider](#)中的。

## delete-cluster

以下代码示例演示了如何使用 delete-cluster。

### AWS CLI

#### 删除空集群

以下 delete-cluster 示例将删除指定的空集群。

```
aws ecs delete-cluster --cluster MyCluster
```

输出：

```
{
  "cluster": {
    "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
    "status": "INACTIVE",
    "clusterName": "MyCluster",
    "registeredContainerInstancesCount": 0,
  }
}
```

```
    "pendingTasksCount": 0,  
    "runningTasksCount": 0,  
    "activeServicesCount": 0  
    "statistics": [],  
    "tags": []  
  }  
}
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[删除集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCluster](#)中的。

## delete-service

以下代码示例演示了如何使用 delete-service。

### AWS CLI

#### 删除服务

以下 ecs delete-service 示例将从集群中删除指定的服务。您可以包含 --force 参数来删除服务，即使它尚未缩减至 0 个任务。

```
aws ecs delete-service --cluster MyCluster --service MyService1 --force
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[删除服务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteService](#)中的。

## delete-task-definitions

以下代码示例演示了如何使用 delete-task-definitions。

### AWS CLI

#### 删除任务定义

以下delete-task-definitions示例删除了非活动任务定义。

```
aws ecs delete-task-definitions \  
  --task-definition curltest:1
```

输出：

```
{
  "taskDefinitions": [
    {
      "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-definition/
curltest:1",
      "containerDefinitions": [
        {
          "name": "ctest",
          "image": "mreferre/eksutils",
          "cpu": 0,
          "portMappings": [],
          "essential": true,
          "entryPoint": [
            "sh",
            "-c"
          ],
          "command": [
            "curl ${ECS_CONTAINER_METADATA_URI_V4}/task"
          ],
          "environment": [],
          "mountPoints": [],
          "volumesFrom": [],
          "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
              "awslogs-create-group": "true",
              "awslogs-group": "/ecs/curltest",
              "awslogs-region": "us-east-1",
              "awslogs-stream-prefix": "ecs"
            }
          }
        }
      ]
    },
    {
      "family": "curltest",
      "taskRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
      "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
      "networkMode": "awsvpc",
      "revision": 1,
      "volumes": [],
      "status": "DELETE_IN_PROGRESS",
      "compatibilities": [
        "EC2",

```

```

        "FARGATE"
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "cpu": "256",
    "memory": "512",
    "registeredAt": "2021-09-10T12:56:24.704000+00:00",
    "deregisteredAt": "2023-03-14T15:20:59.419000+00:00",
    "registeredBy": "arn:aws:sts::123456789012:assumed-role/Admin/jdoe"
    }
],
"failures": []
}

```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的 [Amazon ECS 任务定义](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTaskDefinitions](#)中的。

## delete-task-set

以下代码示例演示了如何使用 delete-task-set。

### AWS CLI

#### 删除任务集

以下delete-task-set示例说明如何删除任务集。即使任务集尚未缩放为零，也可以添加用于删除该任务集的--force参数。

```

aws ecs delete-task-set \
  --cluster MyCluster \
  --service MyService \
  --task-set arn:aws:ecs:us-west-2:123456789012:task-set/MyCluster/MyService/ecs-
svc/1234567890123456789 \
  --force

```

输出：

```

{
  "taskSet": {
    "id": "ecs-svc/1234567890123456789",

```

```
    "taskSetArn": "arn:aws:ecs:us-west-2:123456789012:task-set/MyCluster/MyService/ecs-svc/1234567890123456789",
    "status": "DRAINING",
    "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/sample-fargate:2",
    "computedDesiredCount": 0,
    "pendingCount": 0,
    "runningCount": 0,
    "createdAt": 1557130260.276,
    "updatedAt": 1557130290.707,
    "launchType": "EC2",
    "networkConfiguration": {
      "awsvpcConfiguration": {
        "subnets": [
          "subnet-12345678"
        ],
        "securityGroups": [
          "sg-12345678"
        ],
        "assignPublicIp": "DISABLED"
      }
    },
    "loadBalancers": [],
    "serviceRegistries": [],
    "scale": {
      "value": 0.0,
      "unit": "PERCENT"
    },
    "stabilityStatus": "STABILIZING",
    "stabilityStatusAt": 1557130290.707
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTaskSet](#)中的。

## deregister-container-instance

以下代码示例演示了如何使用 deregister-container-instance。

### AWS CLI

#### 从集群中注销容器实例

以下deregister-container-instance示例从指定集群中注销容器实例。如果容器实例中仍有任务在运行，则必须在注销注册之前停止这些任务，或者使用--force选项。

```
aws ecs deregister-container-instance \  
  --cluster arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster \  
  --container-instance arn:aws:ecs:us-west-2:123456789012:container-instance/  
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \  
  --force
```

输出：

```
{  
  "containerInstance": {  
    "remainingResources": [  
      {  
        "integerValue": 1024,  
        "doubleValue": 0.0,  
        "type": "INTEGER",  
        "longValue": 0,  
        "name": "CPU"  
      },  
      {  
        "integerValue": 985,  
        "doubleValue": 0.0,  
        "type": "INTEGER",  
        "longValue": 0,  
        "name": "MEMORY"  
      },  
      {  
        "type": "STRINGSET",  
        "integerValue": 0,  
        "name": "PORTS",  
        "stringSetValue": [  
          "22",  
          "2376",  
          "2375",  
          "51678",  
          "51679"  
        ],  
        "longValue": 0,  
        "doubleValue": 0.0  
      },  
      {
```

```
        "type": "STRINGSET",
        "integerValue": 0,
        "name": "PORTS_UDP",
        "stringSetValue": [],
        "longValue": 0,
        "doubleValue": 0.0
    }
],
"agentConnected": true,
"attributes": [
    {
        "name": "ecs.capability.secrets.asm.environment-variables"
    },
    {
        "name": "com.amazonaws.ecs.capability.logging-driver.syslog"
    },
    {
        "value": "ami-01a82c3fce2c3ba58",
        "name": "ecs.ami-id"
    },
    {
        "name": "ecs.capability.secrets.asm.bootstrap.log-driver"
    },
    {
        "name": "com.amazonaws.ecs.capability.logging-driver.none"
    },
    {
        "name": "ecs.capability.ecr-endpoint"
    },
    {
        "name": "com.amazonaws.ecs.capability.logging-driver.json-file"
    },
    {
        "value": "vpc-1234567890123467",
        "name": "ecs.vpc-id"
    },
    {
        "name": "ecs.capability.execution-role-awslogs"
    },
    {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
    },
    {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
    }
]
```

```
    },
    {
      "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
    },
    {
      "name": "ecs.capability.docker-plugin.local"
    },
    {
      "name": "ecs.capability.task-eni"
    },
    {
      "name": "ecs.capability.task-cpu-mem-limit"
    },
    {
      "name": "ecs.capability.secrets.ssm.bootstrap.log-driver"
    },
    {
      "name": "com.amazonaws.ecs.capability.docker-remote-api.1.30"
    },
    {
      "name": "com.amazonaws.ecs.capability.docker-remote-api.1.31"
    },
    {
      "name": "com.amazonaws.ecs.capability.docker-remote-api.1.32"
    },
    {
      "name": "ecs.capability.execution-role-ecr-pull"
    },
    {
      "name": "ecs.capability.container-health-check"
    },
    {
      "value": "subnet-1234567890123467",
      "name": "ecs.subnet-id"
    },
    {
      "value": "us-west-2a",
      "name": "ecs.availability-zone"
    },
    {
      "value": "t2.micro",
      "name": "ecs.instance-type"
    },
    {
```



```
    "name": "com.amazonaws.ecs.capability.task-iam-role-network-host"
  },
  {
    "name": "ecs.capability.aws-appmesh"
  },
  {
    "name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
  },
  {
    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.24"
  },
  {
    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.25"
  },
  {
    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.26"
  },
  {
    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.27"
  },
  {
    "name": "com.amazonaws.ecs.capability.privileged-container"
  },
  {
    "name": "ecs.capability.container-ordering"
  },
  {
    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.28"
  },
  {
    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.29"
  },
  {
    "value": "x86_64",
    "name": "ecs.cpu-architecture"
  },
  {
    "value": "93f43776-2018.10.0",
    "name": "ecs.capability.cni-plugin-version"
  },
  {
    "name": "ecs.capability.secrets.ssm.environment-variables"
  },
  {
```

```
        "name": "ecs.capability.pid-ipc-namespace-sharing"
      },
      {
        "name": "com.amazonaws.ecs.capability.ecr-auth"
      },
      {
        "value": "linux",
        "name": "ecs.os-type"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.20"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.21"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.22"
      },
      {
        "name": "ecs.capability.task-eia"
      },
      {
        "name": "ecs.capability.private-registry-
authentication.secretsmanager"
      },
      {
        "name": "com.amazonaws.ecs.capability.task-iam-role"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.23"
      }
    ],
    "pendingTasksCount": 0,
    "tags": [],
    "containerInstanceArn": "arn:aws:ecs:us-west-2:123456789012:container-
instance/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "registeredResources": [
      {
        "integerValue": 1024,
        "doubleValue": 0.0,
        "type": "INTEGER",
        "longValue": 0,
        "name": "CPU"
      }
    ],
  },
}
```

```
    {
      "integerValue": 985,
      "doubleValue": 0.0,
      "type": "INTEGER",
      "longValue": 0,
      "name": "MEMORY"
    },
    {
      "type": "STRINGSET",
      "integerValue": 0,
      "name": "PORTS",
      "stringSetValue": [
        "22",
        "2376",
        "2375",
        "51678",
        "51679"
      ],
      "longValue": 0,
      "doubleValue": 0.0
    },
    {
      "type": "STRINGSET",
      "integerValue": 0,
      "name": "PORTS_UDP",
      "stringSetValue": [],
      "longValue": 0,
      "doubleValue": 0.0
    }
  ],
  "status": "INACTIVE",
  "registeredAt": 1557768075.681,
  "version": 4,
  "versionInfo": {
    "agentVersion": "1.27.0",
    "agentHash": "aabe65ee",
    "dockerVersion": "DockerVersion: 18.06.1-ce"
  },
  "attachments": [],
  "runningTasksCount": 0,
  "ec2InstanceId": "i-12345678901234678"
}
```

有关更多信息，请参阅《ECS 开发人员指南》中的[注销容器实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterContainerInstance](#)中的。

## deregister-task-definition

以下代码示例演示了如何使用 deregister-task-definition。

### AWS CLI

#### 取消注册任务定义

以下deregister-task-definition示例取消注册默认区域中curler任务定义的第一个修订版。

```
aws ecs deregister-task-definition --task-definition curler:1
```

请注意，在生成的输出中，任务定义状态显示INACTIVE：

```
{
  "taskDefinition": {
    "status": "INACTIVE",
    "family": "curler",
    "volumes": [],
    "taskDefinitionArn": "arn:aws:ecs:us-west-2:123456789012:task-definition/curler:1",
    "containerDefinitions": [
      {
        "environment": [],
        "name": "curler",
        "mountPoints": [],
        "image": "curl:latest",
        "cpu": 100,
        "portMappings": [],
        "entryPoint": [],
        "memory": 256,
        "command": [
          "curl -v http://example.com/"
        ],
        "essential": true,
        "volumesFrom": []
      }
    ]
  }
}
```

```
    ],  
    "revision": 1  
  }  
}
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的 [Amazon ECS 任务定义](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterTaskDefinition](#)中的。

## describe-capacity-providers

以下代码示例演示了如何使用 describe-capacity-providers。

### AWS CLI

示例 1：描述所有容量提供商

以下describe-capacity-providers示例检索有关所有容量提供商的详细信息。

```
aws ecs describe-capacity-providers
```

输出：

```
{  
  "capacityProviders": [  
    {  
      "capacityProviderArn": "arn:aws:ecs:us-west-2:123456789012:capacity-provider/MyCapacityProvider",  
      "name": "MyCapacityProvider",  
      "status": "ACTIVE",  
      "autoScalingGroupProvider": {  
        "autoScalingGroupArn": "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111:autoScalingGroupName/MyAutoScalingGroup",  
        "managedScaling": {  
          "status": "ENABLED",  
          "targetCapacity": 100,  
          "minimumScalingStepSize": 1,  
          "maximumScalingStepSize": 1000  
        },  
        "managedTerminationProtection": "ENABLED"  
      },  
    },  
  ],  
}
```

```

        "tags": []
    },
    {
        "capacityProviderArn": "arn:aws:ecs:us-west-2:123456789012:capacity-
provider/FARGATE",
        "name": "FARGATE",
        "status": "ACTIVE",
        "tags": []
    },
    {
        "capacityProviderArn": "arn:aws:ecs:us-west-2:123456789012:capacity-
provider/FARGATE_SPOT",
        "name": "FARGATE_SPOT",
        "status": "ACTIVE",
        "tags": []
    }
]
}

```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[集群容量提供程序](#)。

#### 示例 2：描述特定的容量提供商

以下describe-capacity-providers示例检索有关特定容量提供商的详细信息。使用该--include TAGS参数会将与容量提供商关联的标签添加到输出中。

```

aws ecs describe-capacity-providers \
  --capacity-providers MyCapacityProvider \
  --include TAGS

```

输出：

```

{
  "capacityProviders": [
    {
      "capacityProviderArn": "arn:aws:ecs:us-west-2:123456789012:capacity-
provider/MyCapacityProvider",
      "name": "MyCapacityProvider",
      "status": "ACTIVE",
      "autoScalingGroupProvider": {
        "autoScalingGroupArn": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111:autoScalingGroupName/MyAutoScalingGroup",

```

```
        "managedScaling": {
            "status": "ENABLED",
            "targetCapacity": 100,
            "minimumScalingStepSize": 1,
            "maximumScalingStepSize": 1000
        },
        "managedTerminationProtection": "ENABLED"
    },
    "tags": [
        {
            "key": "environment",
            "value": "production"
        }
    ]
}
]
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[集群容量提供程序](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeCapacityProviders](#) 中的。

## describe-clusters

以下代码示例演示了如何使用 describe-clusters。

### AWS CLI

#### 示例 1：描述集群

以下 describe-clusters 示例将检索指定集群的详细信息。

```
aws ecs describe-clusters \
  --cluster default
```

输出：

```
{
  "clusters": [
    {
      "status": "ACTIVE",
      "clusterName": "default",
```

```

        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 1,
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/default"
    }
],
"failures": []
}

```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的 [Amazon ECS 集群](#)。

### 示例 2：描述带有附加选项的集群

以下 `describe-clusters` 示例将指定 ATTACHMENTS 选项。它以附件的形式检索有关指定集群的详细信息以及附加到集群的资源列表。在集群中使用容量提供程序时，资源（无论是 AutoScaling 计划还是扩展策略）都将表示为 `asp` 或 `as_policy` 附件。

```

aws ecs describe-clusters \
  --include ATTACHMENTS \
  --clusters sampleCluster

```

输出：

```

{
  "clusters": [
    {
      "clusterArn": "arn:aws:ecs:af-south-1:123456789222:cluster/sampleCluster",
      "clusterName": "sampleCluster",
      "status": "ACTIVE",
      "registeredContainerInstancesCount": 0,
      "runningTasksCount": 0,
      "pendingTasksCount": 0,
      "activeServicesCount": 0,
      "statistics": [],
      "tags": [],
      "settings": [],
      "capacityProviders": [
        "sampleCapacityProvider"
      ],
      "defaultCapacityProviderStrategy": [],

```



```

    "attachments": [
      {
        "id": "a1b2c3d4-5678-901b-cdef-EXAMPLE22222",
        "type": "as_policy",
        "status": "CREATED",
        "details": [
          {
            "name": "capacityProviderName",
            "value": "sampleCapacityProvider"
          },
          {
            "name": "scalingPolicyName",
            "value": "ECManagedAutoScalingPolicy-3048e262-
fe39-4eaf-826d-6f975d303188"
          }
        ]
      }
    ],
    "attachmentsStatus": "UPDATE_COMPLETE"
  }
],
"failures": []
}

```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的 [Amazon ECS 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeClusters](#)中的。

## describe-container-instances

以下代码示例演示了如何使用 describe-container-instances。

### AWS CLI

#### 描述容器实例

以下describe-container-instances示例使用容器实例 UUID 作为标识符检索update集群中容器实例的详细信息。

```

aws ecs describe-container-instances \
  --cluster update \
  --container-instances a1b2c3d4-5678-90ab-cdef-11111EXAMPLE

```

输出：

```
{
  "failures": [],
  "containerInstances": [
    {
      "status": "ACTIVE",
      "registeredResources": [
        {
          "integerValue": 2048,
          "longValue": 0,
          "type": "INTEGER",
          "name": "CPU",
          "doubleValue": 0.0
        },
        {
          "integerValue": 3955,
          "longValue": 0,
          "type": "INTEGER",
          "name": "MEMORY",
          "doubleValue": 0.0
        },
        {
          "name": "PORTS",
          "longValue": 0,
          "doubleValue": 0.0,
          "stringSetValue": [
            "22",
            "2376",
            "2375",
            "51678"
          ],
          "type": "STRINGSET",
          "integerValue": 0
        }
      ],
      "ec2InstanceId": "i-A1B2C3D4",
      "agentConnected": true,
      "containerInstanceArn": "arn:aws:ecs:us-west-2:123456789012:container-
instance/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "pendingTasksCount": 0,
      "remainingResources": [
        {
          "integerValue": 2048,
```

```
        "longValue": 0,
        "type": "INTEGER",
        "name": "CPU",
        "doubleValue": 0.0
    },
    {
        "integerValue": 3955,
        "longValue": 0,
        "type": "INTEGER",
        "name": "MEMORY",
        "doubleValue": 0.0
    },
    {
        "name": "PORTS",
        "longValue": 0,
        "doubleValue": 0.0,
        "stringSetValue": [
            "22",
            "2376",
            "2375",
            "51678"
        ],
        "type": "STRINGSET",
        "integerValue": 0
    }
],
"runningTasksCount": 0,
"versionInfo": {
    "agentVersion": "1.0.0",
    "agentHash": "4023248",
    "dockerVersion": "DockerVersion: 1.5.0"
}
}
]
```

有关更多信息，请参阅 [Amazon ECS 开发人员指南中的亚马逊 ECS 容器实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeContainerInstances](#) 中的。

## describe-services

以下代码示例演示了如何使用 describe-services。

## AWS CLI

### 描述服务

以下describe-services示例检索默认集群中my-http-service服务的详细信息。

```
aws ecs describe-services --services my-http-service
```

输出：

```
{
  "services": [
    {
      "status": "ACTIVE",
      "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
amazon-ecs-sample:1",
      "pendingCount": 0,
      "loadBalancers": [],
      "desiredCount": 10,
      "createdAt": 1466801808.595,
      "serviceName": "my-http-service",
      "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/default",
      "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/my-http-
service",
      "deployments": [
        {
          "status": "PRIMARY",
          "pendingCount": 0,
          "createdAt": 1466801808.595,
          "desiredCount": 10,
          "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/amazon-ecs-sample:1",
          "updatedAt": 1428326312.703,
          "id": "ecs-svc/1234567890123456789",
          "runningCount": 10
        }
      ],
      "events": [
        {
          "message": "(service my-http-service) has reached a steady
state.",
          "id": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
          "createdAt": 1466801812.435
        }
      ]
    }
  ]
}
```

```
    }
  ],
  "runningCount": 10
}
],
"failures": []
}
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[服务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeServices](#)中的。

## describe-task-definition

以下代码示例演示了如何使用 describe-task-definition。

### AWS CLI

#### 描述任务定义

以下describe-task-definition示例检索任务定义的详细信息。

```
aws ecs describe-task-definition \
  --task-definition hello_world:8
```

输出：

```
{
  "tasks": [
    {
      "attachments": [
        {
          "id": "17f3dff6-a9e9-4d83-99a9-7eb5193c2634",
          "type": "ElasticNetworkInterface",
          "status": "ATTACHED",
          "details": [
            {
              "name": "subnetId",
              "value": "subnet-0d0eab1bb38d5ca64"
            },
            {
              "name": "networkInterfaceId",
              "value": "eni-0d542ffb4a12aa6d9"
            }
          ]
        }
      ]
    }
  ]
}
```

```

    },
    {
      "name": "macAddress",
      "value": "0e:6d:18:f6:2d:29"
    },
    {
      "name": "privateDnsName",
      "value": "ip-10-0-1-170.ec2.internal"
    },
    {
      "name": "privateIPv4Address",
      "value": "10.0.1.170"
    }
  ]
}
],
"attributes": [
  {
    "name": "ecs.cpu-architecture",
    "value": "x86_64"
  }
],
"availabilityZone": "us-east-1b",
"clusterArn": "arn:aws:ecs:us-east-1:053534965804:cluster/fargate-
cluster",
"connectivity": "CONNECTED",
"connectivityAt": "2023-11-28T11:10:52.907000-05:00",
"containers": [
  {
    "containerArn": "arn:aws:ecs:us-east-1:053534965804:container/
fargate-cluster/
c524291ae4154100b601a543108b193a/772c4784-92ae-414e-8df2-03d3358e39fa",
    "taskArn": "arn:aws:ecs:us-east-1:053534965804:task/fargate-
cluster/c524291ae4154100b601a543108b193a",
    "name": "web",
    "image": "nginx",
    "imageDigest":
"sha256:10d1f5b58f74683ad34eb29287e07dab1e90f10af243f151bb50aa5dbb4d62ee",
    "runtimeId": "c524291ae4154100b601a543108b193a-265927825",
    "lastStatus": "RUNNING",
    "networkBindings": [],
    "networkInterfaces": [
      {
        "attachmentId": "17f3dff6-a9e9-4d83-99a9-7eb5193c2634",

```

```
        "privateIpv4Address": "10.0.1.170"
      }
    ],
    "healthStatus": "HEALTHY",
    "cpu": "99",
    "memory": "100"
  },
  {
    "containerArn": "arn:aws:ecs:us-east-1:053534965804:container/
fargate-cluster/c524291ae4154100b601a543108b193a/c051a779-40d2-48ca-
ad5e-6ec875ceb610",
    "taskArn": "arn:aws:ecs:us-east-1:053534965804:task/fargate-
cluster/c524291ae4154100b601a543108b193a",
    "name": "aws-guardduty-agent-FvWGoDU",
    "imageDigest":
"sha256:359b8b014e5076c625daa1056090e522631587a7afa3b2e055edda6bd1141017",
    "runtimeId": "c524291ae4154100b601a543108b193a-505093495",
    "lastStatus": "RUNNING",
    "networkBindings": [],
    "networkInterfaces": [
      {
        "attachmentId": "17f3dff6-a9e9-4d83-99a9-7eb5193c2634",
        "privateIpv4Address": "10.0.1.170"
      }
    ],
    "healthStatus": "UNKNOWN"
  }
],
"cpu": "256",
"createdAt": "2023-11-28T11:10:49.299000-05:00",
"desiredStatus": "RUNNING",
"enableExecuteCommand": false,
"group": "family:webserver",
"healthStatus": "HEALTHY",
"lastStatus": "RUNNING",
"launchType": "FARGATE",
"memory": "512"
"platformVersion": "1.4.0",
"platformFamily": "Linux",
"pullStartedAt": "2023-11-28T11:10:59.773000-05:00",
"pullStoppedAt": "2023-11-28T11:11:12.624000-05:00",
"startedAt": "2023-11-28T11:11:20.316000-05:00",
"tags": [],
```

```

        "taskArn": "arn:aws:ecs:us-east-1:053534965804:task/fargate-cluster/
c524291ae4154100b601a543108b193a",
        "taskDefinitionArn": "arn:aws:ecs:us-east-1:053534965804:task-
definition/webserver:5",
        "version": 4,
        "ephemeralStorage": {
            "sizeInGiB": 20
        }
    },
    "failures": []
}

```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的 [Amazon ECS 任务定义](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTaskDefinition](#)中的。

## describe-task-sets

以下代码示例演示了如何使用 describe-task-sets。

### AWS CLI

#### 描述任务集

以下describe-task-sets示例描述了使用外部部署程序的服务中的任务集。

```

aws ecs describe-task-sets \
  --cluster MyCluster \
  --service MyService \
  --task-sets arn:aws:ecs:us-west-2:123456789012:task-set/MyCluster/MyService/ecs-
svc/1234567890123456789

```

输出：

```

{
  "taskSets": [
    {
      "id": "ecs-svc/1234567890123456789",
      "taskSetArn": "arn:aws:ecs:us-west-2:123456789012:task-set/MyCluster/
MyService/ecs-svc/1234567890123456789",
      "status": "ACTIVE",

```



```
    "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/sample-fargate:2",
    "computedDesiredCount": 0,
    "pendingCount": 0,
    "runningCount": 0,
    "createdAt": 1557207715.195,
    "updatedAt": 1557207740.014,
    "launchType": "EC2",
    "networkConfiguration": {
      "awsvpcConfiguration": {
        "subnets": [
          "subnet-12344321"
        ],
        "securityGroups": [
          "sg-1234431"
        ],
        "assignPublicIp": "DISABLED"
      }
    },
    "loadBalancers": [],
    "serviceRegistries": [],
    "scale": {
      "value": 0.0,
      "unit": "PERCENT"
    },
    "stabilityStatus": "STEADY_STATE",
    "stabilityStatusAt": 1557207740.014
  }
],
"failures": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTaskSets](#)中的。

## describe-tasks

以下代码示例演示了如何使用 describe-tasks。

### AWS CLI

#### 示例 1：描述单个任务

以下 `describe-tasks` 示例将检索集群中任务的详细信息。您可以使用任务的 ID 或完整 ARN 来指定任务。此示例使用任务的完整 ARN。

```
aws ecs describe-tasks \  
  --cluster MyCluster \  
  --tasks arn:aws:ecs:us-east-1:123456789012:task/  
MyCluster/4d590253bb114126b7afa7b58EXAMPLE
```

输出：

```
{  
  "tasks": [  
    {  
      "attachments": [],  
      "attributes": [  
        {  
          "name": "ecs.cpu-architecture",  
          "value": "x86_64"  
        }  
      ],  
      "availabilityZone": "us-east-1b",  
      "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",  
      "connectivity": "CONNECTED",  
      "connectivityAt": "2021-08-11T12:21:26.681000-04:00",  
      "containerInstanceArn": "arn:aws:ecs:us-east-1:123456789012:container-  
instance/test/025c7e2c5e054a6790a29fc1fEXAMPLE",  
      "containers": [  
        {  
          "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/  
MyCluster/4d590253bb114126b7afa7b58eea9221/a992d1cc-ea46-474a-b6e8-24688EXAMPLE",  
          "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/  
MyCluster/4d590253bb114126b7afa7b58EXAMPLE",  
          "name": "simple-app",  
          "image": "httpd:2.4",  
          "runtimeId":  
"91251eed27db90006ad67b1a08187290869f216557717dd5c39b37c94EXAMPLE",  
          "lastStatus": "RUNNING",  
          "networkBindings": [  
            {  
              "bindIP": "0.0.0.0",  
              "containerPort": 80,  
              "hostPort": 80,  
              "protocol": "tcp"  
            }  
          ]  
        }  
      ]  
    }  
  ]  
}
```

```

        }
      ],
      "networkInterfaces": [],
      "healthStatus": "UNKNOWN",
      "cpu": "10",
      "memory": "300"
    }
  ],
  "cpu": "10",
  "createdAt": "2021-08-11T12:21:26.681000-04:00",
  "desiredStatus": "RUNNING",
  "enableExecuteCommand": false,
  "group": "service:testupdate",
  "healthStatus": "UNKNOWN",
  "lastStatus": "RUNNING",
  "launchType": "EC2",
  "memory": "300",
  "overrides": {
    "containerOverrides": [
      {
        "name": "simple-app"
      }
    ],
    "inferenceAcceleratorOverrides": []
  },
  "pullStartedAt": "2021-08-11T12:21:28.234000-04:00",
  "pullStoppedAt": "2021-08-11T12:21:33.793000-04:00",
  "startedAt": "2021-08-11T12:21:34.945000-04:00",
  "startedBy": "ecs-svc/968695068243EXAMPLE",
  "tags": [],
  "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/MyCluster/4d590253bb114126b7afa7b58eea9221",
  "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-definition/console-sample-app-static2:1",
  "version": 2
}
],
"failures": []
}

```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的 [Amazon ECS 任务定义](#)。

示例 2：描述多个任务

以下 `describe-tasks` 示例将检索集群中多个任务的详细信息。您可以使用任务的 ID 或完整 ARN 来指定任务。此示例使用任务的完整 ID。

```
aws ecs describe-tasks \  
  --cluster MyCluster \  
  --tasks "74de0355a10a4f979ac495c14EXAMPLE" "d789e94343414c25b9f6bd59eEXAMPLE"
```

输出：

```
{  
  "tasks": [  
    {  
      "attachments": [  
        {  
          "id": "d9e7735a-16aa-4128-bc7a-b2d51EXAMPLE",  
          "type": "ElasticNetworkInterface",  
          "status": "ATTACHED",  
          "details": [  
            {  
              "name": "subnetId",  
              "value": "subnet-0d0eab1bb3EXAMPLE"  
            },  
            {  
              "name": "networkInterfaceId",  
              "value": "eni-0fa40520aeEXAMPLE"  
            },  
            {  
              "name": "macAddress",  
              "value": "0e:89:76:28:07:b3"  
            },  
            {  
              "name": "privateDnsName",  
              "value": "ip-10-0-1-184.ec2.internal"  
            },  
            {  
              "name": "privateIPv4Address",  
              "value": "10.0.1.184"  
            }  
          ]  
        }  
      ],  
      "attributes": [  
        {
```

```
        "name": "ecs.cpu-architecture",
        "value": "x86_64"
    }
],
"availabilityZone": "us-east-1b",
"clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
"connectivity": "CONNECTED",
"connectivityAt": "2021-12-20T12:13:37.875000-05:00",
"containers": [
    {
        "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/
MyCluster/74de0355a10a4f979ac495c14EXAMPLE/aad3ba00-83b3-4dac-84d4-11f8cEXAMPLE",
        "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/
MyCluster/74de0355a10a4f979ac495c14EXAMPLE",
        "name": "web",
        "image": "nginx",
        "runtimeId": "74de0355a10a4f979ac495c14EXAMPLE-265927825",
        "lastStatus": "RUNNING",
        "networkBindings": [],
        "networkInterfaces": [
            {
                "attachmentId": "d9e7735a-16aa-4128-bc7a-b2d51EXAMPLE",
                "privateIpv4Address": "10.0.1.184"
            }
        ],
        "healthStatus": "UNKNOWN",
        "cpu": "99",
        "memory": "100"
    }
],
"cpu": "256",
"createdAt": "2021-12-20T12:13:20.226000-05:00",
"desiredStatus": "RUNNING",
"enableExecuteCommand": false,
"group": "service:tdsevicetag",
"healthStatus": "UNKNOWN",
"lastStatus": "RUNNING",
"launchType": "FARGATE",
"memory": "512",
"overrides": {
    "containerOverrides": [
        {
            "name": "web"
        }
    ]
}
```

```
    ],
    "inferenceAcceleratorOverrides": [],
  },
  "platformVersion": "1.4.0",
  "platformFamily": "Linux",
  "pullStartedAt": "2021-12-20T12:13:42.665000-05:00",
  "pullStoppedAt": "2021-12-20T12:13:46.543000-05:00",
  "startedAt": "2021-12-20T12:13:48.086000-05:00",
  "startedBy": "ecs-svc/988401040018EXAMPLE",
  "tags": [],
  "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/MyCluster/74de0355a10a4f979ac495c14EXAMPLE",
  "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-definition/webserver:2",
  "version": 3,
  "ephemeralStorage": {
    "sizeInGiB": 20
  }
},
{
  "attachments": [
    {
      "id": "214eb5a9-45cd-4bf8-87bc-57fefEXAMPLE",
      "type": "ElasticNetworkInterface",
      "status": "ATTACHED",
      "details": [
        {
          "name": "subnetId",
          "value": "subnet-0d0eab1bb3EXAMPLE"
        },
        {
          "name": "networkInterfaceId",
          "value": "eni-064c7766daEXAMPLE"
        },
        {
          "name": "macAddress",
          "value": "0e:76:83:01:17:a9"
        },
        {
          "name": "privateDnsName",
          "value": "ip-10-0-1-41.ec2.internal"
        },
        {
          "name": "privateIPv4Address",
```

```
        "value": "10.0.1.41"
      }
    ]
  },
  "attributes": [
    {
      "name": "ecs.cpu-architecture",
      "value": "x86_64"
    }
  ],
  "availabilityZone": "us-east-1b",
  "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
  "connectivity": "CONNECTED",
  "connectivityAt": "2021-12-20T12:13:35.243000-05:00",
  "containers": [
    {
      "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/MyCluster/d789e94343414c25b9f6bd59eEXAMPLE/9afef792-609b-43a5-bb6a-3efdbEXAMPLE",
      "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/MyCluster/d789e94343414c25b9f6bd59eEXAMPLE",
      "name": "web",
      "image": "nginx",
      "runtimeId": "d789e94343414c25b9f6bd59eEXAMPLE-265927825",
      "lastStatus": "RUNNING",
      "networkBindings": [],
      "networkInterfaces": [
        {
          "attachmentId": "214eb5a9-45cd-4bf8-87bc-57fefEXAMPLE",
          "privateIpv4Address": "10.0.1.41"
        }
      ],
      "healthStatus": "UNKNOWN",
      "cpu": "99",
      "memory": "100"
    }
  ],
  "cpu": "256",
  "createdAt": "2021-12-20T12:13:20.226000-05:00",
  "desiredStatus": "RUNNING",
  "enableExecuteCommand": false,
  "group": "service:tdsevicetag",
  "healthStatus": "UNKNOWN",
  "lastStatus": "RUNNING",
```

```
    "launchType": "FARGATE",
    "memory": "512",
    "overrides": {
      "containerOverrides": [
        {
          "name": "web"
        }
      ],
      "inferenceAcceleratorOverrides": []
    },
    "platformVersion": "1.4.0",
    "platformFamily": "Linux",
    "pullStartedAt": "2021-12-20T12:13:44.611000-05:00",
    "pullStoppedAt": "2021-12-20T12:13:48.251000-05:00",
    "startedAt": "2021-12-20T12:13:49.326000-05:00",
    "startedBy": "ecs-svc/988401040018EXAMPLE",
    "tags": [],
    "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/MyCluster/d789e94343414c25b9f6bd59eEXAMPLE",
    "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-definition/webserver:2",
    "version": 3,
    "ephemeralStorage": {
      "sizeInGiB": 20
    }
  },
  "failures": []
}
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的 [Amazon ECS 任务定义](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTasks](#)中的。

## execute-command

以下代码示例演示了如何使用 execute-command。

### AWS CLI

运行交互式 /bin/sh 命令



以下execute-command示例对以 ID 为的任务命名的 MyContainer 容器运行交互式 /bin/sh 命令。arn:aws:ecs:us-east-1:123456789012:task/MyCluster/d789e94343414c25b9f6bd59eEXAMPLE

```
aws ecs execute-command \  
  --cluster MyCluster \  
  --task arn:aws:ecs:us-east-1:123456789012:task/MyCluster/  
d789e94343414c25b9f6bd59eEXAMPLE \  
  --container MyContainer \  
  --interactive \  
  --command "/bin/sh"
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊 ECS 开发人员指南](#)》中的“[使用 Amazon ECS Exec 进行调试](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ExecuteCommand](#)中的。

## list-account-settings

以下代码示例演示了如何使用 list-account-settings。

### AWS CLI

示例 1：查看账户的账户设置

以下list-account-settings示例显示了账户的有效账户设置。

```
aws ecs list-account-settings --effective-settings
```

输出：

```
{  
  "settings": [  
    {  
      "name": "containerInstanceLongArnFormat",  
      "value": "enabled",  
      "principalArn": "arn:aws:iam::123456789012:root"  
    },  
    {  
      "name": "serviceLongArnFormat",  
      "value": "enabled",  
    }  
  ]  
}
```

```

        "principalArn": "arn:aws:iam::123456789012:root"
      },
      {
        "name": "taskLongArnFormat",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:root"
      }
    ]
  }
}

```

## 示例 2：查看特定 IAM 用户或 IAM 角色的账户设置

以下 `list-account-settings` 示例显示了指定 IAM 用户或 IAM 角色的账户设置。

```
aws ecs list-account-settings --principal-arn arn:aws:iam::123456789012:user/MyUser
```

输出：

```

{
  "settings": [
    {
      "name": "serviceLongArnFormat",
      "value": "enabled",
      "principalArn": "arn:aws:iam::123456789012:user/MyUser"
    }
  ]
}

```

有关更多信息，请参阅 [Amazon ECS 开发人员指南中的亚马逊资源名称 \(ARN\) 和 ID](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAccountSettings](#) 中的。

## list-attributes

以下代码示例演示了如何使用 `list-attributes`。

### AWS CLI

列出包含特定属性的容器实例

以下示例列出了默认集群中具有该 `stack=production` 属性的容器实例的属性。

```
aws ecs list-attributes \
```

```
--target-type container-instance \  
--attribute-name stack \  
--attribute-value production \  
--cluster default
```

输出：

```
{  
  "attributes": [  
    {  
      "name": "stack",  
      "targetId": "arn:aws:ecs:us-west-2:130757420319:container-  
instance/1c3be8ed-df30-47b4-8f1e-6e68ebd01f34",  
      "value": "production"  
    }  
  ]  
}
```

有关更多信息，请参阅 [《亚马逊 ECS 开发人员指南》](#) 中的 [Amazon ECS 容器代理配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAttributes](#) 中的。

## list-clusters

以下代码示例演示了如何使用 list-clusters。

### AWS CLI

列出您的可用集群

以下 list-clusters 示例将列出所有可用的集群。

```
aws ecs list-clusters
```

输出：

```
{  
  "clusterArns": [  
    "arn:aws:ecs:us-west-2:123456789012:cluster/MyECSCluster1",  
    "arn:aws:ecs:us-west-2:123456789012:cluster/AnotherECSCluster"  
  ]  
}
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的 [Amazon ECS 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListClusters](#)中的。

## list-container-instances

以下代码示例演示了如何使用 `list-container-instances`。

### AWS CLI

列出集群中的容器实例

以下`list-container-instances`示例列出了集群中所有可用的容器实例。

```
aws ecs list-container-instances --cluster MyCluster
```

输出：

```
{
  "containerInstanceArns": [
    "arn:aws:ecs:us-west-2:123456789012:container-instance/MyCluster/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "arn:aws:ecs:us-west-2:123456789012:container-instance/MyCluster/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE"
  ]
}
```

有关更多信息，请参阅 [Amazon ECS 开发人员指南中的亚马逊 ECS 容器实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListContainerInstances](#)中的。

## list-services-by-namespace

以下代码示例演示了如何使用 `list-services-by-namespace`。

### AWS CLI

列出命名空间中的服务

以下`list-services-by-namespace`示例列出了在您的默认区域中为指定命名空间配置的所有服务。

```
aws ecs list-services-by-namespace \  
  --namespace service-connect
```

输出：

```
{  
  "serviceArns": [  
    "arn:aws:ecs:us-west-2:123456789012:service/MyCluster/MyService",  
    "arn:aws:ecs:us-west-2:123456789012:service/tutorial/service-connect-nginx-  
service"  
  ]  
}
```

有关更多信息，请参阅 Amazon ECS 开发人员指南中的 [Service Connect](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListServicesByNamespace](#) 中的。

## list-services

以下代码示例演示了如何使用 `list-services`。

### AWS CLI

列出集群中的服务

以下 `list-services` 示例演示如何列出集群中运行的服务。

```
aws ecs list-services --cluster MyCluster
```

输出：

```
{  
  "serviceArns": [  
    "arn:aws:ecs:us-west-2:123456789012:service/MyCluster/MyService"  
  ]  
}
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的 [服务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListServices](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出资源的标签

以下 `list-tags-for-resource` 示例列出了特定集群的标签。

```
aws ecs list-tags-for-resource \
  --resource-arn arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster
```

输出：

```
{
  "tags": [
    {
      "key": "key1",
      "value": "value1"
    },
    {
      "key": "key2",
      "value": "value2"
    },
    {
      "key": "key3",
      "value": "value3"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## list-task-definition-families

以下代码示例演示了如何使用 `list-task-definition-families`。

### AWS CLI

示例 1：列出已注册的任务定义系列

以下`list-task-definition-families`示例列出了所有已注册的任务定义系列。

```
aws ecs list-task-definition-families
```

输出：

```
{
  "families": [
    "node-js-app",
    "web-timer",
    "hpcc",
    "hpcc-c4-8xlarge"
  ]
}
```

示例 2：筛选已注册的任务定义系列

以下`list-task-definition-families`示例列出了以“hpcc”开头的任务定义修订版。

```
aws ecs list-task-definition-families --family-prefix hpcc
```

输出：

```
{
  "families": [
    "hpcc",
    "hpcc-c4-8xlarge"
  ]
}
```

有关更多信息，请参阅 Amazon ECS 开发人员指南中的[任务定义参数](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTaskDefinitionFamilies](#)中的。

## list-task-definitions

以下代码示例演示了如何使用 `list-task-definitions`。

### AWS CLI

示例 1：列出已注册的任务定义

以下 `list-task-definitions` 示例列出了所有已注册的任务定义。

```
aws ecs list-task-definitions
```

输出：

```
{
  "taskDefinitionArns": [
    "arn:aws:ecs:us-west-2:123456789012:task-definition/sleep300:2",
    "arn:aws:ecs:us-west-2:123456789012:task-definition/sleep360:1",
    "arn:aws:ecs:us-west-2:123456789012:task-definition/wordpress:3",
    "arn:aws:ecs:us-west-2:123456789012:task-definition/wordpress:4",
    "arn:aws:ecs:us-west-2:123456789012:task-definition/wordpress:5",
    "arn:aws:ecs:us-west-2:123456789012:task-definition/wordpress:6"
  ]
}
```

示例 2：列出家庭中已注册的任务定义

以下 `list-task-definitions` 示例列出了指定系列的任务定义修订版。

```
aws ecs list-task-definitions --family-prefix wordpress
```

输出：

```
{
  "taskDefinitionArns": [
    "arn:aws:ecs:us-west-2:123456789012:task-definition/wordpress:3",
    "arn:aws:ecs:us-west-2:123456789012:task-definition/wordpress:4",
    "arn:aws:ecs:us-west-2:123456789012:task-definition/wordpress:5",
    "arn:aws:ecs:us-west-2:123456789012:task-definition/wordpress:6"
  ]
}
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的 [Amazon ECS 任务定义](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTaskDefinitions](#) 中的。

## list-tasks

以下代码示例演示了如何使用 `list-tasks`。



## AWS CLI

### 示例 1：列出集群中的任务

以下 `list-tasks` 示例将列出集群中的所有任务。

```
aws ecs list-tasks --cluster default
```

输出：

```
{
  "taskArns": [
    "arn:aws:ecs:us-west-2:123456789012:task/a1b2c3d4-5678-90ab-
cdef-11111EXAMPLE",
    "arn:aws:ecs:us-west-2:123456789012:task/a1b2c3d4-5678-90ab-
cdef-22222EXAMPLE"
  ]
}
```

### 示例 2：列出特定容器实例上的任务

以下 `list-tasks` 示例将使用容器实例 UUID 作为筛选器列出容器实例上的任务。

```
aws ecs list-tasks --cluster default --container-instance a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE
```

输出：

```
{
  "taskArns": [
    "arn:aws:ecs:us-west-2:123456789012:task/a1b2c3d4-5678-90ab-
cdef-44444EXAMPLE"
  ]
}
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的 [Amazon ECS 任务定义](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTasks](#) 中的。

## **put-account-setting-default**

以下代码示例演示了如何使用 `put-account-setting-default`。

## AWS CLI

### 修改默认账户设置

以下put-account-setting-default示例修改了您账户中所有 IAM 用户或角色的默认账户设置。除非 IAM 用户或角色为自己明确覆盖这些设置，否则这些更改将适用于整个 AWS 账户。

```
aws ecs put-account-setting-default --name serviceLongArnFormat --value enabled
```

输出：

```
{
  "setting": {
    "name": "serviceLongArnFormat",
    "value": "enabled",
    "principalArn": "arn:aws:iam::123456789012:root"
  }
}
```

有关更多信息，请参阅 [Amazon ECS 开发人员指南中的亚马逊资源名称 \(ARN\) 和 ID](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutAccountSettingDefault](#)中的。

## put-account-setting

以下代码示例演示了如何使用 put-account-setting。

## AWS CLI

### 修改您的 IAM 用户账户的账户设置

以下put-account-setting示例为您的 IAM 用户serviceLongArnFormat账户启用账户设置。

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled
```

输出：

```
{
  "setting": {
```

```
    "name": "serviceLongArnFormat",
    "value": "enabled",
    "principalArn": "arn:aws:iam::130757420319:user/your_username"
  }
}
```

有关更多信息，请参阅 Amazon ECS 开发人员指南中的[修改账户设置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutAccountSetting](#)中的。

## put-account-settings

以下代码示例演示了如何使用 put-account-settings。

### AWS CLI

修改 IAM 用户或 IAM 角色的账户设置

以下put-account-setting示例修改了指定 IAM 用户或 IAM 角色的账户设置。

```
aws ecs put-account-setting \
  --name serviceLongArnFormat \
  --value enabled \
  --principal-arn arn:aws:iam::123456789012:user/MyUser
```

输出：

```
{
  "setting": {
    "name": "serviceLongArnFormat",
    "value": "enabled",
    "principalArn": "arn:aws:iam::123456789012:user/MyUser"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutAccountSettings](#)中的。

## put-attributes

以下代码示例演示了如何使用 put-attributes。

## AWS CLI

### 创建属性并将其与 Amazon ECS 资源关联

以下内容`put-attributes`将名称为 `stack` 且值为 `production` 的属性应用于容器实例。

```
aws ecs put-attributes \  
  --attributes name=stack,value=production,targetId=arn:aws:ecs:us-  
west-2:130757420319:container-instance/1c3be8ed-df30-47b4-8f1e-6e68ebd01f34
```

输出：

```
{  
  "attributes": [  
    {  
      "name": "stack",  
      "targetId": "arn:aws:ecs:us-west-2:130757420319:container-  
instance/1c3be8ed-df30-47b4-8f1e-6e68ebd01f34",  
      "value": "production"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutAttributes](#)中的。

## put-cluster-capacity-providers

以下代码示例演示了如何使用 `put-cluster-capacity-providers`。

### AWS CLI

#### 示例 1：向集群添加现有容量提供商

以下`put-cluster-capacity-providers`示例将现有容量提供程序添加到集群。使用 `create-capacity-provider` 命令创建容量提供程序。该`describe-clusters`命令用于描述当前的容量提供商以及与集群相关的默认容量提供者策略。向集群添加新的容量提供商时，除了要与集群关联的新容量提供商外，还必须指定所有现有的容量提供商。您还必须指定要与集群关联的默认容量提供者策略。在此示例中，MyCluster集群具有与之关联的MyCapacityProvider1容量提供商，您希望添加MyCapacityProvider2容量提供程序并将其包含在默认容量提供程序策略中，以便任务在两个容量提供商之间均匀分配。

```
aws ecs put-cluster-capacity-providers \  
  --cluster MyCluster \  
  --capacity-providers MyCapacityProvider1 MyCapacityProvider2 \  
  --default-capacity-provider-strategy  
capacityProvider=MyCapacityProvider1,weight=1  
capacityProvider=MyCapacityProvider2,weight=1
```

输出：

```
{  
  "cluster": {  
    "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",  
    "clusterName": "MyCluster",  
    "status": "ACTIVE",  
    "registeredContainerInstancesCount": 0,  
    "runningTasksCount": 0,  
    "pendingTasksCount": 0,  
    "activeServicesCount": 0,  
    "statistics": [],  
    "tags": [],  
    "settings": [  
      {  
        "name": "containerInsights",  
        "value": "enabled"  
      }  
    ],  
    "capacityProviders": [  
      "MyCapacityProvider1",  
      "MyCapacityProvider2"  
    ],  
    "defaultCapacityProviderStrategy": [  
      {  
        "capacityProvider": "MyCapacityProvider1",  
        "weight": 1,  
        "base": 0  
      },  
      {  
        "capacityProvider": "MyCapacityProvider2",  
        "weight": 1,  
        "base": 0  
      }  
    ],  
    "attachments": [  

```

```

    {
      "id": "0fb0c8f4-6edd-4de1-9b09-17e470ee1918",
      "type": "as_policy",
      "status": "ACTIVE",
      "details": [
        {
          "name": "capacityProviderName",
          "value": "MyCapacityProvider1"
        },
        {
          "name": "scalingPolicyName",
          "value": "ECSManagedAutoScalingPolicy-a1b2c3d4-5678-90ab-
cdef-EXAMPLE111111"
        }
      ]
    },
    {
      "id": "ae592060-2382-4663-9476-b015c685593c",
      "type": "as_policy",
      "status": "ACTIVE",
      "details": [
        {
          "name": "capacityProviderName",
          "value": "MyCapacityProvider2"
        },
        {
          "name": "scalingPolicyName",
          "value": "ECSManagedAutoScalingPolicy-a1b2c3d4-5678-90ab-
cdef-EXAMPLE222222"
        }
      ]
    }
  ],
  "attachmentsStatus": "UPDATE_IN_PROGRESS"
}

```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[集群容量提供程序](#)。

## 示例 2：从集群中移除容量提供商

以下 `put-cluster-capacity-providers` 示例从集群中移除容量提供商。该 `describe-clusters` 命令用于描述与集群关联的当前容量提供商。从集群中移除容量提供商时，必须指定

要与集群保持关联的容量提供商，以及要与集群关联的默认容量提供者策略。在此示例中，集群具有与其关联的MyCapacityProvider1和MyCapacityProvider2容量提供程序，您想移除MyCapacityProvider2容量提供商，因此您只能MyCapacityProvider1在命令中以及更新的默认容量提供程序策略中指定。

```
aws ecs put-cluster-capacity-providers \  
  --cluster MyCluster \  
  --capacity-providers MyCapacityProvider1 \  
  --default-capacity-provider-strategy  
  capacityProvider=MyCapacityProvider1,weight=1,base=0
```

输出：

```
{  
  "cluster": {  
    "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",  
    "clusterName": "MyCluster",  
    "status": "ACTIVE",  
    "registeredContainerInstancesCount": 0,  
    "runningTasksCount": 0,  
    "pendingTasksCount": 0,  
    "activeServicesCount": 0,  
    "statistics": [],  
    "tags": [],  
    "settings": [  
      {  
        "name": "containerInsights",  
        "value": "enabled"  
      }  
    ],  
    "capacityProviders": [  
      "MyCapacityProvider1"  
    ],  
    "defaultCapacityProviderStrategy": [  
      "capacityProvider": "MyCapacityProvider1",  
      "weight": 1,  
      "base": 0  
    ],  
    "attachments": [  
      {  
        "id": "0fb0c8f4-6edd-4de1-9b09-17e470ee1918",  
        "type": "as_policy",
```

```

        "status": "ACTIVE",
        "details": [
            {
                "name": "capacityProviderName",
                "value": "MyCapacityProvider1"
            },
            {
                "name": "scalingPolicyName",
                "value": "ECManagedAutoScalingPolicy-a1b2c3d4-5678-90ab-
cdef-EXAMPLE11111"
            }
        ]
    },
    {
        "id": "ae592060-2382-4663-9476-b015c685593c",
        "type": "as_policy",
        "status": "DELETING",
        "details": [
            {
                "name": "capacityProviderName",
                "value": "MyCapacityProvider2"
            },
            {
                "name": "scalingPolicyName",
                "value": "ECManagedAutoScalingPolicy-a1b2c3d4-5678-90ab-
cdef-EXAMPLE22222"
            }
        ]
    }
],
"attachmentsStatus": "UPDATE_IN_PROGRESS"
}
}

```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[集群容量提供程序](#)。

示例 3：从集群中移除所有容量提供商

以下put-cluster-capacity-providers示例将所有现有的容量提供程序从集群中移除。

```

aws ecs put-cluster-capacity-providers \
  --cluster MyCluster \
  --capacity-providers [] \

```



```
--default-capacity-provider-strategy []
```

输出：

```
{
  "cluster": {
    "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
    "clusterName": "MyCluster",
    "status": "ACTIVE",
    "registeredContainerInstancesCount": 0,
    "runningTasksCount": 0,
    "pendingTasksCount": 0,
    "activeServicesCount": 0,
    "statistics": [],
    "tags": [],
    "settings": [
      {
        "name": "containerInsights",
        "value": "enabled"
      }
    ],
    "capacityProviders": [],
    "defaultCapacityProviderStrategy": [],
    "attachments": [
      {
        "id": "0fb0c8f4-6edd-4de1-9b09-17e470ee1918",
        "type": "as_policy",
        "status": "DELETING",
        "details": [
          {
            "name": "capacityProviderName",
            "value": "MyCapacityProvider1"
          },
          {
            "name": "scalingPolicyName",
            "value": "ECSManagedAutoScalingPolicy-a1b2c3d4-5678-90ab-
cdef-EXAMPLE11111"
          }
        ]
      },
      {
        "id": "ae592060-2382-4663-9476-b015c685593c",
        "type": "as_policy",
```

```

        "status": "DELETING",
        "details": [
            {
                "name": "capacityProviderName",
                "value": "MyCapacityProvider2"
            },
            {
                "name": "scalingPolicyName",
                "value": "ECManagedAutoScalingPolicy-a1b2c3d4-5678-90ab-
cdef-EXAMPLE22222"
            }
        ]
    },
    "attachmentsStatus": "UPDATE_IN_PROGRESS"
}

```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[集群容量提供程序](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[PutClusterCapacityProviders](#)中的。

## register-task-definition

以下代码示例演示了如何使用 register-task-definition。

### AWS CLI

示例 1：使用 JSON 文件注册任务定义

以下 register-task-definition 示例将任务定义注册到指定的家族。容器定义以 JSON 格式保存在指定的文件位置。

```

aws ecs register-task-definition \
  --cli-input-json file://<path_to_json_file>/sleep360.json

```

sleep360.json 的内容：

```

{
  "containerDefinitions": [
    {
      "name": "sleep",

```

```
        "image": "busybox",
        "cpu": 10,
        "command": [
            "sleep",
            "360"
        ],
        "memory": 10,
        "essential": true
    }
],
"family": "sleep360"
}
```

输出：

```
{
  "taskDefinition": {
    "status": "ACTIVE",
    "family": "sleep360",
    "placementConstraints": [],
    "compatibilities": [
      "EXTERNAL",
      "EC2"
    ],
    "volumes": [],
    "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-definition/sleep360:1",
    "containerDefinitions": [
      {
        "environment": [],
        "name": "sleep",
        "mountPoints": [],
        "image": "busybox",
        "cpu": 10,
        "portMappings": [],
        "command": [
          "sleep",
          "360"
        ],
        "memory": 10,
        "essential": true,
        "volumesFrom": []
      }
    ]
  }
}
```

```
    ],
    "revision": 1
  }
}
```

有关更多信息，请参阅 Amazon ECS 开发人员指南中的[任务定义示例](#)。

## 示例 2：使用 JSON 字符串参数注册任务定义

以下 `register-task-definition` 示例使用作为带有转义双引号的 JSON 字符串参数提供的容器定义注册任务定义。

```
aws ecs register-task-definition \
  --family sleep360 \
  --container-definitions "[{\"name\":\"sleep\",\"image\":\"busybox\",\"cpu\":10,
  \"command\":[\"sleep\",\"360\"],\"memory\":10,\"essential\":true}]"
```

输出与前面的示例相同。

有关更多信息，请参阅 Amazon ECS 开发人员指南中的[创建任务定义](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterTaskDefinition](#) 中的。

## run-task

以下代码示例演示了如何使用 `run-task`。

### AWS CLI

#### 在默认集群上运行任务

以下 `run-task` 示例在默认集群上运行任务并使用客户端令牌。

```
aws ecs run-task \
  --cluster default \
  --task-definition sleep360:1 \
  --client-token 550e8400-e29b-41d4-a716-446655440000
```

输出：

```
{
  "tasks": [
    {
```

```
"attachments": [],
"attributes": [
  {
    "name": "ecs.cpu-architecture",
    "value": "x86_64"
  }
],
"availabilityZone": "us-east-1b",
"capacityProviderName": "example-capacity-provider",
"clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/default",
"containerInstanceArn": "arn:aws:ecs:us-east-1:123456789012:container-
instance/default/bc4d2ec611d04bb7bb97e83ceEXAMPLE",
"containers": [
  {
    "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/
default/d6f51cc5bbc94a47969c92035e9f66f8/75853d2d-711e-458a-8362-0f0aEXAMPLE",
    "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/default/
d6f51cc5bbc94a47969c9203EXAMPLE",
    "name": "sleep",
    "image": "busybox",
    "lastStatus": "PENDING",
    "networkInterfaces": [],
    "cpu": "10",
    "memory": "10"
  }
],
"cpu": "10",
"createdAt": "2023-11-21T16:59:34.403000-05:00",
"desiredStatus": "RUNNING",
"enableExecuteCommand": false,
"group": "family:sleep360",
"lastStatus": "PENDING",
"launchType": "EC2",
"memory": "10",
"overrides": {
  "containerOverrides": [
    {
      "name": "sleep"
    }
  ],
  "inferenceAcceleratorOverrides": []
},
"tags": [],
```

```
        "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/default/
d6f51cc5bbc94a47969c9203EXAMPLE",
        "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-
definition/sleep360:1",
        "version": 1
    }
],
"failures": []
}
```

有关更多信息，请参阅 Amazon ECS 开发人员指南中的[运行任务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RunTask](#)中的。

## start-task

以下代码示例演示了如何使用 start-task。

### AWS CLI

#### 开始新任务

以下内容使用默认集群中指定容器实例上 sleep360 任务定义的最新版本 start-task 启动任务。

```
aws ecs start-task \
  --task-definition sleep360 \
  --container-instances 765936fadbdd46b5991a4bd70c2a43d4
```

输出：

```
{
  "tasks": [
    {
      "taskArn": "arn:aws:ecs:us-west-2:130757420319:task/
default/666fdccc2e2d4b6894dd422f4eeee8f8",
      "clusterArn": "arn:aws:ecs:us-west-2:130757420319:cluster/default",
      "taskDefinitionArn": "arn:aws:ecs:us-west-2:130757420319:task-
definition/sleep360:3",
      "containerInstanceArn": "arn:aws:ecs:us-west-2:130757420319:container-
instance/default/765936fadbdd46b5991a4bd70c2a43d4",
      "overrides": {
        "containerOverrides": [
          {
```

```

        "name": "sleep"
      }
    ]
  },
  "lastStatus": "PENDING",
  "desiredStatus": "RUNNING",
  "cpu": "128",
  "memory": "128",
  "containers": [
    {
      "containerArn": "arn:aws:ecs:us-
west-2:130757420319:container/75f11ed4-8a3d-4f26-a33b-ad1db9e02d41",
      "taskArn": "arn:aws:ecs:us-west-2:130757420319:task/
default/666fdccc2e2d4b6894dd422f4eeee8f8",
      "name": "sleep",
      "lastStatus": "PENDING",
      "networkInterfaces": [],
      "cpu": "10",
      "memory": "10"
    }
  ],
  "version": 1,
  "createdAt": 1563421494.186,
  "group": "family:sleep360",
  "launchType": "EC2",
  "attachments": [],
  "tags": []
}
],
"failures": []
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartTask](#)中的。

## stop-task

以下代码示例演示了如何使用 stop-task。

### AWS CLI

#### 停止任务

以下操作stop-task会阻止指定任务在默认集群中运行。

```
aws ecs stop-task \  
  --task 666fdccc2e2d4b6894dd422f4eeee8f8
```

输出：

```
{  
  "task": {  
    "taskArn": "arn:aws:ecs:us-west-2:130757420319:task/  
default/666fdccc2e2d4b6894dd422f4eeee8f8",  
    "clusterArn": "arn:aws:ecs:us-west-2:130757420319:cluster/default",  
    "taskDefinitionArn": "arn:aws:ecs:us-west-2:130757420319:task-definition/  
sleep360:3",  
    "containerInstanceArn": "arn:aws:ecs:us-west-2:130757420319:container-  
instance/default/765936fadbdd46b5991a4bd70c2a43d4",  
    "overrides": {  
      "containerOverrides": []  
    },  
    "lastStatus": "STOPPED",  
    "desiredStatus": "STOPPED",  
    "cpu": "128",  
    "memory": "128",  
    "containers": [],  
    "version": 2,  
    "stoppedReason": "Taskfailedtostart",  
    "stopCode": "TaskFailedToStart",  
    "connectivity": "CONNECTED",  
    "connectivityAt": 1563421494.186,  
    "pullStartedAt": 1563421494.252,  
    "pullStoppedAt": 1563421496.252,  
    "executionStoppedAt": 1563421497,  
    "createdAt": 1563421494.186,  
    "stoppingAt": 1563421497.252,  
    "stoppedAt": 1563421497.252,  
    "group": "family:sleep360",  
    "launchType": "EC2",  
    "attachments": [],  
    "tags": []  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopTask](#)中的。



## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

### AWS CLI

为资源添加标签

以下 `tag-resource` 示例向指定资源添加单个标签。

```
aws ecs tag-resource \  
  --resource-arn arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster \  
  --tags key=key1,value=value1
```

此命令不生成任何输出。

向资源添加多个标签

以下 `tag-resource` 示例向指定资源添加多个标签。

```
aws ecs tag-resource \  
  --resource-arn arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster \  
  --tags key=key1,value=value1 key=key2,value=value2 key=key3,value=value3
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 `untag-resource`。

### AWS CLI

从资源中移除标签

以下 `untag-resource` 示例从指定资源中删除列出的标签。

```
aws ecs untag-resource \  
  --resource-arn arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster \  
  --tag-keys key1,key2
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-cluster-settings

以下代码示例演示了如何使用 update-cluster-settings。

### AWS CLI

#### 修改集群的设置

以下update-cluster-settings示例为集default群启用 CloudWatch 容器见解。

```
aws ecs update-cluster-settings \  
  --cluster default \  
  --settings name=containerInsights,value=enabled
```

输出：

```
{  
  "cluster": {  
    "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",  
    "clusterName": "default",  
    "status": "ACTIVE",  
    "registeredContainerInstancesCount": 0,  
    "runningTasksCount": 0,  
    "pendingTasksCount": 0,  
    "activeServicesCount": 0,  
    "statistics": [],  
    "tags": [],  
    "settings": [  
      {  
        "name": "containerInsights",  
        "value": "enabled"  
      }  
    ]  
  }  
}
```

有关更多信息，请参阅 Amazon ECS 开发人员指南中的[修改账户设置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateClusterSettings](#)中的。

## update-container-agent

以下代码示例演示了如何使用 update-container-agent。

### AWS CLI

更新 Amazon ECS 容器实例上的容器代理

以下update-container-agent示例更新默认集群中指定容器实例上的容器代理。

```
aws ecs update-container-agent --cluster default --container-instance
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
```

输出：

```
{
  "containerInstance": {
    "status": "ACTIVE",
    ...
    "agentUpdateStatus": "PENDING",
    "versionInfo": {
      "agentVersion": "1.0.0",
      "agentHash": "4023248",
      "dockerVersion": "DockerVersion: 1.5.0"
    }
  }
}
```

有关更多信息，请参阅 [Amazon ECS 开发人员指南中的更新 Amazon ECS 容器代理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateContainerAgent](#)中的。

## update-container-instances-state

以下代码示例演示了如何使用 update-container-instances-state。

### AWS CLI

更新容器实例的状态

以下内容update-container-instances-state更新了指定容器实例的状态，如果该容器实例注册到DRAINING该容器实例，则会将其从集群中移除。

```
aws ecs update-container-instances-state \  
  --container-instances 765936fadbdd46b5991a4bd70c2a43d4 \  
  --status DRAINING
```

输出：

```
{  
  "containerInstances": [  
    {  
      "containerInstanceArn": "arn:aws:ecs:us-west-2:130757420319:container-  
instance/default/765936fadbdd46b5991a4bd70c2a43d4",  
      "ec2InstanceId": "i-013d87ffbb4d513bf",  
      "version": 4390,  
      "versionInfo": {  
        "agentVersion": "1.29.0",  
        "agentHash": "a190a73f",  
        "dockerVersion": "DockerVersion:18.06.1-ce"  
      },  
      "remainingResources": [  
        {  
          "name": "CPU",  
          "type": "INTEGER",  
          "doubleValue": 0,  
          "longValue": 0,  
          "integerValue": 1536  
        },  
        {  
          "name": "MEMORY",  
          "type": "INTEGER",  
          "doubleValue": 0,  
          "longValue": 0,  
          "integerValue": 2681  
        },  
        {  
          "name": "PORTS",  
          "type": "STRINGSET",  
          "doubleValue": 0,  
          "longValue": 0,  
          "integerValue": 0,  
          "stringSetValue": [  
            "22",  
            "2376",  
            "2375",
```

```
        "51678",
        "51679"
    ]
},
{
    "name": "PORTS_UDP",
    "type": "STRINGSET",
    "doubleValue": 0,
    "longValue": 0,
    "integerValue": 0,
    "stringSetValue": []
}
],
"registeredResources": [
    {
        "name": "CPU",
        "type": "INTEGER",
        "doubleValue": 0,
        "longValue": 0,
        "integerValue": 2048
    },
    {
        "name": "MEMORY",
        "type": "INTEGER",
        "doubleValue": 0,
        "longValue": 0,
        "integerValue": 3705
    },
    {
        "name": "PORTS",
        "type": "STRINGSET",
        "doubleValue": 0,
        "longValue": 0,
        "integerValue": 0,
        "stringSetValue": [
            "22",
            "2376",
            "2375",
            "51678",
            "51679"
        ]
    }
},
{
    "name": "PORTS_UDP",
```

```
        "type": "STRINGSET",
        "doubleValue": 0,
        "longValue": 0,
        "integerValue": 0,
        "stringSetValue": []
      }
    ],
    "status": "DRAINING",
    "agentConnected": true,
    "runningTasksCount": 2,
    "pendingTasksCount": 0,
    "attributes": [
      {
        "name": "ecs.capability.secrets.asm.environment-variables"
      },
      {
        "name": "ecs.capability.branch-cni-plugin-version",
        "value": "e0703516-"
      },
      {
        "name": "ecs.ami-id",
        "value": "ami-00e0090ac21971297"
      },
      {
        "name": "ecs.capability.secrets.asm.bootstrap.log-driver"
      },
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.none"
      },
      {
        "name": "ecs.capability.ecr-endpoint"
      },
      {
        "name": "ecs.capability.docker-plugin.local"
      },
      {
        "name": "ecs.capability.task-cpu-mem-limit"
      },
      {
        "name": "ecs.capability.secrets.ssm.bootstrap.log-driver"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.30"
      }
    ]
  },
}
```

```
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.31"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.32"
},
{
  "name": "ecs.availability-zone",
  "value": "us-west-2c"
},
{
  "name": "ecs.capability.aws-appmesh"
},
{
  "name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.24"
},
{
  "name": "ecs.capability.task-eni-trunking"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.25"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.26"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.27"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.28"
},
{
  "name": "com.amazonaws.ecs.capability.privileged-container"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.29"
},
{
  "name": "ecs.cpu-architecture",
  "value": "x86_64"
},
},
```

```
{
  "name": "com.amazonaws.ecs.capability.ecr-auth"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.20"
},
{
  "name": "ecs.os-type",
  "value": "linux"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.21"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.22"
},
{
  "name": "ecs.capability.task-eia"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.23"
},
{
  "name": "ecs.capability.private-registry-
authentication.secretsmanager"
},
{
  "name": "com.amazonaws.ecs.capability.logging-driver.syslog"
},
{
  "name": "com.amazonaws.ecs.capability.logging-driver.json-file"
},
{
  "name": "ecs.capability.execution-role-awslogs"
},
{
  "name": "ecs.vpc-id",
  "value": "vpc-1234"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
```



```
    },
    {
      "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
    },
    {
      "name": "ecs.capability.task-eni"
    },
    {
      "name": "ecs.capability.execution-role-ecr-pull"
    },
    {
      "name": "ecs.capability.container-health-check"
    },
    {
      "name": "ecs.subnet-id",
      "value": "subnet-1234"
    },
    {
      "name": "ecs.instance-type",
      "value": "c5.large"
    },
    {
      "name": "com.amazonaws.ecs.capability.task-iam-role-network-
host"
    },
    {
      "name": "ecs.capability.container-ordering"
    },
    {
      "name": "ecs.capability.cni-plugin-version",
      "value": "91ccefc8-2019.06.0"
    },
    {
      "name": "ecs.capability.pid-ipc-namespace-sharing"
    },
    {
      "name": "ecs.capability.secrets.ssm.environment-variables"
    },
    {
      "name": "com.amazonaws.ecs.capability.task-iam-role"
    }
  ],
  "registeredAt": 1560788724.507,
  "attachments": [],
```

```
        "tags": []
      }
    ],
    "failures": []
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateContainerInstancesState](#)中的。

## update-service-primary-task-set

以下代码示例演示了如何使用 update-service-primary-task-set。

### AWS CLI

#### 更新服务的主任务集

以下update-service-primary-task-set示例更新了指定服务的主任务集。

```
aws ecs update-service-primary-task-set \
  --cluster MyCluster \
  --service MyService \
  --primary-task-set arn:aws:ecs:us-west-2:123456789012:task-set/MyCluster/
MyService/ecs-svc/1234567890123456789
```

输出：

```
{
  "taskSet": {
    "id": "ecs-svc/1234567890123456789",
    "taskSetArn": "arn:aws:ecs:us-west-2:123456789012:task-set/MyCluster/
MyService/ecs-svc/1234567890123456789",
    "status": "PRIMARY",
    "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
sample-fargate:2",
    "computedDesiredCount": 1,
    "pendingCount": 0,
    "runningCount": 0,
    "createdAt": 1557128360.711,
    "updatedAt": 1557129412.653,
    "launchType": "EC2",
    "networkConfiguration": {
      "awsvpcConfiguration": {
```

```
        "subnets": [
            "subnet-12344321"
        ],
        "securityGroups": [
            "sg-12344312"
        ],
        "assignPublicIp": "DISABLED"
    }
},
"loadBalancers": [],
"serviceRegistries": [],
"scale": {
    "value": 50.0,
    "unit": "PERCENT"
},
"stabilityStatus": "STABILIZING",
"stabilityStatusAt": 1557129279.914
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateServicePrimaryTaskSet](#)中的。

## update-service

以下代码示例演示了如何使用 update-service。

### AWS CLI

#### 示例 1：更改服务中使用的任务定义

以下 update-service 示例将 my-http-service 服务更新为使用 amazon-ecs-sample 任务定义。

```
aws ecs update-service --service my-http-service --task-definition amazon-ecs-sample
```

#### 示例 2：更改服务中的任务数

以下 update-service 示例将服务 my-http-service 所需的任务计数更新为 3。

```
aws ecs update-service --service my-http-service --desired-count 3
```

有关更多信息，请参阅《Amazon ECS 开发人员指南》中的[更新服务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateService](#)中的。

## update-task-set

以下代码示例演示了如何使用 update-task-set。

### AWS CLI

#### 更新任务集

以下update-task-set示例更新任务集以调整比例。

```
aws ecs update-task-set \  
  --cluster MyCluster \  
  --service MyService \  
  --task-set arn:aws:ecs:us-west-2:123456789012:task-set/MyCluster/MyService/ecs-  
svc/1234567890123456789 \  
  --scale value=50,unit=PERCENT
```

输出：

```
{  
  "taskSet": {  
    "id": "ecs-svc/1234567890123456789",  
    "taskSetArn": "arn:aws:ecs:us-west-2:123456789012:task-set/MyCluster/  
MyService/ecs-svc/1234567890123456789",  
    "status": "ACTIVE",  
    "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/  
sample-fargate:2",  
    "computedDesiredCount": 0,  
    "pendingCount": 0,  
    "runningCount": 0,  
    "createdAt": 1557128360.711,  
    "updatedAt": 1557129279.914,  
    "launchType": "EC2",  
    "networkConfiguration": {  
      "awsvpcConfiguration": {  
        "subnets": [  
          "subnet-12344321"  
        ],  
        "securityGroups": [  
          "sg-12344321"  
        ],  
      }  
    }  
  }  
}
```

```
        "assignPublicIp": "DISABLED"
      }
    },
    "loadBalancers": [],
    "serviceRegistries": [],
    "scale": {
      "value": 50.0,
      "unit": "PERCENT"
    },
    "stabilityStatus": "STABILIZING",
    "stabilityStatusAt": 1557129279.914
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateTaskSet](#)中的。

## 使用 Amazon EFS 示例 AWS CLI

以下代码示例向您展示了如何在 Amazon EFS 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-file-system**

以下代码示例演示了如何使用 create-file-system。

AWS CLI

#### 创建加密文件系统

以下create-file-system示例使用默认 CMK 创建加密文件系统。它还添加了标签Name=my-file-system。

```
aws efs create-file-system \  
  --performance-mode generalPurpose \  
  --throughput-mode bursting \  
  --encrypted \  
  --tags Key=Name,Value=my-file-system
```

输出：

```
{  
  "OwnerId": "123456789012",  
  "CreationToken": "console-d7f56c5f-e433-41ca-8307-9d9c0example",  
  "FileSystemId": "fs-c7a0456e",  
  "FileSystemArn": "arn:aws:elasticfilesystem:us-west-2:123456789012:file-system/  
fs-48499b4d",  
  "CreationTime": 1595286880.0,  
  "LifecycleState": "creating",  
  "Name": "my-file-system",  
  "NumberOfMountTargets": 0,  
  "SizeInBytes": {  
    "Value": 0,  
    "ValueInIA": 0,  
    "ValueInStandard": 0  
  },  
  "PerformanceMode": "generalPurpose",  
  "Encrypted": true,  
  "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/a59b3472-e62c-42e4-  
adcf-30d92example",  
  "ThroughputMode": "bursting",  
  "Tags": [  
    {  
      "Key": "Name",  
      "Value": "my-file-system"  
    }  
  ]  
}
```

有关更多信息，请参阅 [《亚马逊弹性文件系统用户指南》](#) 中的 [创建 Amazon E FS 文件系统](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFileSystem](#)中的。

## create-mount-target

以下代码示例演示了如何使用 create-mount-target。

### AWS CLI

#### 创建挂载目标

以下create-mount-target示例为指定的文件系统创建挂载目标。

```
aws efs create-mount-target \  
  --file-system-id fs-c7a0456e \  
  --subnet-id subnet-02bf4c428bexample \  
  --security-groups sg-068f739363example
```

输出：

```
{  
  "OwnerId": "123456789012",  
  "MountTargetId": "fsmt-f9a14450",  
  "FileSystemId": "fs-c7a0456e",  
  "SubnetId": "subnet-02bf4c428bexample",  
  "LifecycleState": "creating",  
  "IpAddress": "10.0.1.24",  
  "NetworkInterfaceId": "eni-02d542216aexample",  
  "AvailabilityZoneId": "use2-az2",  
  "AvailabilityZoneName": "us-east-2b",  
  "VpcId": "vpc-0123456789abcdef0"  
}
```

有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的[创建挂载目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateMountTarget](#)中的。

## delete-file-system

以下代码示例演示了如何使用 delete-file-system。

### AWS CLI

#### 删除文件系统

以下delete-file-system示例删除了指定的文件系统。

```
aws efs delete-file-system \  
  --file-system-id fs-c7a0456e
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊弹性文件系统用户指南](#)》中的删除 Amazon E FS 文件系统。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFileSystem](#)中的。

## delete-mount-target

以下代码示例演示了如何使用 delete-mount-target。

### AWS CLI

删除装载目标

以下delete-mount-target示例删除了指定的挂载目标。

```
aws efs delete-mount-target \  
  --mount-target-id fsmt-f9a14450
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的[创建挂载目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMountTarget](#)中的。

## describe-file-systems

以下代码示例演示了如何使用 describe-file-systems。

### AWS CLI

描述文件系统

以下describe-file-systems示例描述了指定的文件系统。

```
aws efs describe-file-systems \  
  --file-system-id fs-c7a0456e
```



输出：

```
{
  "FileSystems": [
    {
      "OwnerId": "123456789012",
      "CreationToken": "console-d7f56c5f-e433-41ca-8307-9d9c0example",
      "FileSystemId": "fs-c7a0456e",
      "FileSystemArn": "arn:aws:elasticfilesystem:us-west-2:123456789012:file-
system/fs-48499b4d",
      "CreationTime": 1595286880.0,
      "LifecycleState": "available",
      "Name": "my-file-system",
      "NumberOfMountTargets": 3,
      "SizeInBytes": {
        "Value": 6144,
        "Timestamp": 1600991437.0,
        "ValueInIA": 0,
        "ValueInStandard": 6144
      },
      "PerformanceMode": "generalPurpose",
      "Encrypted": true,
      "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/a59b3472-e62c-42e4-
adcf-30d92example",
      "ThroughputMode": "bursting",
      "Tags": [
        {
          "Key": "Name",
          "Value": "my-file-system"
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅《[亚马逊弹性文件系统用户指南](#)》中的管理 Amazon E FS 文件系统。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFileSystems](#)中的。

## describe-mount-targets

以下代码示例演示了如何使用 describe-mount-targets。

## AWS CLI

### 描述挂载目标

以下describe-mount-targets示例描述了指定的挂载目标。

```
aws efs describe-mount-targets \  
  --mount-target-id fsmt-f9a14450
```

输出：

```
{  
  "MountTargets": [  
    {  
      "OwnerId": "123456789012",  
      "MountTargetId": "fsmt-f9a14450",  
      "FileSystemId": "fs-c7a0456e",  
      "SubnetId": "subnet-02bf4c428bexample",  
      "LifecycleState": "creating",  
      "IpAddress": "10.0.1.24",  
      "NetworkInterfaceId": "eni-02d542216aexample",  
      "AvailabilityZoneId": "use2-az2",  
      "AvailabilityZoneName": "us-east-2b",  
      "VpcId": "vpc-0123456789abcdef0"  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的[创建挂载目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeMountTargets](#)中的。

## describe-tags

以下代码示例演示了如何使用 describe-tags。

### AWS CLI

#### 描述文件系统的标签

以下describe-tags示例描述了指定文件系统的标签。

```
aws efs describe-tags \  
  --file-system-id fs-c7a0456e
```

```
--file-system-id fs-c7a0456e
```

输出：

```
{
  "Tags": [
    {
      "Key": "Name",
      "Value": "my-file-system"
    },
    {
      "Key": "Department",
      "Value": "Business Intelligence"
    }
  ]
}
```

有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的管理文件系统[标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTags](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

检索资源的标签

以下`list-tags-for-resource`示例检索与指定文件系统关联的标签。

```
aws efs list-tags-for-resource \
  --resource-id fs-c7a0456e
```

输出：

```
{
  "Tags": [
    {
      "Key": "Name",
      "Value": "my-file-system"
    },
  ],
}
```

```
{
  "Key": "Department",
  "Value": "Business Intelligence"
}
]
```

有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的管理文件系统[标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源添加标签

以下tag-resource示例将标签Department=Business Intelligence添加到指定的文件系统。

```
aws efs tag-resource \
  --resource-id fs-c7a0456e \
  --tags Key=Department,Value="Business Intelligence"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的管理文件系统[标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从资源中移除标签

以下untag-resource示例从指定的文件系统中删除带有Department标签密钥的标签。

```
aws efs untag-resource \
```

```
--resource-id fs-c7a0456e \  
--tag-keys Department
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的管理文件系统[标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## 使用 Amazon EKS 的示例 AWS CLI

以下代码示例向您展示如何在 Amazon EKS 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-encryption-config**

以下代码示例演示了如何使用 `associate-encryption-config`。

AWS CLI

将加密配置关联到现有集群

以下`associate-encryption-config`示例在尚未启用加密的现有 EKS 集群上启用加密。

```
aws eks associate-encryption-config \  
  --cluster-name my-eks-cluster \  
  --encryption-config '[{"resources":["secrets"],"provider":  
{"keyArn":"arn:aws:kms:region-code:account:key/key"}]'
```

输出：

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "InProgress",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\\\"arn:aws:kms:region-code:account:key/key\\\"}]}]"
      }
    ],
    "createdAt": "2024-03-14T11:01:26.297000-04:00",
    "errors": []
  }
}
```

有关更多信息，请参阅 Amazon EKS 用户指南中的在[现有集群上启用秘密加密](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateEncryptionConfig](#) 中的。

## associate-identity-provider-config

以下代码示例演示了如何使用 `associate-identity-provider-config`。

### AWS CLI

将身份提供商关联到您的 Amazon EKS 集群

以下 `associate-identity-provider-config` 示例将身份提供商关联到您的 Amazon EKS 集群。

```
aws eks associate-identity-provider-config \
  --cluster-name my-eks-cluster \
  --oidc 'identityProviderConfigName=my-identity-provider,issuerUrl=https://
oidc.eks.us-east-2.amazonaws.com/
id/38D6A4619A0A69E342B113ED7F1A7652,clientId=kubernetes,usernameClaim=email,usernamePrefix=m
username-prefix,groupsClaim=my-claim,groupsPrefix=my-groups-
prefix,requiredClaims={Claim1=value1,Claim2=value2}' \
  --tags env=dev
```

输出：

```
{
  "update": {
    "id": "8c6c1bef-61fe-42ac-a242-89412387b8e7",
    "status": "InProgress",
    "type": "AssociateIdentityProviderConfig",
    "params": [
      {
        "type": "IdentityProviderConfig",
        "value": "[{\"type\": \"oidc\", \"name\": \"my-identity-provider\"}]"
      }
    ],
    "createdAt": "2024-04-11T13:46:49.648000-04:00",
    "errors": []
  },
  "tags": {
    "env": "dev"
  }
}
```

有关更多信息，请参阅 [Amazon EKS 用户指南中的通过 OpenID Connect 身份提供商对集群的用户进行身份验证——关联 OIDC 身份提供商](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateIdentityProviderConfig](#) 中的。

## create-addon

以下代码示例演示了如何使用 create-addon。

### AWS CLI

示例 1：为相应的 EKS 集群版本创建具有默认兼容版本的 Amazon EKS 附加组件

以下 create-addon 示例命令为相应的 EKS 集群版本创建具有默认兼容版本的 Amazon EKS 附加组件。

```
aws eks create-addon \
  --cluster-name my-eks-cluster \
  --addon-name my-eks-addon \
  --service-account-role-arn arn:aws:iam::111122223333:role/role-name
```

输出：

```
{
  "addon": {
    "addonName": "my-eks-addon",
    "clusterName": "my-eks-cluster",
    "status": "CREATING",
    "addonVersion": "v1.15.1-eksbuild.1",
    "health": {
      "issues": []
    },
    "addonArn": "arn:aws:eks:us-east-2:111122223333:addon/my-eks-cluster/my-eks-addon/1ec71ee1-b9c2-8915-4e17-e8be0a55a149",
    "createdAt": "2024-03-14T12:20:03.264000-04:00",
    "modifiedAt": "2024-03-14T12:20:03.283000-04:00",
    "serviceAccountRoleArn": "arn:aws:iam::111122223333:role/role-name",
    "tags": {}
  }
}
```

有关更多信息，请参阅 [Amazon EKS 用户指南中的管理 Amazon EKS 插件-创建插件](#)。

示例 2：创建带有特定附加组件版本的 Amazon EKS 附加组件

以下 `create-addon` 示例命令创建具有特定附加组件版本的 Amazon EKS 附加组件。

```
aws eks create-addon \
  --cluster-name my-eks-cluster \
  --addon-name my-eks-addon \
  --service-account-role-arn arn:aws:iam::111122223333:role/role-name \
  --addon-version v1.16.4-eksbuild.2
```

输出：

```
{
  "addon": {
    "addonName": "my-eks-addon",
    "clusterName": "my-eks-cluster",
    "status": "CREATING",
    "addonVersion": "v1.16.4-eksbuild.2",
    "health": {
      "issues": []
    },
  },
}
```



```

    "addonArn": "arn:aws:eks:us-east-2:111122223333:addon/my-eks-cluster/my-eks-
addon/34c71ee6-7738-6c8b-c6bd-3921a176b5ff",
    "createdAt": "2024-03-14T12:30:24.507000-04:00",
    "modifiedAt": "2024-03-14T12:30:24.521000-04:00",
    "serviceAccountRoleArn": "arn:aws:iam::111122223333:role/role-name",
    "tags": {}
  }
}

```

有关更多信息，请参阅 [Amazon EKS 用户指南中的管理 Amazon EKS 插件-创建插件](#)。

示例 3：使用自定义配置值创建 Amazon EKS 插件并解决冲突详情

以下 create-addon 示例命令使用自定义配置值创建一个 Amazon EKS 插件并解决冲突详情。

```

aws eks create-addon \
  --cluster-name my-eks-cluster \
  --addon-name my-eks-addon \
  --service-account-role-arn arn:aws:iam::111122223333:role/role-name \
  --addon-version v1.16.4-eksbuild.2 \
  --configuration-values '{"resources":{"limits":{"cpu":"100m"}}}' \
  --resolve-conflicts OVERWRITE

```

输出：

```

{
  "addon": {
    "addonName": "my-eks-addon",
    "clusterName": "my-eks-cluster",
    "status": "CREATING",
    "addonVersion": "v1.16.4-eksbuild.2",
    "health": {
      "issues": []
    },
    "addonArn": "arn:aws:eks:us-east-2:111122223333:addon/my-eks-cluster/my-eks-
addon/a6c71ee9-0304-9237-1be8-25af1b0f1ffb",
    "createdAt": "2024-03-14T12:35:58.313000-04:00",
    "modifiedAt": "2024-03-14T12:35:58.327000-04:00",
    "serviceAccountRoleArn": "arn:aws:iam::111122223333:role/role-name",
    "tags": {},
    "configurationValues": "{\"resources\":{\"limits\":{\"cpu\":\"100m\"}}}"
  }
}

```

有关更多信息，请参阅 [Amazon EKS 用户指南中的管理 Amazon EKS 插件-创建插件](#)。

#### 示例 4：使用自定义 JSON 配置值文件创建 Amazon EKS 附加组件

以下 `create-addon` 示例命令使用自定义配置值创建一个 Amazon EKS 附加组件，并详细说明如何解决冲突。

```
aws eks create-addon \  
  --cluster-name my-eks-cluster \  
  --addon-name my-eks-addon \  
  --service-account-role-arn arn:aws:iam::111122223333:role/role-name \  
  --addon-version v1.16.4-eksbuild.2 \  
  --configuration-values 'file://configuration-values.json' \  
  --resolve-conflicts OVERWRITE \  
  --tags '{"eks-addon-key-1": "value-1" , "eks-addon-key-2": "value-2"}'
```

`configuration-values.json` 的内容：

```
{  
  "resources": {  
    "limits": {  
      "cpu": "150m"  
    }  
  },  
  "env": {  
    "AWS_VPC_K8S_CNI_LOGLEVEL": "ERROR"  
  }  
}
```

输出：

```
{  
  "addon": {  
    "addonName": "my-eks-addon",  
    "clusterName": "my-eks-cluster",  
    "status": "CREATING",  
    "addonVersion": "v1.16.4-eksbuild.2",  
    "health": {  
      "issues": []  
    },  
    "addonArn": "arn:aws:eks:us-east-2:111122223333:addon/my-eks-cluster/my-eks-addon/d8c71ef8-fbd8-07d0-fb32-6a7be19eeced",  
  }  
}
```

```

    "createdAt": "2024-03-14T13:10:51.763000-04:00",
    "modifiedAt": "2024-03-14T13:10:51.777000-04:00",
    "serviceAccountRoleArn": "arn:aws:iam::111122223333:role/role-name",
    "tags": {
      "eks-addon-key-1": "value-1",
      "eks-addon-key-2": "value-2"
    },
    "configurationValues": "{\n  \n  \"resources\": {\n    \n    \"limits\": {\n\n      \n      \"cpu\": \"150m\"\n    }\n  },\n  \n  \"env\": {\n\n    \n    \"AWS_VPC_K8S_CNI_LOGLEVEL\": \"ERROR\"\n  }\n}"
  }
}

```

有关更多信息，请参阅 [Amazon EKS 用户指南中的管理 Amazon EKS 插件-创建插件](#)。

#### 示例 5：使用自定义 YAML 配置值文件创建 Amazon EKS 附加组件

以下 create-addon 示例命令使用自定义配置值创建一个 Amazon EKS 附加组件，并详细说明如何解决冲突。

```

aws eks create-addon \
  --cluster-name my-eks-cluster \
  --addon-name my-eks-addon \
  --service-account-role-arn arn:aws:iam::111122223333:role/role-name \
  --addon-version v1.16.4-eksbuild.2 \
  --configuration-values 'file://configuration-values.yaml' \
  --resolve-conflicts OVERWRITE \
  --tags '{"eks-addon-key-1": "value-1" , "eks-addon-key-2": "value-2"}'

```

configuration-values.yaml 的内容：

```

resources:
  limits:
    cpu: '100m'
env:
  AWS_VPC_K8S_CNI_LOGLEVEL: 'DEBUG'

```

输出：

```

{
  "addon": {
    "addonName": "my-eks-addon",

```

```

    "clusterName": "my-eks-cluster",
    "status": "CREATING",
    "addonVersion": "v1.16.4-eksbuild.2",
    "health": {
      "issues": []
    },
    "addonArn": "arn:aws:eks:us-east-2:111122223333:addon/my-eks-cluster/my-eks-
addon/d4c71efb-3909-6f36-a548-402cd4b5d59e",
    "createdAt": "2024-03-14T13:15:45.220000-04:00",
    "modifiedAt": "2024-03-14T13:15:45.237000-04:00",
    "serviceAccountRoleArn": "arn:aws:iam::111122223333:role/role-name",
    "tags": {
      "eks-addon-key-3": "value-3",
      "eks-addon-key-4": "value-4"
    },
    "configurationValues": "resources:\n      limits:\n          cpu: '100m'\nenv:\n
AWS_VPC_K8S_CNI_LOGLEVEL: 'INFO'"
  }
}

```

有关更多信息，请参阅 [Amazon EKS 用户指南中的管理 Amazon EKS 插件-创建插件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateAddon](#) 中的。

## create-cluster

以下代码示例演示了如何使用 `create-cluster`。

### AWS CLI

#### 创建新集群

此示例命令将在您的默认区域中创建一个名为 `prod` 的集群。

命令:

```

aws eks create-cluster --name prod \
--role-arn arn:aws:iam::012345678910:role/eks-service-role-
AWSServiceRoleForAmazonEKS-J7ONKE3BQ4PI \
--resources-vpc-config subnetIds=subnet-6782e71e,subnet-
e7e761ac,securityGroupIds=sg-6979fe18

```

输出：

```
{
  "cluster": {
    "name": "prod",
    "arn": "arn:aws:eks:us-west-2:012345678910:cluster/prod",
    "createdAt": 1527808069.147,
    "version": "1.10",
    "roleArn": "arn:aws:iam::012345678910:role/eks-service-role-
AWSServiceRoleForAmazonEKS-J70NKE3BQ4PI",
    "resourcesVpcConfig": {
      "subnetIds": [
        "subnet-6782e71e",
        "subnet-e7e761ac"
      ],
      "securityGroupIds": [
        "sg-6979fe18"
      ],
      "vpcId": "vpc-950809ec"
    },
    "status": "CREATING",
    "certificateAuthority": {}
  }
}
```

### 创建启用了私有端点访问和日志记录的新集群

此示例命令在您的默认区域中创建一个名为 `example` 的集群，该集群禁用了公共端点访问，启用了私有端点访问和所有日志记录类型。

命令:

```
aws eks create-cluster --name example --kubernetes-version 1.12 \
--role-arn arn:aws:iam::012345678910:role/example-cluster-ServiceRole-1XWBQWYSFRE2Q
\
--resources-vpc-config
  subnetIds=subnet-0a188dccd2f9a632f,subnet-09290d93da4278664,subnet-0f21dd86e0e91134a,subnet-
\
--logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

输出:

```
{
```

```
"cluster": {
  "name": "example",
  "arn": "arn:aws:eks:us-west-2:012345678910:cluster/example",
  "createdAt": 1565804921.901,
  "version": "1.12",
  "roleArn": "arn:aws:iam::012345678910:role/example-cluster-ServiceRole-1XWBQWYSFRE2Q",
  "resourcesVpcConfig": {
    "subnetIds": [
      "subnet-0a188dccd2f9a632f",
      "subnet-09290d93da4278664",
      "subnet-0f21dd86e0e91134a",
      "subnet-0173dead68481a583",
      "subnet-051f70a57ed6fcab6",
      "subnet-01322339c5c7de9b4"
    ],
    "securityGroupIds": [
      "sg-0c5b580845a031c10"
    ],
    "vpcId": "vpc-0f622c01f68d4afec",
    "endpointPublicAccess": false,
    "endpointPrivateAccess": true
  },
  "logging": {
    "clusterLogging": [
      {
        "types": [
          "api",
          "audit",
          "authenticator",
          "controllerManager",
          "scheduler"
        ],
        "enabled": true
      }
    ]
  },
  "status": "CREATING",
  "certificateAuthority": {},
  "platformVersion": "eks.3"
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCluster](#)中的。

## create-fargate-profile

以下代码示例演示了如何使用 create-fargate-profile。

### AWS CLI

示例 1：为具有命名空间的选择器创建 EKS Fargate 配置文件

以下create-fargate-profile示例为具有命名空间的选择器创建 EKS Fargate 配置文件。

```
aws eks create-fargate-profile \  
  --cluster-name my-eks-cluster \  
  --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \  
  --fargate-profile-name my-fargate-profile \  
  --selectors '[{"namespace": "default"}]'
```

输出：

```
{  
  "fargateProfile": {  
    "fargateProfileName": "my-fargate-profile",  
    "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/my-  
eks-cluster/my-fargate-profile/a2c72bca-318e-abe8-8ed1-27c6d4892e9e",  
    "clusterName": "my-eks-cluster",  
    "createdAt": "2024-03-19T12:38:47.368000-04:00",  
    "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",  
    "subnets": [  
      "subnet-09d912bb63ef21b9a",  
      "subnet-04ad87f71c6e5ab4d",  
      "subnet-0e2907431c9988b72"  
    ],  
    "selectors": [  
      {  
        "namespace": "default"  
      }  
    ],  
    "status": "CREATING",  
    "tags": {}  
  }  
}
```

有关更多信息，请参阅 [Amazon EKS 用户指南中的 Fargate 个人资料——创建 Fargate 个人资料](#)。

## 示例 2：为具有命名空间和标签的选择器创建 EKS Fargate 配置文件

以下 `create-fargate-profile` 示例为带有命名空间和标签的选择器创建 EKS Fargate 配置文件。

```
aws eks create-fargate-profile \  
  --cluster-name my-eks-cluster \  
  --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \  
  --fargate-profile-name my-fargate-profile \  
  --selectors '[{"namespace": "default", "labels": {"labelname1":  
"labelvalue1"}}]'
```

输出：

```
{  
  "fargateProfile": {  
    "fargateProfileName": "my-fargate-profile",  
    "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/my-  
eks-cluster/my-fargate-profile/88c72bc7-e8a4-fa34-44e4-2f1397224bb3",  
    "clusterName": "my-eks-cluster",  
    "createdAt": "2024-03-19T12:33:48.125000-04:00",  
    "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",  
    "subnets": [  
      "subnet-09d912bb63ef21b9a",  
      "subnet-04ad87f71c6e5ab4d",  
      "subnet-0e2907431c9988b72"  
    ],  
    "selectors": [  
      {  
        "namespace": "default",  
        "labels": {  
          "labelname1": "labelvalue1"  
        }  
      }  
    ],  
    "status": "CREATING",  
    "tags": {}  
  }  
}
```

有关更多信息，请参阅 [Amazon EKS 用户指南中的 Fargate 个人资料——创建 Fargate 个人资料](#)。



示例 3：为选择器创建 EKS Fargate 配置文件，其中包含命名空间和标签，以及要启动 Pod 的子网 ID。

以下 `create-fargate-profile` 示例为选择器创建 EKS Fargate 配置文件，该选择器具有命名空间和标签，以及要启动 Pod 的子网 ID。

```
aws eks create-fargate-profile \  
  --cluster-name my-eks-cluster \  
  --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \  
  --fargate-profile-name my-fargate-profile \  
  --selectors '[{"namespace": "default", "labels": {"labelname1":  
"labelvalue1"}}]' \  
  --subnets ["subnet-09d912bb63ef21b9a", "subnet-04ad87f71c6e5ab4d",  
"subnet-0e2907431c9988b72"]'
```

输出：

```
{  
  "fargateProfile": {  
    "fargateProfileName": "my-fargate-profile",  
    "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/my-  
eks-cluster/my-fargate-profile/e8c72bc8-e87b-5eb6-57cb-ed4fe57577e3",  
    "clusterName": "my-eks-cluster",  
    "createdAt": "2024-03-19T12:35:58.640000-04:00",  
    "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",  
    "subnets": [  
      "subnet-09d912bb63ef21b9a",  
      "subnet-04ad87f71c6e5ab4d",  
      "subnet-0e2907431c9988b72"  
    ],  
    "selectors": [  
      {  
        "namespace": "default",  
        "labels": {  
          "labelname1": "labelvalue1"  
        }  
      }  
    ],  
    "status": "CREATING",  
    "tags": {}  
  }  
}
```

有关更多信息，请参阅 Amazon EKS [AWS KS 用户指南中的 Fargate 个人资料——创建 Fargate 个人资料](#)。

示例 4：为具有多个命名空间和标签的选择器创建 EKS Fargate 配置文件，以及用于启动 Pod 的子网 ID

以下 create-fargate-profile 示例为选择器创建 EKS Fargate 配置文件，该选择器具有多个命名空间和标签，以及要启动 Pod 的子网 ID。

```
aws eks create-fargate-profile \
  --cluster-name my-eks-cluster \
  --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \
  --fargate-profile-name my-fargate-profile \
  --selectors '[{"namespace": "default1", "labels": {"labelname1": "labelvalue1",
"labelname2": "labelvalue2"}}, {"namespace": "default2", "labels": {"labelname1":
"labelvalue1", "labelname2": "labelvalue2"}}]' \
  --subnets ["subnet-09d912bb63ef21b9a", "subnet-04ad87f71c6e5ab4d",
"subnet-0e2907431c9988b72"] \
  --tags '{"eks-fargate-profile-key-1": "value-1" , "eks-fargate-profile-key-2":
"value-2"}'
```

输出：

```
{
  "fargateProfile": {
    "fargateProfileName": "my-fargate-profile",
    "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/my-
eks-cluster/my-fargate-profile/4cc72bbf-b766-8ee6-8d29-e62748feb3cd",
    "clusterName": "my-eks-cluster",
    "createdAt": "2024-03-19T12:15:55.271000-04:00",
    "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
    "subnets": [
      "subnet-09d912bb63ef21b9a",
      "subnet-04ad87f71c6e5ab4d",
      "subnet-0e2907431c9988b72"
    ],
    "selectors": [
      {
        "namespace": "default1",
        "labels": {
          "labelname2": "labelvalue2",
          "labelname1": "labelvalue1"
        }
      }
    ]
  }
}
```

```

    },
    {
      "namespace": "default2",
      "labels": {
        "labelname2": "labelvalue2",
        "labelname1": "labelvalue1"
      }
    }
  ],
  "status": "CREATING",
  "tags": {
    "eks-fargate-profile-key-2": "value-2",
    "eks-fargate-profile-key-1": "value-1"
  }
}
}

```

有关更多信息，请参阅 Amazon EKS [AWS KS 用户指南中的 Fargate 个人资料——创建 Fargate 个人资料](#)。

示例 5：使用命名空间和标签的通配符选择器创建 EKS Fargate 配置文件，以及用于启动 Pod 的子网 ID

以下 `create-fargate-profile` 示例为选择器创建 EKS Fargate 配置文件，该选择器具有多个命名空间和标签，以及要启动 Pod 的子网 ID。

```

aws eks create-fargate-profile \
  --cluster-name my-eks-cluster \
  --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \
  --fargate-profile-name my-fargate-profile \
  --selectors '[{"namespace": "prod*", "labels": {"labelname*?": "*value1"}}, {"namespace": "*dev*", "labels": {"labelname*?": "*value*"}}]' \
  --subnets ["subnet-09d912bb63ef21b9a", "subnet-04ad87f71c6e5ab4d", "subnet-0e2907431c9988b72"] \
  --tags '{"eks-fargate-profile-key-1": "value-1" , "eks-fargate-profile-key-2": "value-2"}'

```

输出：

```

{
  "fargateProfile": {
    "fargateProfileName": "my-fargate-profile",

```

```

    "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/my-
eks-cluster/my-fargate-profile/e8c72bd6-5966-0bfe-b77b-1802893e5a6f",
    "clusterName": "my-eks-cluster",
    "createdAt": "2024-03-19T13:05:20.550000-04:00",
    "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
    "subnets": [
      "subnet-09d912bb63ef21b9a",
      "subnet-04ad87f71c6e5ab4d",
      "subnet-0e2907431c9988b72"
    ],
    "selectors": [
      {
        "namespace": "prod*",
        "labels": {
          "labelname*?": "*value1"
        }
      },
      {
        "namespace": "*dev*",
        "labels": {
          "labelname*?": "*value*"
        }
      }
    ],
    "status": "CREATING",
    "tags": {
      "eks-fargate-profile-key-2": "value-2",
      "eks-fargate-profile-key-1": "value-1"
    }
  }
}

```

有关更多信息，请参阅 Amazon EKS [AWS KS 用户指南中的 Fargate 个人资料——创建 Fargate 个人资料](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateFargateProfile](#) 中的。

## create-nodegroup

以下代码示例演示了如何使用 create-nodegroup。

## AWS CLI

示例 1：为 Amazon EKS 集群创建托管节点组

以下 `create-nodegroup` 示例为 Amazon EKS 集群创建托管节点组。

```
aws eks create-nodegroup \  
  --cluster-name my-eks-cluster \  
  --nodegroup-name my-eks-nodegroup \  
  --node-role arn:aws:iam::111122223333:role/role-name \  
  --subnets "subnet-0e2907431c9988b72" "subnet-04ad87f71c6e5ab4d"  
"subnet-09d912bb63ef21b9a" \  
  --scaling-config minSize=1,maxSize=3,desiredSize=1 \  
  --region us-east-2
```

输出：

```
{  
  "nodegroup": {  
    "nodegroupName": "my-eks-nodegroup",  
    "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-  
cluster/my-eks-nodegroup/bac7550f-b8b8-5fbb-4f3e-7502a931119e",  
    "clusterName": "my-eks-cluster",  
    "version": "1.26",  
    "releaseVersion": "1.26.12-20240329",  
    "createdAt": "2024-04-04T13:19:32.260000-04:00",  
    "modifiedAt": "2024-04-04T13:19:32.260000-04:00",  
    "status": "CREATING",  
    "capacityType": "ON_DEMAND",  
    "scalingConfig": {  
      "minSize": 1,  
      "maxSize": 3,  
      "desiredSize": 1  
    },  
    "instanceTypes": [  
      "t3.medium"  
    ],  
    "subnets": [  
      "subnet-0e2907431c9988b72", "subnet-04ad87f71c6e5ab4d",  
      "subnet-09d912bb63ef21b9a"  
    ],  
    "amiType": "AL2_x86_64",  
    "nodeRole": "arn:aws:iam::111122223333:role/role-name",
```

```

    "diskSize": 20,
    "health": {
      "issues": []
    },
    "updateConfig": {
      "maxUnavailable": 1
    },
    "tags": {}
  }
}

```

有关更多信息，请参阅 Amazon EKS 用户指南中的[创建托管节点组](#)。

示例 2：为具有自定义实例类型和磁盘大小的 Amazon EKS 集群创建托管节点组

以下 create-nodegroup 示例为具有自定义实例类型和磁盘大小的 Amazon EKS 集群创建托管节点组。

```

aws eks create-nodegroup \
  --cluster-name my-eks-cluster \
  --nodegroup-name my-eks-nodegroup \
  --node-role arn:aws:iam::111122223333:role/role-name \
  --subnets "subnet-0e2907431c9988b72" "subnet-04ad87f71c6e5ab4d"
"subnet-09d912bb63ef21b9a" \
  --scaling-config minSize=1,maxSize=3,desiredSize=1 \
  --capacity-type ON_DEMAND \
  --instance-types 'm5.large' \
  --disk-size 50 \
  --region us-east-2

```

输出：

```

{
  "nodegroup": {
    "nodegroupName": "my-eks-nodegroup",
    "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-cluster/my-eks-nodegroup/c0c7551b-e4f9-73d9-992c-a450fdb82322",
    "clusterName": "my-eks-cluster",
    "version": "1.26",
    "releaseVersion": "1.26.12-20240329",
    "createdAt": "2024-04-04T13:46:07.595000-04:00",
    "modifiedAt": "2024-04-04T13:46:07.595000-04:00",
    "status": "CREATING",
  }
}

```

```
    "capacityType": "ON_DEMAND",
    "scalingConfig": {
      "minSize": 1,
      "maxSize": 3,
      "desiredSize": 1
    },
    "instanceTypes": [
      "m5.large"
    ],
    "subnets": [
      "subnet-0e2907431c9988b72",
      "subnet-04ad87f71c6e5ab4d",
      "subnet-09d912bb63ef21b9a"
    ],
    "amiType": "AL2_x86_64",
    "nodeRole": "arn:aws:iam::111122223333:role/role-name",
    "diskSize": 50,
    "health": {
      "issues": []
    },
    "updateConfig": {
      "maxUnavailable": 1
    },
    "tags": {}
  }
}
```

有关更多信息，请参阅 Amazon EKS 用户指南中的[创建托管节点组](#)。

示例 3：使用自定义实例类型、磁盘大小、ami 类型、容量类型、更新配置、标签、污点和标签为 Amazon EKS 集群创建托管节点组。

以下 `create-nodegroup` 示例为具有自定义实例类型、磁盘大小、ami 类型、容量类型、更新配置、标签、污点和标签的 Amazon EKS 集群创建托管节点组。

```
aws eks create-nodegroup \
  --cluster-name my-eks-cluster \
  --nodegroup-name my-eks-nodegroup \
  --node-role arn:aws:iam::111122223333:role/role-name \
  --subnets "subnet-0e2907431c9988b72" "subnet-04ad87f71c6e5ab4d" \
  "subnet-09d912bb63ef21b9a" \
  --scaling-config minSize=1,maxSize=5,desiredSize=4 \
  --instance-types 't3.large' \
```

```
--disk-size 50 \  
--ami-type AL2_x86_64 \  
--capacity-type SPOT \  
--update-config maxUnavailable=2 \  
--labels '{"my-eks-nodegroup-label-1": "value-1" , "my-eks-nodegroup-label-2":  
"value-2"}' \  
--taints '{"key": "taint-key-1" , "value": "taint-value-1", "effect":  
"NO_EXECUTE"}' \  
--tags '{"my-eks-nodegroup-key-1": "value-1" , "my-eks-nodegroup-key-2":  
"value-2"}'
```

输出：

```
{  
  "nodegroup": {  
    "nodegroupName": "my-eks-nodegroup",  
    "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-  
cluster/my-eks-nodegroup/88c75524-97af-0cb9-a9c5-7c0423ab5314",  
    "clusterName": "my-eks-cluster",  
    "version": "1.26",  
    "releaseVersion": "1.26.12-20240329",  
    "createdAt": "2024-04-04T14:05:07.940000-04:00",  
    "modifiedAt": "2024-04-04T14:05:07.940000-04:00",  
    "status": "CREATING",  
    "capacityType": "SPOT",  
    "scalingConfig": {  
      "minSize": 1,  
      "maxSize": 5,  
      "desiredSize": 4  
    },  
    "instanceTypes": [  
      "t3.large"  
    ],  
    "subnets": [  
      "subnet-0e2907431c9988b72",  
      "subnet-04ad87f71c6e5ab4d",  
      "subnet-09d912bb63ef21b9a"  
    ],  
    "amiType": "AL2_x86_64",  
    "nodeRole": "arn:aws:iam::111122223333:role/role-name",  
    "labels": {  
      "my-eks-nodegroup-label-2": "value-2",  
      "my-eks-nodegroup-label-1": "value-1"  
    }  
  }  
}
```



```
    },
    "taints": [
      {
        "key": "taint-key-1",
        "value": "taint-value-1",
        "effect": "NO_EXECUTE"
      }
    ],
    "diskSize": 50,
    "health": {
      "issues": []
    },
    "updateConfig": {
      "maxUnavailable": 2
    },
    "tags": {
      "my-eks-nodegroup-key-1": "value-1",
      "my-eks-nodegroup-key-2": "value-2"
    }
  }
}
```

有关更多信息，请参阅 Amazon EKS 用户指南中的[创建托管节点组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateNodegroup](#) 中的。

## delete-addon

以下代码示例演示了如何使用 delete-addon。

### AWS CLI

示例 1。删除 Amazon EKS 附加组件但在 EKS 集群上保留该附加软件

以下 delete-addon 示例命令删除 Amazon EKS 附加组件，但在 EKS 集群上保留该附加软件。

```
aws eks delete-addon \
  --cluster-name my-eks-cluster \
  --addon-name my-eks-addon \
  --preserve
```

输出：

```
{
  "addon": {
    "addonName": "my-eks-addon",
    "clusterName": "my-eks-cluster",
    "status": "DELETING",
    "addonVersion": "v1.9.3-eksbuild.7",
    "health": {
      "issues": []
    },
    "addonArn": "arn:aws:eks:us-east-2:111122223333:addon/my-eks-cluster/my-eks-
addon/a8c71ed3-944e-898b-9167-c763856af4b8",
    "createdAt": "2024-03-14T11:49:09.009000-04:00",
    "modifiedAt": "2024-03-14T12:03:49.776000-04:00",
    "tags": {}
  }
}
```

有关更多信息，请参阅[管理 Amazon EKS 插件-删除亚马逊 EKS 中的附加组件](#)。

示例 2。删除 Amazon EKS 附加组件并从 EKS 集群中删除该附加软件

以下delete-addon示例命令删除 Amazon EKS 附加组件，并从 EKS 集群中删除该附加软件。

```
aws eks delete-addon \
  --cluster-name my-eks-cluster \
  --addon-name my-eks-addon
```

输出：

```
{
  "addon": {
    "addonName": "my-eks-addon",
    "clusterName": "my-eks-cluster",
    "status": "DELETING",
    "addonVersion": "v1.15.1-eksbuild.1",
    "health": {
      "issues": []
    },
    "addonArn": "arn:aws:eks:us-east-2:111122223333:addon/my-eks-cluster/my-eks-
addon/bac71ed1-ec43-3bb6-88ea-f243cdb58954",
    "createdAt": "2024-03-14T11:45:31.983000-04:00",
    "modifiedAt": "2024-03-14T11:58:40.136000-04:00",
    "serviceAccountRoleArn": "arn:aws:iam::111122223333:role/role-name",
  }
}
```

```
    "tags": {}
  }
}
```

有关更多信息，请参阅[管理 Amazon EKS 插件-删除亚马逊 EKS 中的附加组件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAddon](#)中的。

## delete-cluster

以下代码示例演示了如何使用 delete-cluster。

### AWS CLI

删除 Amazon EKS 集群控制平面

以下delete-cluster示例删除了 Amazon EKS 集群控制平面。

```
aws eks delete-cluster \
  --name my-eks-cluster
```

输出：

```
{
  "cluster": {
    "name": "my-eks-cluster",
    "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster",
    "createdAt": "2024-03-14T11:31:44.348000-04:00",
    "version": "1.27",
    "endpoint": "https://DALSJ343KE23J3RN45653DSKJTT647TYD.y14.us-
east-2.eks.amazonaws.com",
    "roleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-cluster-
ServiceRole-zMF6CBakwbW",
    "resourcesVpcConfig": {
      "subnetIds": [
        "subnet-0fb75d2d8401716e7",
        "subnet-02184492f67a3d0f9",
        "subnet-04098063527aab776",
        "subnet-0e2907431c9988b72",
        "subnet-04ad87f71c6e5ab4d",
        "subnet-09d912bb63ef21b9a"
      ],
      "securityGroupIds": [
```

```
        "sg-0c1327f6270afbb36"
    ],
    "clusterSecurityGroupId": "sg-01c84d09d70f39a7f",
    "vpcId": "vpc-0012b8e1cc0abb17d",
    "endpointPublicAccess": true,
    "endpointPrivateAccess": true,
    "publicAccessCidrs": [
        "0.0.0.0/0"
    ]
},
"kubernetesNetworkConfig": {
    "serviceIpv4Cidr": "10.100.0.0/16",
    "ipFamily": "ipv4"
},
"logging": {
    "clusterLogging": [
        {
            "types": [
                "api",
                "audit",
                "authenticator",
                "controllerManager",
                "scheduler"
            ],
            "enabled": true
        }
    ]
},
"identity": {
    "oidc": {
        "issuer": "https://oidc.eks.us-east-2.amazonaws.com/id/
DALSJ343KE23J3RN45653DSKJTT647TYD"
    }
},
"status": "DELETING",
"certificateAuthority": {
    "data": "XXX_CA_DATA_XXX"
},
"platformVersion": "eks.16",
"tags": {
    "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
    "alpha.eksctl.io/cluster-name": "my-eks-cluster",
    "karpenter.sh/discovery": "my-eks-cluster",
```

```

      "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
      "auto-delete": "no",
      "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
      "EKS-Cluster-Name": "my-eks-cluster",
      "alpha.eksctl.io/cluster-oidc-enabled": "true",
      "aws:cloudformation:logical-id": "ControlPlane",
      "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
      "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
    },
    "accessConfig": {
      "authenticationMode": "API_AND_CONFIG_MAP"
    }
  }
}

```

有关更多信息，请参阅 [Amazon EKS 用户指南中的删除 Amazon EKS 集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteCluster](#) 中的。

## delete-fargate-profile

以下代码示例演示了如何使用 delete-fargate-profile。

### AWS CLI

示例 1：为具有命名空间的选择器创建 EKS Fargate 配置文件

以下 delete-fargate-profile 示例为具有命名空间的选择器创建 EKS Fargate 配置文件。

```

aws eks delete-fargate-profile \
  --cluster-name my-eks-cluster \
  --fargate-profile-name my-fargate-profile

```

输出：

```

{
  "fargateProfile": {
    "fargateProfileName": "my-fargate-profile",
    "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/my-
eks-cluster/my-fargate-profile/1ac72bb3-3fc6-2631-f1e1-98bff53bed62",

```

```

    "clusterName": "my-eks-cluster",
    "createdAt": "2024-03-19T11:48:39.975000-04:00",
    "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
    "subnets": [
      "subnet-09d912bb63ef21b9a",
      "subnet-04ad87f71c6e5ab4d",
      "subnet-0e2907431c9988b72"
    ],
    "selectors": [
      {
        "namespace": "default",
        "labels": {
          "foo": "bar"
        }
      }
    ],
    "status": "DELETING",
    "tags": {}
  }
}

```

有关更多信息，请参阅 Amazon EKS [AWS KS 用户指南中的 Fargate 个人资料——删除 Fargate](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteFargateProfile](#) 中的。

## delete-nodegroup

以下代码示例演示了如何使用 delete-nodegroup。

### AWS CLI

示例 1：删除 Amazon EKS 集群的托管节点组

以下 delete-nodegroup 示例删除了 Amazon EKS 集群的托管节点组。

```

aws eks delete-nodegroup \
  --cluster-name my-eks-cluster \
  --nodegroup-name my-eks-nodegroup

```

输出：

```

{
  "nodegroup": {

```

```
    "nodegroupName": "my-eks-nodegroup",
    "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-
cluster/my-eks-nodegroup/1ec75f5f-0e21-dcc0-b46e-f9c442685cd8",
    "clusterName": "my-eks-cluster",
    "version": "1.26",
    "releaseVersion": "1.26.12-20240329",
    "createdAt": "2024-04-08T13:25:15.033000-04:00",
    "modifiedAt": "2024-04-08T13:25:31.252000-04:00",
    "status": "DELETING",
    "capacityType": "SPOT",
    "scalingConfig": {
      "minSize": 1,
      "maxSize": 5,
      "desiredSize": 4
    },
    "instanceTypes": [
      "t3.large"
    ],
    "subnets": [
      "subnet-0e2907431c9988b72",
      "subnet-04ad87f71c6e5ab4d",
      "subnet-09d912bb63ef21b9a"
    ],
    "amiType": "AL2_x86_64",
    "nodeRole": "arn:aws:iam::111122223333:role/role-name",
    "labels": {
      "my-eks-nodegroup-label-2": "value-2",
      "my-eks-nodegroup-label-1": "value-1"
    },
    "taints": [
      {
        "key": "taint-key-1",
        "value": "taint-value-1",
        "effect": "NO_EXECUTE"
      }
    ],
    "diskSize": 50,
    "health": {
      "issues": []
    },
    "updateConfig": {
      "maxUnavailable": 2
    },
    "tags": {
```

```
        "my-eks-nodegroup-key-1": "value-1",
        "my-eks-nodegroup-key-2": "value-2"
    }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNodegroup](#)中的。

## deregister-cluster

以下代码示例演示了如何使用 `deregister-cluster`。

### AWS CLI

取消注册已连接的集群以将其从 Amazon EKS 控制平面中移除

以下`deregister-cluster`示例取消注册已连接的集群，将其从 Amazon EKS 控制平面中删除。

```
aws eks deregister-cluster \
  --name my-eks-anywhere-cluster
```

输出：

```
{
  "cluster": {
    "name": "my-eks-anywhere-cluster",
    "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-anywhere-cluster",
    "createdAt": "2024-04-12T12:38:37.561000-04:00",
    "status": "DELETING",
    "tags": {},
    "connectorConfig": {
      "activationId": "dfb5ad28-13c3-4e26-8a19-5b2457638c74",
      "activationExpiry": "2024-04-15T12:38:37.082000-04:00",
      "provider": "EKS_ANYWHERE",
      "roleArn": "arn:aws:iam::111122223333:role/AmazonEKSCoordinatorAgentRole"
    }
  }
}
```

有关更多信息，请参阅 Amazon EKS 用户指南中的[注销集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterCluster](#)中的。



## describe-addon-versions

以下代码示例演示了如何使用 describe-addon-versions。

### AWS CLI

示例 1：列出 EKS 集群的所有可用插件

以下 describe-addon-versions 示例列出了所有可用的 AWS 插件。

```
aws eks describe-addon-versions \
  --query 'sort_by(addons &owner)[].{publisher: publisher, owner: owner,
  addonName: addonName, type: type}' \
  --output table
```

输出：

```
-----
|                                     DescribeAddonVersions
|                                     |
+-----+-----+-----+-----+
|                                     |                                     |
|          addonName                 |          owner          |          publisher
|          |          type             |                         |
+-----+-----+-----+-----+
| vpc-cni                             | aws                     | eks
|   | networking                       |                         |
| snapshot-controller                 | aws                     | eks
|   | storage                           |                         |
| kube-proxy                           | aws                     | eks
|   | networking                       |                         |
| eks-pod-identity-agent               | aws                     | eks
|   | security                         |                         |
| coredns                              | aws                     | eks
|   | networking                       |                         |
| aws-mountpoint-s3-csi-driver         | aws                     | s3
|   | storage                           |                         |
| aws-guardduty-agent                  | aws                     | eks
|   | security                         |                         |
| aws-efs-csi-driver                   | aws                     | eks
|   | storage                           |                         |
| aws-ebs-csi-driver                   | aws                     | eks
|   | storage                           |
```

amazon-cloudwatch-observability	aws	eks
observability		
adot	aws	eks
observability		
upwind-security_upwind-operator	aws-marketplace	Upwind Security
security		
upbound_universal-crossplane	aws-marketplace	upbound
infra-management		
tetrade-io_istio-distro	aws-marketplace	tetrade-io
policy-management		
teleport_teleport	aws-marketplace	teleport
policy-management		
stormforge_optimize-live	aws-marketplace	StormForge
cost-management		
splunk_splunk-otel-collector-chart	aws-marketplace	Splunk
monitoring		
solo-io_istio-distro	aws-marketplace	Solo.io
service-mesh		
rafay-systems_rafay-operator	aws-marketplace	rafay-systems
kubernetes-management		
new-relic_kubernetes-operator	aws-marketplace	New Relic
observability		
netapp_trident-operator	aws-marketplace	NetApp Inc.
storage		
leaksignal_leakagent	aws-marketplace	leaksignal
monitoring		
kubecost_kubecost	aws-marketplace	kubecost
cost-management		
kong_konnect-ri	aws-marketplace	kong
ingress-service-type		
kasten_k10	aws-marketplace	Kasten by Veeam
data-protection		
haproxy-technologies_kubernetes-ingress-ee	aws-marketplace	HAProxy
Technologies   ingress-controller		
groundcover_agent	aws-marketplace	groundcover
monitoring		
grafana-labs_kubernetes-monitoring	aws-marketplace	Grafana Labs
monitoring		
factorhouse_kpow	aws-marketplace	factorhouse
monitoring		
dynatrace_dynatrace-operator	aws-marketplace	dynatrace
monitoring		
datree_engine-pro	aws-marketplace	datree
policy-management		

```

| datadog_operator | aws-marketplace | Datadog
|   | monitoring |
| cribl_cribledge | aws-marketplace | Cribl
|   | observability |
| calyptia_fluent-bit | aws-marketplace | Calyptia Inc
|   | observability |
| accuknox_kubearmor | aws-marketplace | AccuKnox
|   | security |
+-----+-----+
+-----+-----+

```

有关更多信息，请参阅 [Amazon EKS 用户指南中的管理 Amazon EKS 插件-创建插件](#)。

示例 2：列出支持 EKS 的指定 Kubernetes 版本的所有可用插件

以下 describe-addon-versions 示例列出了支持 EKS 的指定 Kubernetes 版本的所有可用插件。

```

aws eks describe-addon-versions \
  --kubernetes-version=1.26 \
  --query 'sort_by(addons &owner)[].{publisher: publisher, owner: owner,
addonName: addonName, type: type}' \
  --output table

```

输出：

```

-----
| DescribeAddonVersions
|
+-----+-----+
+-----+-----+
|          addonName          |          owner          |          publisher
|          type              |                          |
+-----+-----+
+-----+-----+
| vpc-cni                    | aws                    | eks
|   | networking              |
| snapshot-controller        | aws                    | eks
|   | storage                  |
| kube-proxy                 | aws                    | eks
|   | networking              |
| eks-pod-identity-agent     | aws                    | eks
|   | security                  |

```

coredns	aws	eks
networking		
aws-mountpoint-s3-csi-driver	aws	s3
storage		
aws-guardduty-agent	aws	eks
security		
aws-efs-csi-driver	aws	eks
storage		
aws-ebs-csi-driver	aws	eks
storage		
amazon-cloudwatch-observability	aws	eks
observability		
adot	aws	eks
observability		
upwind-security_upwind-operator	aws-marketplace	Upwind Security
security		
tetrade-io_istio-distro	aws-marketplace	tetrade-io
policy-management		
stormforge_optimize-live	aws-marketplace	StormForge
cost-management		
splunk_splunk-otel-collector-chart	aws-marketplace	Splunk
monitoring		
solo-io_istio-distro	aws-marketplace	Solo.io
service-mesh		
rafay-systems_rafay-operator	aws-marketplace	rafay-systems
kubernetes-management		
new-relic_kubernetes-operator	aws-marketplace	New Relic
observability		
netapp_trident-operator	aws-marketplace	NetApp Inc.
storage		
leaksignal_leakagent	aws-marketplace	leaksignal
monitoring		
kubecost_kubecost	aws-marketplace	kubecost
cost-management		
kong_konnect-ri	aws-marketplace	kong
ingress-service-type		
haproxy-technologies_kubernetes-ingress-ee	aws-marketplace	HAProxy
Technologies   ingress-controller		
groundcover_agent	aws-marketplace	groundcover
monitoring		
grafana-labs_kubernetes-monitoring	aws-marketplace	Grafana Labs
monitoring		
dynatrace_dynatrace-operator	aws-marketplace	dynatrace
monitoring		

```

| datadog_operator | aws-marketplace | Datadog
|   | monitoring |
| cribl_cribledge | aws-marketplace | Cribl
|   | observability |
| calyptia_fluent-bit | aws-marketplace | Calyptia Inc
|   | observability |
| accuknox_kubearmor | aws-marketplace | AccuKnox
|   | security |
+-----+-----+
+-----+-----+

```

有关更多信息，请参阅 [Amazon EKS 用户指南中的管理 Amazon EKS 插件-创建插件](#)。

示例 3：列出支持 EKS 的指定 Kubernetes 版本的所有可用的 vpc-cni 插件版本

以下 describe-addon-versions 示例列出了支持 EKS 的指定 Kubernetes 版本的所有可用的 vpc-cni 插件版本。

```

aws eks describe-addon-versions \
  --kubernetes-version=1.26 \
  --addon-name=vpc-cni \
  --query='addons[].addonVersions[].addonVersion'

```

输出：

```

[
  "v1.18.0-eksbuild.1",
  "v1.17.1-eksbuild.1",
  "v1.16.4-eksbuild.2",
  "v1.16.3-eksbuild.2",
  "v1.16.2-eksbuild.1",
  "v1.16.0-eksbuild.1",
  "v1.15.5-eksbuild.1",
  "v1.15.4-eksbuild.1",
  "v1.15.3-eksbuild.1",
  "v1.15.1-eksbuild.1",
  "v1.15.0-eksbuild.2",
  "v1.14.1-eksbuild.1",
  "v1.14.0-eksbuild.3",
  "v1.13.4-eksbuild.1",
  "v1.13.3-eksbuild.1",
  "v1.13.2-eksbuild.1",
  "v1.13.0-eksbuild.1",

```

```
"v1.12.6-eksbuild.2",  
"v1.12.6-eksbuild.1",  
"v1.12.5-eksbuild.2",  
"v1.12.0-eksbuild.2"  
]
```

有关更多信息，请参阅 [Amazon EKS 用户指南中的管理 Amazon EKS 插件-创建插件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAddonVersions](#) 中的。

## describe-addon

以下代码示例演示了如何使用 describe-addon。

### AWS CLI

描述在你的 Amazon EKS 集群中积极运行 EKS 插件

以下 describe-addon 示例是在您的 Amazon EKS 集群中主动运行 EKS 插件。

```
aws eks describe-addon \  
  --cluster-name my-eks-cluster \  
  --addon-name vpc-cni
```

输出：

```
{  
  "addon": {  
    "addonName": "vpc-cni",  
    "clusterName": "my-eks-cluster",  
    "status": "ACTIVE",  
    "addonVersion": "v1.16.4-eksbuild.2",  
    "health": {  
      "issues": []  
    },  
    "addonArn": "arn:aws:eks:us-east-2:111122223333:addon/my-eks-cluster/vpc-cni/0ec71efc-98dd-3203-60b0-4b939b2a5e5f",  
    "createdAt": "2024-03-14T13:18:45.417000-04:00",  
    "modifiedAt": "2024-03-14T13:18:49.557000-04:00",  
    "serviceAccountRoleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-addon-vpc-cni-Role1-Yfakrq0C1UTm",  
    "tags": {  
      "eks-addon-key-3": "value-3",  
    }  
  }  
}
```

```

        "eks-addon-key-4": "value-4"
    },
    "configurationValues": "resources:\n    limits:\n        cpu: '100m'\nenv:\n    AWS_VPC_K8S_CNI_LOGLEVEL: 'DEBUG'"
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAddon](#)中的。

## describe-cluster

以下代码示例演示了如何使用 describe-cluster。

### AWS CLI

描述在你的 Amazon EKS 集群中积极运行 EKS 插件

以下describe-cluster示例是在您的 Amazon EKS 集群中主动运行 EKS 插件。

```
aws eks describe-cluster \
    --cluster-name my-eks-cluster
```

输出：

```

{
  "cluster": {
    "name": "my-eks-cluster",
    "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster",
    "createdAt": "2024-03-14T11:31:44.348000-04:00",
    "version": "1.26",
    "endpoint": "https://JSA79429HJDASKJDJ8223829MNDNASW.y14.us-
east-2.eks.amazonaws.com",
    "roleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-cluster-
ServiceRole-zMF6CBakwwbW",
    "resourcesVpcConfig": {
      "subnetIds": [
        "subnet-0fb75d2d8401716e7",
        "subnet-02184492f67a3d0f9",
        "subnet-04098063527aab776",
        "subnet-0e2907431c9988b72",
        "subnet-04ad87f71c6e5ab4d",
        "subnet-09d912bb63ef21b9a"
      ]
    },

```

```
    "securityGroupIds": [
      "sg-0c1327f6270afbb36"
    ],
    "clusterSecurityGroupId": "sg-01c84d09d70f39a7f",
    "vpcId": "vpc-0012b8e1cc0abb17d",
    "endpointPublicAccess": true,
    "endpointPrivateAccess": true,
    "publicAccessCidrs": [
      "22.19.18.2/32"
    ]
  },
  "kubernetesNetworkConfig": {
    "serviceIpv4Cidr": "10.100.0.0/16",
    "ipFamily": "ipv4"
  },
  "logging": {
    "clusterLogging": [
      {
        "types": [
          "api",
          "audit",
          "authenticator",
          "controllerManager",
          "scheduler"
        ],
        "enabled": true
      }
    ]
  },
  "identity": {
    "oidc": {
      "issuer": "https://oidc.eks.us-east-2.amazonaws.com/id/
JSA79429HJDASKJDJ8223829MNDNASW"
    }
  },
  "status": "ACTIVE",
  "certificateAuthority": {
    "data": "CA_DATA_STRING..."
  },
  "platformVersion": "eks.14",
  "tags": {
    "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
    "alpha.eksctl.io/cluster-name": "my-eks-cluster",
    "karpenter.sh/discovery": "my-eks-cluster",
```



```

    "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
    "auto-delete": "no",
    "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
    "EKS-Cluster-Name": "my-eks-cluster",
    "alpha.eksctl.io/cluster-oidc-enabled": "true",
    "aws:cloudformation:logical-id": "ControlPlane",
    "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
    "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
  },
  "health": {
    "issues": []
  },
  "accessConfig": {
    "authenticationMode": "API_AND_CONFIG_MAP"
  }
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCluster](#)中的。

## describe-fargate-profile

以下代码示例演示了如何使用 describe-fargate-profile。

### AWS CLI

描述 Fargate 的个人资料

以下describe-fargate-profile示例描述了 Fargate 配置文件。

```

aws eks describe-fargate-profile \
  --cluster-name my-eks-cluster \
  --fargate-profile-name my-fargate-profile

```

输出：

```

{
  "fargateProfile": {
    "fargateProfileName": "my-fargate-profile",

```

```

    "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/my-
eks-cluster/my-fargate-profile/96c766ce-43d2-f9c9-954c-647334391198",
    "clusterName": "my-eks-cluster",
    "createdAt": "2024-04-11T10:42:52.486000-04:00",
    "podExecutionRoleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-
cluster-farga-FargatePodExecutionRole-1htfAaJdJUE0",
    "subnets": [
      "subnet-09d912bb63ef21b9a",
      "subnet-04ad87f71c6e5ab4d",
      "subnet-0e2907431c9988b72"
    ],
    "selectors": [
      {
        "namespace": "prod*",
        "labels": {
          "labelname*?": "*value1"
        }
      },
      {
        "namespace": "*dev*",
        "labels": {
          "labelname*?": "*value*"
        }
      }
    ],
    "status": "ACTIVE",
    "tags": {
      "eks-fargate-profile-key-2": "value-2",
      "eks-fargate-profile-key-1": "value-1"
    }
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFargateProfile](#)中的。

## describe-identity-provider-config

以下代码示例演示了如何使用 describe-identity-provider-config。

### AWS CLI

描述与您的 Amazon EKS 集群关联的身份提供商配置

以下describe-identity-provider-config示例描述了与您的 Amazon EKS 集群关联的身份提供商配置。

```
aws eks describe-identity-provider-config \  
  --cluster-name my-eks-cluster \  
  --identity-provider-config type=oidc,name=my-identity-provider
```

输出：

```
{  
  "identityProviderConfig": {  
    "oidc": {  
      "identityProviderConfigName": "my-identity-provider",  
      "identityProviderConfigArn": "arn:aws:eks:us-  
east-2:111122223333:identityproviderconfig/my-eks-cluster/oidc/my-identity-  
provider/8ac76722-78e4-cec1-ed76-d49eea058622",  
      "clusterName": "my-eks-cluster",  
      "issuerUrl": "https://oidc.eks.us-east-2.amazonaws.com/  
id/38D6A4619A0A69E342B113ED7F1A7652",  
      "clientId": "kubernetes",  
      "usernameClaim": "email",  
      "usernamePrefix": "my-username-prefix",  
      "groupsClaim": "my-claim",  
      "groupsPrefix": "my-groups-prefix",  
      "requiredClaims": {  
        "Claim1": "value1",  
        "Claim2": "value2"  
      },  
      "tags": {  
        "env": "dev"  
      },  
      "status": "ACTIVE"  
    }  
  }  
}
```

有关更多信息，请参阅 Amazon EKS [用户指南中的通过 OpenID Connect 身份提供商对集群的用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeIdentityProviderConfig](#)中的。

## describe-nodegroup

以下代码示例演示了如何使用 describe-nodegroup。

### AWS CLI

描述 Amazon EKS 集群的托管节点组

以下 describe-nodegroup 示例描述了 Amazon EKS 集群的托管节点组。

```
aws eks describe-nodegroup \  
  --cluster-name my-eks-cluster \  
  --nodegroup-name my-eks-nodegroup
```

输出：

```
{  
  "nodegroup": {  
    "nodegroupName": "my-eks-nodegroup",  
    "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-  
cluster/my-eks-nodegroup/a8c75f2f-df78-a72f-4063-4b69af3de5b1",  
    "clusterName": "my-eks-cluster",  
    "version": "1.26",  
    "releaseVersion": "1.26.12-20240329",  
    "createdAt": "2024-04-08T11:42:10.555000-04:00",  
    "modifiedAt": "2024-04-08T11:44:12.402000-04:00",  
    "status": "ACTIVE",  
    "capacityType": "ON_DEMAND",  
    "scalingConfig": {  
      "minSize": 1,  
      "maxSize": 3,  
      "desiredSize": 1  
    },  
    "instanceTypes": [  
      "t3.medium"  
    ],  
    "subnets": [  
      "subnet-0e2907431c9988b72",  
      "subnet-04ad87f71c6e5ab4d",  
      "subnet-09d912bb63ef21b9a"  
    ],  
    "amiType": "AL2_x86_64",  
    "nodeRole": "arn:aws:iam::111122223333:role/role-name",
```

```
    "labels": {},
    "resources": {
      "autoScalingGroups": [
        {
          "name": "eks-my-eks-nodegroup-a8c75f2f-df78-
a72f-4063-4b69af3de5b1"
        }
      ]
    },
    "diskSize": 20,
    "health": {
      "issues": []
    },
    "updateConfig": {
      "maxUnavailable": 1
    },
    "tags": {}
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeNodegroup](#)中的。

## describe-update

以下代码示例演示了如何使用 describe-update。

### AWS CLI

#### 示例 1：描述集群的更新

以下describe-update示例描述了名为的集群的更新。

```
aws eks describe-update \
  --name my-eks-cluster \
  --update-id 10bddb13-a71b-425a-b0a6-71cd03e59161
```

输出：

```
{
  "update": {
    "id": "10bddb13-a71b-425a-b0a6-71cd03e59161",
    "status": "Successful",
```

```

    "type": "EndpointAccessUpdate",
    "params": [
      {
        "type": "EndpointPublicAccess",
        "value": "false"
      },
      {
        "type": "EndpointPrivateAccess",
        "value": "true"
      }
    ],
    "createdAt": "2024-03-14T10:01:26.297000-04:00",
    "errors": []
  }
}

```

有关更多信息，请参阅亚马逊 EKS 用户指南中的[更新 Amazon EKS 集群 Kubernetes 版本](#)。

## 示例 2：描述集群的更新

以下describe-update示例描述了名为的集群的更新。

```

aws eks describe-update \
  --name my-eks-cluster \
  --update-id e4994991-4c0f-475a-a040-427e6da52966

```

输出：

```

{
  "update": {
    "id": "e4994991-4c0f-475a-a040-427e6da52966",
    "status": "Successful",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\": [\"secrets\"], \"provider\": {\"keyArn\": \"arn:aws:kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": "2024-03-14T11:01:26.297000-04:00",
    "errors": []
  }
}

```

```
}
```

有关更多信息，请参阅亚马逊 EKS 用户指南中的[更新 Amazon EKS 集群 Kubernetes 版本](#)。

### 示例 3：描述集群的更新

以下describe-update示例描述了名为的集群的更新。

```
aws eks describe-update \  
  --name my-eks-cluster \  
  --update-id b5f0ba18-9a87-4450-b5a0-825e6e84496f
```

输出：

```
{  
  "update": {  
    "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",  
    "status": "Successful",  
    "type": "VersionUpdate",  
    "params": [  
      {  
        "type": "Version",  
        "value": "1.29"  
      },  
      {  
        "type": "PlatformVersion",  
        "value": "eks.1"  
      }  
    ],  
    "createdAt": "2024-03-14T12:05:26.297000-04:00",  
    "errors": []  
  }  
}
```

有关更多信息，请参阅亚马逊 EKS 用户指南中的[更新 Amazon EKS 集群 Kubernetes 版本](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeUpdate](#)中的。

## disassociate-identity-provider-config

以下代码示例演示了如何使用 disassociate-identity-provider-config。

## AWS CLI

### 取消身份提供商与您的 Amazon EKS 集群的关联

以下 `disassociate-identity-provider-config` 示例取消身份提供商与您的 Amazon EKS 集群的关联。

```
aws eks disassociate-identity-provider-config \  
  --cluster-name my-eks-cluster \  
  --identity-provider-config 'type=oidc,name=my-identity-provider'
```

输出：

```
{  
  "update": {  
    "id": "5f78d14e-c57b-4857-a3e4-cf664ae20949",  
    "status": "InProgress",  
    "type": "DisassociateIdentityProviderConfig",  
    "params": [  
      {  
        "type": "IdentityProviderConfig",  
        "value": "[]"  
      }  
    ],  
    "createdAt": "2024-04-11T13:53:43.314000-04:00",  
    "errors": []  
  }  
}
```

有关更多信息，请参阅 [Amazon EKS 用户指南中的 OpenID Connect 身份提供商对集群的用户进行身份验证-解除 OIDC 身份提供商与集群的关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateIdentityProviderConfig](#) 中的。

## get-token

以下代码示例演示了如何使用 `get-token`。

## AWS CLI

示例 1：获取名为 `my-eks-cluster` 的 Amazon EKS 集群的身份验证令牌



以下`get-token`示例获取名为的 Amazon EKS 集群的身份验证令牌 `my-eks-cluster`。

```
aws eks get-token \  
  --cluster-name my-eks-cluster
```

输出：

```
{  
  "kind": "ExecCredential",  
  "apiVersion": "client.authentication.k8s.io/v1beta1",  
  "spec": {},  
  "status": {  
    "expirationTimestamp": "2024-04-11T20:59:56Z",  
    "token": "k8s-aws-v1.EXAMPLE_TOKEN_DATA_STRING..."  
  }  
}
```

示例 2：为名为 `my-eks-cluster` 的 Amazon EKS 集群获取身份验证令牌，方法是在签署令牌时假设这个 `roLearn` 作为证书

以下`get-token`示例获取名为 Amazon EKS 集群的身份验证令牌，该令牌在签名 `my-eks-cluster` 令牌时假设使用此 `roLearn` 作为证书。

```
aws eks get-token \  
  --cluster-name my-eks-cluster \  
  --role-arn arn:aws:iam::111122223333:role/eksctl-EKS-Linux-Cluster-v1-24-  
cluster-ServiceRole-j1k7AfTIQtnM
```

输出：

```
{  
  "kind": "ExecCredential",  
  "apiVersion": "client.authentication.k8s.io/v1beta1",  
  "spec": {},  
  "status": {  
    "expirationTimestamp": "2024-04-11T21:05:26Z",  
    "token": "k8s-aws-v1.EXAMPLE_TOKEN_DATA_STRING..."  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetToken](#)中的。

## list-addons

以下代码示例演示了如何使用 `list-addons`。

### AWS CLI

列出您的 Amazon EKS 集群中所有已安装的名为 `my-eks-cluster` 的插件

以下 `list-addons` 示例列出了您的 Amazon EKS 集群中所有已安装的名为 `my-eks-cluster` 的插件。

```
aws eks list-addons \
  --cluster-name my-eks-cluster
```

输出：

```
{
  "addons": [
    "kube-proxy",
    "vpc-cni"
  ]
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAddons](#) 中的。

## list-clusters

以下代码示例演示了如何使用 `list-clusters`。

### AWS CLI

列出您的 Amazon EKS 集群中所有已安装的名为 `my-eks-cluster` 的插件

以下 `list-clusters` 示例列出了您的 Amazon EKS 集群中所有已安装的名为 `my-eks-cluster` 的插件。

```
aws eks list-clusters
```

输出：

```
{
```

```
    "clusters": [  
      "prod",  
      "qa",  
      "stage",  
      "my-eks-cluster"  
    ]  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListClusters](#)中的。

## list-fargate-profiles

以下代码示例演示了如何使用 `list-fargate-profiles`。

### AWS CLI

列出您的 Amazon EKS 集群中名为 `my-eks-cluster` 的所有 fargate 配置文件

以下 `list-fargate-profiles` 示例列出了名为 `my-eks-cluster` 的 Amazon EKS 集群中的所有 fargate 配置文件。

```
aws eks list-fargate-profiles \  
  --cluster-name my-eks-cluster
```

输出：

```
{  
  "fargateProfileNames": [  
    "my-fargate-profile"  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFargateProfiles](#)中的。

## list-identity-provider-configs

以下代码示例演示了如何使用 `list-identity-provider-configs`。

### AWS CLI

列出与 Amazon EKS 集群相关的身份提供商

以下 `list-identity-provider-configs` 示例列出了与 Amazon EKS 集群关联的身份提供商。

```
aws eks list-identity-provider-configs \  
  --cluster-name my-eks-cluster
```

输出：

```
{  
  "identityProviderConfigs": [  
    {  
      "type": "oidc",  
      "name": "my-identity-provider"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon EKS [用户指南中的通过 OpenID Connect 身份提供商对集群的用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListIdentityProviderConfigs](#) 中的。

## list-nodegroups

以下代码示例演示了如何使用 `list-nodegroups`。

AWS CLI

列出 Amazon EKS 集群中的所有节点组

以下 `list-nodegroups` 示例列出了 Amazon EKS 集群中的所有节点组。

```
aws eks list-nodegroups \  
  --cluster-name my-eks-cluster
```

输出：

```
{  
  "nodegroups": [  
    "my-eks-managed-node-group",  
    "my-eks-nodegroup"  
  ]  
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListNodegroups](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

示例 1：列出 Amazon EKS 集群 ARN 的所有标签

以下`list-tags-for-resource`示例列出了 Amazon EKS 集群 ARN 的所有标签。

```
aws eks list-tags-for-resource \
  --resource-arn arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster
```

输出：

```
{
  "tags": {
    "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
    "alpha.eksctl.io/cluster-name": "my-eks-cluster",
    "karpenter.sh/discovery": "my-eks-cluster",
    "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
    "auto-delete": "no",
    "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
    "EKS-Cluster-Name": "my-eks-cluster",
    "alpha.eksctl.io/cluster-oidc-enabled": "true",
    "aws:cloudformation:logical-id": "ControlPlane",
    "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
    "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
  }
}
```

示例 2：列出 Amazon EKS 节点组的所有标签 ARN

以下`list-tags-for-resource`示例列出了 Amazon EKS 节点组 ARN 的所有标签。

```
aws eks list-tags-for-resource \  
  --resource-arn arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-cluster/my-  
eks-managed-node-group/60c71ed2-2cfb-020f-a5f4-ad32477f198c
```

输出：

```
{  
  "tags": {  
    "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-nodegroup-my-eks-  
managed-node-group",  
    "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-  
east-2:111122223333:stack/eksctl-my-eks-cluster-nodegroup-my-eks-managed-node-group/  
eaa20310-e219-11ee-b851-0ab9ad8228ff",  
    "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",  
    "EKS-Cluster-Name": "my-eks-cluster",  
    "alpha.eksctl.io/nodegroup-type": "managed",  
    "NodeGroup Name 1": "my-eks-managed-node-group",  
    "k8s.io/cluster-autoscaler/enabled": "true",  
    "nodegroup-role": "worker",  
    "alpha.eksctl.io/cluster-name": "my-eks-cluster",  
    "alpha.eksctl.io/nodegroup-name": "my-eks-managed-node-group",  
    "karpenter.sh/discovery": "my-eks-cluster",  
    "NodeGroup Name 2": "AmazonLinux-Linux-Managed-NG-v1-26-v1",  
    "auto-delete": "no",  
    "k8s.io/cluster-autoscaler/my-eks-cluster": "owned",  
    "aws:cloudformation:logical-id": "ManagedNodeGroup",  
    "alpha.eksctl.io/eksctl-version": "0.173.0-dev  
+a7ee89342.2024-03-01T03:40:57Z"  
  }  
}
```

示例 3：列出 Amazon EKS Fargate 个人资料上的所有标签 arnE

以下list-tags-for-resource示例列出了亚马逊 EKS Fargate 个人资料 ARN 的所有标签。

```
aws eks list-tags-for-resource \  
  --resource-arn arn:aws:eks:us-east-2:111122223333:fargateprofile/my-eks-cluster/  
my-fargate-profile/d6c76780-e541-0725-c816-36754cab734b
```

输出：

```
{
```

```
    "tags": {
      "eks-fargate-profile-key-2": "value-2",
      "eks-fargate-profile-key-1": "value-1"
    }
  }
```

示例 4：列出 Amazon EKS 附加组件 ARN 的所有标签

以下 `list-tags-for-resource` 示例列出了亚马逊 EKS 附加组件 ARN 的所有标签。

```
aws eks list-tags-for-resource \
  --resource-arn arn:aws:eks:us-east-2:111122223333:addon/my-eks-cluster/vpc-
cni/0ec71efc-98dd-3203-60b0-4b939b2a5e5f
```

输出：

```
{
  "tags": {
    "eks-addon-key-2": "value-2",
    "eks-addon-key-1": "value-1"
  }
}
```

示例 5：列出 Amazon EKS OIDC 身份提供商 ARN 的所有标签

以下 `list-tags-for-resource` 示例列出了 Amazon EKS OIDC 身份提供商 ARN 的所有标签。

```
aws eks list-tags-for-resource \
  --resource-arn arn:aws:eks:us-east-2:111122223333:identityproviderconfig/my-eks-
cluster/oidc/my-identity-provider/8ac76722-78e4-cec1-ed76-d49eea058622
```

输出：

```
{
  "tags": {
    "my-identity-provider": "test"
  }
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## list-update

以下代码示例演示了如何使用 `list-update`。

### AWS CLI

示例 1：列出与 Amazon EKS 集群名称相关的更新

以下 `list-updates` 示例列出了 Amazon EKS 集群名称的所有更新 ID。

```
aws eks list-updates \  
  --name my-eks-cluster
```

输出：

```
{  
  "updateIds": [  
    "5f78d14e-c57b-4857-a3e4-cf664ae20949",  
    "760e5a3f-adad-48c7-88d3-7ac283c09c26",  
    "cd4ec863-bc55-47d5-a377-3971502f529b",  
    "f12657ce-e869-4f17-b158-a82ab8b7d937"  
  ]  
}
```

示例 2：列出 Amazon EKS 节点组的所有更新 ID

以下 `list-updates` 示例列出了 Amazon EKS 节点组的所有更新 ID。

```
aws eks list-updates \  
  --name my-eks-cluster \  
  --nodegroup-name my-eks-managed-node-group
```

输出：

```
{  
  "updateIds": [  
    "8c6c1bef-61fe-42ac-a242-89412387b8e7"  
  ]  
}
```

示例 3：列出 Amazon EKS Add-One 上的所有更新 ID



以下`list-updates`示例列出了 Amazon EKS 附加组件的所有更新 ID。

```
aws eks list-updates \  
  --name my-eks-cluster \  
  --addon-name vpc-cni
```

输出：

```
{  
  "updateIds": [  
    "9cdba8d4-79fb-3c83-afe8-00b508d33268"  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUpdate](#)中的。

## list-updates

以下代码示例演示了如何使用 `list-updates`。

### AWS CLI

#### 列出集群的更新

此示例命令列出了在您的默认区域`example`中命名的集群的当前更新。

命令：

```
aws eks list-updates --name example
```

输出：

```
{  
  "updateIds": [  
    "10bddb13-a71b-425a-b0a6-71cd03e59161"  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUpdates](#)中的。

## register-cluster

以下代码示例演示了如何使用 register-cluster。

### AWS CLI

示例 1：向亚马逊 EKS 注册外部 EKS\_ANYWHERE Kubernetes 集群

以下register-cluster示例将外部 EKS\_ANYWHERE Kubernetes 集群注册到亚马逊 EKS。

```
aws eks register-cluster \  
  --name my-eks-anywhere-cluster \  
  --connector-config 'roleArn=arn:aws:iam::111122223333:role/  
AmazonEKSCoordinatorAgentRole,provider=EKS_ANYWHERE'
```

输出：

```
{  
  "cluster": {  
    "name": "my-eks-anywhere-cluster",  
    "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-anywhere-cluster",  
    "createdAt": "2024-04-12T12:38:37.561000-04:00",  
    "status": "PENDING",  
    "tags": {},  
    "connectorConfig": {  
      "activationId": "xxxxxxxxACTIVATION_IDxxxxxxxx",  
      "activationCode": "xxxxxxxxACTIVATION_CODExxxxxxxx",  
      "activationExpiry": "2024-04-15T12:38:37.082000-04:00",  
      "provider": "EKS_ANYWHERE",  
      "roleArn": "arn:aws:iam::111122223333:role/AmazonEKSCoordinatorAgentRole"  
    }  
  }  
}
```

有关更多信息，请参阅 Amazon EKS 用户指南中的[连接外部集群](#)。

示例 2：将任何外部 Kubernetes 集群注册到亚马逊 EKS

以下register-cluster示例将外部 EKS\_ANYWHERE Kubernetes 集群注册到亚马逊 EKS。

```
aws eks register-cluster \  
  --name my-eks-anywhere-cluster \  
  --connector-config 'roleArn=arn:aws:iam::111122223333:role/  
AmazonEKSCoordinatorAgentRole,provider=EKS_ANYWHERE'
```

```
--connector-config 'roleArn=arn:aws:iam::111122223333:role/
AmazonEKSCollectorAgentRole,provider=OTHER'
```

输出：

```
{
  "cluster": {
    "name": "my-onprem-k8s-cluster",
    "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-onprem-k8s-cluster",
    "createdAt": "2024-04-12T12:42:10.861000-04:00",
    "status": "PENDING",
    "tags": {},
    "connectorConfig": {
      "activationId": "xxxxxxxxACTIVATION_IDxxxxxxxx",
      "activationCode": "xxxxxxxxACTIVATION_CODExxxxxxxx",
      "activationExpiry": "2024-04-15T12:42:10.339000-04:00",
      "provider": "OTHER",
      "roleArn": "arn:aws:iam::111122223333:role/AmazonEKSCollectorAgentRole"
    }
  }
}
```

有关更多信息，请参阅 Amazon EKS 用户指南中的[连接外部集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RegisterCluster](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

示例 1：向 Amazon EKS 集群添加指定标签

以下 tag-resource 示例将指定的标签添加到 Amazon EKS 集群。

```
aws eks tag-resource \
  --resource-arn arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster \
  --tag 'my-eks-cluster-test-1=test-value-1,my-eks-cluster-dev-1=dev-value-2'
```

此命令不生成任何输出。

## 示例 2：向 Amazon EKS 节点组添加指定标签

以下 `tag-resource` 示例将指定的标签添加到 Amazon EKS 节点组。

```
aws eks tag-resource \  
  --resource-arn arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-cluster/my-  
eks-managed-node-group/60c71ed2-2cfb-020f-a5f4-ad32477f198c \  
  --tag 'my-eks-nodegroup-test-1=test-value-1,my-eks-nodegroup-dev-1=dev-value-2'
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 `untag-resource`。

### AWS CLI

#### 示例 1：从 Amazon EKS 集群中删除指定的标签

以下 `untag-resource` 示例从 Amazon EKS 集群中删除指定的标签。

```
aws eks untag-resource \  
  --resource-arn arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster \  
  --tag-keys "my-eks-cluster-test-1" "my-eks-cluster-dev-1"
```

此命令不生成任何输出。

#### 示例 2：从 Amazon EKS 节点组中删除指定的标签

以下 `untag-resource` 示例从 Amazon EKS 节点组中删除指定的标签。

```
aws eks untag-resource \  
  --resource-arn arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-cluster/my-  
eks-managed-node-group/60c71ed2-2cfb-020f-a5f4-ad32477f198c \  
  --tag-keys "my-eks-nodegroup-test-1" "my-eks-nodegroup-dev-1"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-addon

以下代码示例演示了如何使用 update-addon。

### AWS CLI

示例 1。使用服务账户角色更新 Amazon EKS 附加组件 ARN

以下update-addon示例命令使用服务账户角色 ARN 更新 Amazon EKS 附加组件。

```
aws eks update-addon \  
  --cluster-name my-eks-cluster \  
  --addon-name vpc-cni \  
  --service-account-role-arn arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-  
addon-vpc-cni-Role1-Yfakrq0C1UTm
```

输出：

```
{  
  "update": {  
    "id": "c00d2de2-c2e4-3d30-929e-46b8edec2ce4",  
    "status": "InProgress",  
    "type": "AddonUpdate",  
    "params": [  
      {  
        "type": "ServiceAccountRoleArn",  
        "value": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-  
addon-vpc-cni-Role1-Yfakrq0C1UTm"  
      }  
    ],  
    "updatedAt": "2024-04-12T16:04:55.614000-04:00",  
    "errors": []  
  }  
}
```

有关更多信息，请参阅 [Amazon EKS 用户指南中的管理 Amazon EKS 插件-更新插件](#)。

示例 2。使用特定附加组件版本更新 Amazon EKS 附加组件

以下update-addon示例命令使用特定的附加组件版本更新 Amazon EKS 附加组件。

```
aws eks update-addon \  
  --cluster-name my-eks-cluster \  
  --addon-name vpc-cni \  
  --addon-version vpc-cni-1.10.0-1.0
```

```
--cluster-name my-eks-cluster \  
--addon-name vpc-cni \  
--service-account-role-arn arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-  
addon-vpc-cni-Role1-Yfakrq0C1UTm \  
--addon-version v1.16.4-eksbuild.2
```

输出：

```
{  
  "update": {  
    "id": "f58dc0b0-2b18-34bd-bc6a-e4abc0011f36",  
    "status": "InProgress",  
    "type": "AddonUpdate",  
    "params": [  
      {  
        "type": "AddonVersion",  
        "value": "v1.16.4-eksbuild.2"  
      },  
      {  
        "type": "ServiceAccountRoleArn",  
        "value": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-  
addon-vpc-cni-Role1-Yfakrq0C1UTm"  
      }  
    ],  
    "createdAt": "2024-04-12T16:07:16.550000-04:00",  
    "errors": []  
  }  
}
```

有关更多信息，请参阅 [Amazon EKS 用户指南中的管理 Amazon EKS 插件-更新](#) 插件。

示例 3。使用自定义配置值更新 Amazon EKS 插件并解决冲突详情

以下 update-addon 示例命令使用自定义配置值更新 Amazon EKS 插件并解决冲突的详细信息。

```
aws eks update-addon \  
  --cluster-name my-eks-cluster \  
  --addon-name vpc-cni \  
  --service-account-role-arn arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-  
addon-vpc-cni-Role1-Yfakrq0C1UTm \  
  --addon-version v1.16.4-eksbuild.2 \  
  --configuration-values '{"resources": {"limits":{"cpu":"100m"}, "requests":  
{"cpu":"50m"}}}' \  

```

```
--resolve-conflicts PRESERVE
```

输出：

```
{
  "update": {
    "id": "cd9f2173-a8d8-3004-a90f-032f14326520",
    "status": "InProgress",
    "type": "AddonUpdate",
    "params": [
      {
        "type": "AddonVersion",
        "value": "v1.16.4-eksbuild.2"
      },
      {
        "type": "ServiceAccountRoleArn",
        "value": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-
addon-vpc-cni-Role1-Yfakrq0C1UTm"
      },
      {
        "type": "ResolveConflicts",
        "value": "PRESERVE"
      },
      {
        "type": "ConfigurationValues",
        "value": "{\"resources\": {\"limits\": {\"cpu\": \"100m\"}, \"requests
\": {\"cpu\": \"50m\"}}}"
      }
    ],
    "createdAt": "2024-04-12T16:16:27.363000-04:00",
    "errors": []
  }
}
```

有关更多信息，请参阅 [Amazon EKS 用户指南中的管理 Amazon EKS 插件-更新](#) 插件。

示例 4。使用自定义 JSON 配置值文件更新 Amazon EKS 插件

以下 update-addon 示例命令使用自定义 JSON 配置值更新 Amazon EKS 插件并解决冲突详细信息。

```
aws eks update-addon \
```

```
--cluster-name my-eks-cluster \  
--addon-name vpc-cni \  
--service-account-role-arn arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-  
addon-vpc-cni-Role1-Yfakrq0C1UTm \  
--addon-version v1.17.1-eksbuild.1 \  
--configuration-values 'file://configuration-values.json' \  
--resolve-conflicts PRESERVE
```

configuration-values.json 的内容：

```
{  
  "resources": {  
    "limits": {  
      "cpu": "100m"  
    },  
    "requests": {  
      "cpu": "50m"  
    }  
  },  
  "env": {  
    "AWS_VPC_K8S_CNI_LOGLEVEL": "ERROR"  
  }  
}
```

输出：

```
{  
  "update": {  
    "id": "6881a437-174f-346b-9a63-6e91763507cc",  
    "status": "InProgress",  
    "type": "AddonUpdate",  
    "params": [  
      {  
        "type": "AddonVersion",  
        "value": "v1.17.1-eksbuild.1"  
      },  
      {  
        "type": "ServiceAccountRoleArn",  
        "value": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-  
addon-vpc-cni-Role1-Yfakrq0C1UTm"  
      },  
      {  
        "type": "ResolveConflicts",
```



```

        "value": "PRESERVE"
      },
      {
        "type": "ConfigurationValues",
        "value": "{\n  \"resources\": {\n    \"limits\": {\n      \"cpu\": \"100m\"\n    },\n    \"requests\": {\n      \"cpu\": \"50m\"\n    }\n  },\n  \"env\": {\n    \"AWS_VPC_K8S_CNI_LOGLEVEL\": \"ERROR\"\n  }\n}"
      }
    ],
    "createdAt": "2024-04-12T16:22:55.519000-04:00",
    "errors": []
  }
}

```

有关更多信息，请参阅 [Amazon EKS 用户指南中的管理 Amazon EKS 插件-更新插件](#)。

示例 5。使用自定义 YAML 配置值文件更新 Amazon EKS 附加组件

以下 update-addon 示例命令使用自定义 YAML 配置值更新 Amazon EKS 插件并解决冲突的详细信息。

```

aws eks update-addon \
  --cluster-name my-eks-cluster \
  --addon-name vpc-cni \
  --service-account-role-arn arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-addon-vpc-cni-Role1-Yfakrq0C1UTm \
  --addon-version v1.18.0-eksbuild.1 \
  --configuration-values 'file://configuration-values.yaml' \
  --resolve-conflicts PRESERVE

```

configuration-values.yaml 的内容：

```

resources:
  limits:
    cpu: '100m'
  requests:
    cpu: '50m'
env:
  AWS_VPC_K8S_CNI_LOGLEVEL: 'DEBUG'

```

输出：

```
{
  "update": {
    "id": "a067a4c9-69d0-3769-ace9-d235c5b16701",
    "status": "InProgress",
    "type": "AddonUpdate",
    "params": [
      {
        "type": "AddonVersion",
        "value": "v1.18.0-eksbuild.1"
      },
      {
        "type": "ServiceAccountRoleArn",
        "value": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-
addon-vpc-cni-Role1-Yfakrq0C1UTm"
      },
      {
        "type": "ResolveConflicts",
        "value": "PRESERVE"
      },
      {
        "type": "ConfigurationValues",
        "value": "resources:\n      limits:\n          cpu: '100m'\n
requests:\n      cpu: '50m'\nenv:\n      AWS_VPC_K8S_CNI_LOGLEVEL: 'DEBUG'"
      }
    ],
    "createdAt": "2024-04-12T16:25:07.212000-04:00",
    "errors": []
  }
}
```

有关更多信息，请参阅 [Amazon EKS 用户指南中的管理 Amazon EKS 插件-更新](#) 插件。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateAddon](#) 中的。

## update-cluster-config

以下代码示例演示了如何使用 `update-cluster-config`。

### AWS CLI

#### 更新集群终端节点访问权限

此示例命令更新集群以禁用终端节点公共访问并启用私有终端节点访问。

**命令:**

```
aws eks update-cluster-config --name example \  
--resources-vpc-config endpointPublicAccess=false,endpointPrivateAccess=true
```

**输出:**

```
{  
  "update": {  
    "id": "ec883c93-2e9e-407c-a22f-8f6fa6e67d4f",  
    "status": "InProgress",  
    "type": "EndpointAccessUpdate",  
    "params": [  
      {  
        "type": "EndpointPublicAccess",  
        "value": "false"  
      },  
      {  
        "type": "EndpointPrivateAccess",  
        "value": "true"  
      }  
    ],  
    "createdAt": 1565806986.506,  
    "errors": []  
  }  
}
```

**为集群启用日志记录**

此示例命令为名为的集群启用所有集群控制平面日志记录类型example。

**命令:**

```
aws eks update-cluster-config --name example \  
--logging '{"clusterLogging":[{"types":  
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

**输出:**

```
{  
  "update": {
```

```

    "id": "7551c64b-1d27-4b1e-9f8e-c45f056eb6fd",
    "status": "InProgress",
    "type": "LoggingUpdate",
    "params": [
      {
        "type": "ClusterLogging",
        "value": "{\"clusterLogging\": [{\"types\": [\"api\", \"audit\",
\\authenticator\", \"controllerManager\", \"scheduler\"], \"enabled\": true}]}"
      }
    ],
    "createdAt": 1565807210.37,
    "errors": []
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateClusterConfig](#)中的。

## update-cluster-version

以下代码示例演示了如何使用 `update-cluster-version`。

### AWS CLI

将名为 `my-eks-cluster` 的 Amazon EKS 集群更新到指定的 Kubernetes 版本

以下 `update-cluster-version` 示例将 Amazon EKS 集群更新到指定的 Kubernetes 版本。

```

aws eks update-cluster-version \
  --name my-eks-cluster \
  --kubernetes-version 1.27

```

输出：

```

{
  "update": {
    "id": "e4091a28-ea14-48fd-a8c7-975aeb469e8a",
    "status": "InProgress",
    "type": "VersionUpdate",
    "params": [
      {
        "type": "Version",
        "value": "1.27"
      }
    ]
  }
}

```

```
    },
    {
      "type": "PlatformVersion",
      "value": "eks.16"
    }
  ],
  "createdAt": "2024-04-12T16:56:01.082000-04:00",
  "errors": []
}
}
```

有关更多信息，请参阅亚马逊 EKS 用户指南中的[更新 Amazon EKS 集群 Kubernetes 版本](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateClusterVersion](#)中的。

## update-kubeconfig

以下代码示例演示了如何使用 update-kubeconfig。

### AWS CLI

示例 1：通过创建或更新 kubeconfig 来配置你的 kubectl，这样你就可以连接到名为 `` 的 Amazon EKS 集群 my-eks-cluster

以下 update-kubeconfig 示例通过创建或更新 kubeconfig 来配置您的 kubectl，以便您可以连接到名为的 Amazon EKS 集群。 my-eks-cluster

```
aws eks update-kubeconfig \  
  --name my-eks-cluster
```

输出：

```
Updated context arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster in /Users/  
xxx/.kube/config
```

有关更多信息，请参阅 Amazon EKS 用户指南中的[为亚马逊 EKS 集群创建或更新 kubeconfig 文件](#)。

示例 2：通过创建或更新 kubeconfig ( 使用 role-arn 选项代入集群身份验证角色 ) 来配置你的 kubectl，这样你就可以连接到名为 `` 的 Amazon EKS 集群 my-eks-cluster

以下 `update-kubeconfig` 示例通过创建或更新 `kubeconfig` ( 使用 `role-arn` 选项代入集群身份验证角色 ) 来配置您的 `kubectl` , 以便您可以连接到名为的 Amazon EKS 集群。 `my-eks-cluster`

```
aws eks update-kubeconfig \  
  --name my-eks-cluster \  
  --role-arn arn:aws:iam::111122223333:role/eksctl-EKS-Linux-Cluster-v1-24-  
cluster-ServiceRole-j1k7AfTIQtnM
```

输出 :

```
Updated context arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster in /Users/  
xxx/.kube/config
```

有关更多信息 , 请参阅 Amazon EKS [用户指南中的为亚马逊 EKS 集群创建或更新 kubeconfig 文件](#)。

示例 3 : 通过创建或更新 `kubeconfig` ( 使用 `role-arn` 选项来代入集群身份验证角色以及自定义集群别名和用户别名 ) 来配置你的 `kubectl` , 这样你就可以连接到名为 `` 的 Amazon EKS 集群 `my-eks-cluster`

以下 `update-kubeconfig` 示例通过创建或更新 `kubeconfig` ( 使用 `role-arn` 选项来代入集群身份验证角色以及自定义集群别名和用户别名 ) 来配置您的 `kubectl` , 以便您可以连接到名为的 Amazon EKS 集群。 `my-eks-cluster`

```
aws eks update-kubeconfig \  
  --name my-eks-cluster \  
  --role-arn arn:aws:iam::111122223333:role/eksctl-EKS-Linux-Cluster-v1-24-  
cluster-ServiceRole-j1k7AfTIQtnM \  
  --alias stage-eks-cluster \  
  --user-alias john
```

输出 :

```
Updated context stage-eks-cluster in /Users/dubaria/.kube/config
```

有关更多信息 , 请参阅 Amazon EKS [用户指南中的为亚马逊 EKS 集群创建或更新 kubeconfig 文件](#)。

示例 4 : 打印 `kubeconfig` 文件条目以供查看并配置你的 `kubectl` , 这样你就可以连接到名为 `` 的亚马逊 EKS 集群 `my-eks-cluster`

以下 update-kubeconfig 示例通过创建或更新 kubeconfig ( 使用 role-arn 选项来代入集群身份验证角色以及自定义集群别名和用户别名 ) 来配置您的 kubectl , 以便您可以连接到名为的 Amazon EKS 集群。 my-eks-cluster

```
aws eks update-kubeconfig \  
  --name my-eks-cluster \  
  --role-arn arn:aws:iam::111122223333:role/eksctl-EKS-Linux-Cluster-v1-24-  
cluster-ServiceRole-j1k7AfTIQtnM \  
  --alias stage-eks-cluster \  
  --user-alias john \  
  --verbose
```

输出 :

```
Updated context stage-eks-cluster in /Users/dubaria/.kube/config  
Entries:  
  
context:  
cluster: arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster  
user: john  
name: stage-eks-cluster  
  
name: john  
user:  
exec:  
  apiVersion: client.authentication.k8s.io/v1beta1  
  args:  
  - --region  
  - us-east-2  
  - eks  
  - get-token  
  - --cluster-name  
  - my-eks-cluster  
  - --output  
  - json  
  - --role  
  - arn:aws:iam::111122223333:role/eksctl-EKS-Linux-Cluster-v1-24-cluster-  
ServiceRole-j1k7AfTIQtnM  
  command: aws  
  
cluster:  
certificate-authority-data: xxx_CA_DATA_xxx  
server: https://DALSJ343KE23J3RN45653DSKJTT647TYD.y14.us-east-2.eks.amazonaws.com
```

```
name: arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster
```

有关更多信息，请参阅 Amazon EKS [用户指南中的为亚马逊 EKS 集群创建或更新 kubeconfig 文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateKubeconfig](#) 中的。

## update-nodegroup-config

以下代码示例演示了如何使用 update-nodegroup-config。

### AWS CLI

示例 1：更新托管节点组以向 Amazon EKS 集群的 EKS 工作节点添加新标签和污点

以下 update-nodegroup-config 示例更新托管节点组，以向 Amazon EKS 集群的 EKS 工作节点添加新的标签和污点。

```
aws eks update-nodegroup-config \
  --cluster-name my-eks-cluster \
  --nodegroup-name my-eks-nodegroup \
  --labels 'addOrUpdateLabels={my-eks-nodegroup-label-1=value-1,my-eks-nodegroup-label-2=value-2}' \
  --taints 'addOrUpdateTaints=[{key=taint-key-1,value=taint-value-1,effect=NO_EXECUTE}]'
```

输出：

```
{
  "update": {
    "id": "e66d21d3-bd8b-3ad1-a5aa-b196dc08c7c1",
    "status": "InProgress",
    "type": "ConfigUpdate",
    "params": [
      {
        "type": "LabelsToAdd",
        "value": "{\"my-eks-nodegroup-label-2\":\"value-2\",\"my-eks-nodegroup-label-1\":\"value-1\"}"
      },
      {
        "type": "TaintsToAdd",
        "value": "[{\"effect\":\"NO_EXECUTE\",\"value\":\"taint-value-1\", \"key\":\"taint-key-1\"}]"
      }
    ]
  }
}
```



```

    }
  ],
  "createdAt": "2024-04-08T12:05:19.161000-04:00",
  "errors": []
}
}

```

有关更多信息，请参阅 Amazon EKS 用户指南中的[更新托管节点组](#)。

示例 2：更新托管节点组以移除 Amazon EKS 集群的 EKS 工作节点的标签和污点

以下 update-nodegroup-config 示例更新托管节点组，以移除 Amazon EKS 集群的 EKS 工作节点的标签和污点。

```

aws eks update-nodegroup-config \
  --cluster-name my-eks-cluster \
  --nodegroup-name my-eks-nodegroup \
  --labels 'removeLabels=my-eks-nodegroup-label-1,my-eks-nodegroup-label-2' \
  --taints 'removeTaints=[{key=taint-key-1,value=taint-value-1,effect=NO_EXECUTE}]'

```

输出：

```

{
  "update": {
    "id": "67a08692-9e59-3ace-a916-13929f44cec3",
    "status": "InProgress",
    "type": "ConfigUpdate",
    "params": [
      {
        "type": "LabelsToRemove",
        "value": "[\"my-eks-nodegroup-label-1\", \"my-eks-nodegroup-label-2\"]"
      },
      {
        "type": "TaintsToRemove",
        "value": "[{\"effect\": \"NO_EXECUTE\", \"value\": \"taint-value-1\", \"key\": \"taint-key-1\"}]"
      }
    ],
    "createdAt": "2024-04-08T12:17:31.817000-04:00",
    "errors": []
  }
}

```

```
}

```

有关更多信息，请参阅 Amazon EKS 用户指南中的[更新托管节点组](#)。

示例 3：更新托管节点组以移除和添加适用于 Amazon EKS 集群的 EKS 工作节点的标签和污点

以下update-nodegroup-config示例更新托管节点组，以便为 Amazon EKS 集群的 EKS 工作节点移除和添加标签和污点。

```
aws eks update-nodegroup-config \
  --cluster-name my-eks-cluster \
  --nodegroup-name my-eks-nodegroup \
  --labels 'addOrUpdateLabels={my-eks-nodegroup-new-label-1=new-value-1,my-eks-
nodegroup-new-label-2=new-value-2},removeLabels=my-eks-nodegroup-label-1, my-eks-
nodegroup-label-2' \
  --taints 'addOrUpdateTaints=[{key=taint-new-key-1,value=taint-new-
value-1,effect=PREFER_NO_SCHEDULE}],removeTaints=[{key=taint-key-1,value=taint-
value-1,effect=NO_EXECUTE}]'
```

输出：

```
{
  "update": {
    "id": "4a9c8c45-6ac7-3115-be71-d6412a2339b7",
    "status": "InProgress",
    "type": "ConfigUpdate",
    "params": [
      {
        "type": "LabelsToAdd",
        "value": "{\"my-eks-nodegroup-new-label-1\":\"new-value-1\",\"my-
eks-nodegroup-new-label-2\":\"new-value-2\"}"
      },
      {
        "type": "LabelsToRemove",
        "value": "[\"my-eks-nodegroup-label-1\",\"my-eks-nodegroup-
label-2\"]"
      },
      {
        "type": "TaintsToAdd",
        "value": "[{\"effect\":\"PREFER_NO_SCHEDULE\",\"value\":\"taint-new-
value-1\",\"key\":\"taint-new-key-1\"}]"
      },
    ],
  },
}
```

```

    {
      "type": "TaintsToRemove",
      "value": "[{\"effect\":\"NO_EXECUTE\",\"value\":\"taint-value-1\"},
    {\"key\":\"taint-key-1\"}]"
    }
  ],
  "createdAt": "2024-04-08T12:30:55.486000-04:00",
  "errors": []
}
}

```

有关更多信息，请参阅 Amazon EKS 用户指南中的[更新托管节点组](#)。

示例 4：更新托管节点组以更新 Amazon EKS 集群的 EKS 工作节点的扩展配置和更新配置

以下update-nodegroup-config示例更新托管节点组以更新 Amazon EKS 集群的 EKS 工作节点的扩展配置和更新配置。

```

aws eks update-nodegroup-config \
  --cluster-name my-eks-cluster \
  --nodegroup-name my-eks-nodegroup \
  --scaling-config minSize=1,maxSize=5,desiredSize=2 \
  --update-config maxUnavailable=2

```

输出：

```

{
  "update": {
    "id": "a977160f-59bf-3023-805d-c9826e460aea",
    "status": "InProgress",
    "type": "ConfigUpdate",
    "params": [
      {
        "type": "MinSize",
        "value": "1"
      },
      {
        "type": "MaxSize",
        "value": "5"
      },
      {
        "type": "DesiredSize",

```

```

        "value": "2"
      },
      {
        "type": "MaxUnavailable",
        "value": "2"
      }
    ],
    "createdAt": "2024-04-08T12:35:17.036000-04:00",
    "errors": []
  }
}

```

有关更多信息，请参阅 Amazon EKS 用户指南中的[更新托管节点组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateNodegroupConfig](#) 中的。

## update-nodegroup-version

以下代码示例演示了如何使用 update-nodegroup-version。

### AWS CLI

示例 1：更新亚马逊 EKS 托管节点组的 Kubernetes 版本或 AMI 版本

以下 update-nodegroup-version 示例将 Amazon EKS 托管节点组的 Kubernetes 版本或 AMI 版本更新为适用于您的 Kubernetes 集群的最新可用版本。

```

aws eks update-nodegroup-version \
  --cluster-name my-eks-cluster \
  --nodegroup-name my-eks-nodegroup \
  --no-force

```

输出：

```

{
  "update": {
    "id": "a94ebfc3-6bf8-307a-89e6-7dbaa36421f7",
    "status": "InProgress",
    "type": "VersionUpdate",
    "params": [
      {
        "type": "Version",

```

```
        "value": "1.26"
      },
      {
        "type": "ReleaseVersion",
        "value": "1.26.12-20240329"
      }
    ],
    "createdAt": "2024-04-08T13:16:00.724000-04:00",
    "errors": []
  }
}
```

有关更多信息，请参阅 Amazon EKS 用户指南中的[更新托管节点组](#)。

示例 2：更新亚马逊 EKS 托管节点组的 Kubernetes 版本或 AMI 版本

以下 update-nodegroup-version 示例将 Amazon EKS 托管节点组的 Kubernetes 版本或 AMI 版本更新为指定的 AMI 发行版本。

```
aws eks update-nodegroup-version \
  --cluster-name my-eks-cluster \
  --nodegroup-name my-eks-nodegroup \
  --kubernetes-version '1.26' \
  --release-version '1.26.12-20240307' \
  --no-force
```

输出：

```
{
  "update": {
    "id": "4db06fe1-088d-336b-bdcd-3fdb94995fb7",
    "status": "InProgress",
    "type": "VersionUpdate",
    "params": [
      {
        "type": "Version",
        "value": "1.26"
      },
      {
        "type": "ReleaseVersion",
        "value": "1.26.12-20240307"
      }
    ]
  }
}
```

```
    ],  
    "createdAt": "2024-04-08T13:13:58.595000-04:00",  
    "errors": []  
  }  
}
```

有关更多信息，请参阅 Amazon EKS 用户指南 `update-managed-node-group` 中的更新托管节点组-  
< <https://docs.aws.amazon.com/eks/latest/userguide/> .html>`。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateNodegroupVersion](#) 中的。

## 使用 Elastic Beanstalk 示例 AWS CLI

以下代码示例向您展示了如何在 Elastic Beanstalk 中 AWS Command Line Interface 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **abort-environment-update**

以下代码示例演示了如何使用 `abort-environment-update`。

#### AWS CLI

##### 中止部署

以下命令中止名 `my-env` 为的环境的正在运行的应用程序版本部署：

```
aws elasticbeanstalk abort-environment-update --environment-name my-env
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AbortEnvironmentUpdate](#)中的。

## check-dns-availability

以下代码示例演示了如何使用 `check-dns-availability`。

### AWS CLI

检查别名记录的可用性

以下命令检查子域 `my-cname.elasticbeanstalk.com` 的可用性：

```
aws elasticbeanstalk check-dns-availability --cname-prefix my-cname
```

输出：

```
{
  "Available": true,
  "FullyQualifiedCNAME": "my-cname.elasticbeanstalk.com"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CheckDnsAvailability](#)中的。

## create-application-version

以下代码示例演示了如何使用 `create-application-version`。

### AWS CLI

创建新的应用程序版本

以下命令创建名为 “” 的应用程序的新版本 “v1” MyApp：

```
aws elasticbeanstalk create-application-version --application-name MyApp
--version-label v1 --description MyAppv1 --source-bundle S3Bucket="my-
bucket",S3Key="sample.war" --auto-create-application
```

由于该 `auto-create-application` 选项，如果应用程序尚不存在，则会自动创建。源包是一个 `.war` 文件，存储在名为 “my-bucket” 的 s3 存储桶中，其中包含 Apache Tomcat 示例应用程序。

输出：

```
{
  "ApplicationVersion": {
    "ApplicationName": "MyApp",
    "VersionLabel": "v1",
    "Description": "MyAppv1",
    "DateCreated": "2015-02-03T23:01:25.412Z",
    "DateUpdated": "2015-02-03T23:01:25.412Z",
    "SourceBundle": {
      "S3Bucket": "my-bucket",
      "S3Key": "sample.war"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateApplicationVersion](#)中的。

## create-application

以下代码示例演示了如何使用 create-application。

### AWS CLI

#### 创建新应用程序

以下命令创建一个名为 MyApp 的新应用程序：

```
aws elasticbeanstalk create-application --application-name MyApp --description "my
application"
```

该create-application命令仅配置应用程序的名称和描述。要上传应用程序的源代码，请使用创建应用程序的初始版本create-application-version。create-application-version还有一个auto-create-application选项，可让您一步创建应用程序和应用程序版本。

输出：

```
{
  "Application": {
    "ApplicationName": "MyApp",
```



```
    "ConfigurationTemplates": [],
    "DateUpdated": "2015-02-12T18:32:21.181Z",
    "Description": "my application",
    "DateCreated": "2015-02-12T18:32:21.181Z"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateApplication](#)中的。

## create-configuration-template

以下代码示例演示了如何使用 create-configuration-template。

### AWS CLI

#### 创建配置模板

以下命令根据应用于环境的设置创建名my-app-v1为的配置模板，ID 为e-rpqsewtp2j：

```
aws elasticbeanstalk create-configuration-template --application-name my-app --
template-name my-app-v1 --environment-id e-rpqsewtp2j
```

输出：

```
{
  "ApplicationName": "my-app",
  "TemplateName": "my-app-v1",
  "DateCreated": "2015-08-12T18:40:39Z",
  "DateUpdated": "2015-08-12T18:40:39Z",
  "SolutionStackName": "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 8 Java 8"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateConfigurationTemplate](#)中的。

## create-environment

以下代码示例演示了如何使用 create-environment。

### AWS CLI

#### 为应用程序创建新环境

以下命令为名为“my-app”的 Java 应用程序的“v1”版本创建新环境：

```
aws elasticbeanstalk create-environment --application-name my-app --environment-name
my-env --cname-prefix my-app --version-label v1 --solution-stack-name "64bit Amazon
Linux 2015.03 v2.0.0 running Tomcat 8 Java 8"
```

输出：

```
{
  "ApplicationName": "my-app",
  "EnvironmentName": "my-env",
  "VersionLabel": "v1",
  "Status": "Launching",
  "EnvironmentId": "e-izqpassy4h",
  "SolutionStackName": "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 8 Java 8",
  "CNAME": "my-app.elasticbeanstalk.com",
  "Health": "Grey",
  "Tier": {
    "Type": "Standard",
    "Name": "WebServer",
    "Version": " "
  },
  "DateUpdated": "2015-02-03T23:04:54.479Z",
  "DateCreated": "2015-02-03T23:04:54.479Z"
}
```

v1是之前使用上传的应用程序版本的标签 create-application-version。

指定 JSON 文件来定义环境配置选项

以下create-environment命令指定myoptions.json应使用名为的 JSON 文件来覆盖从解决方案堆栈或配置模板中获取的值：

```
aws elasticbeanstalk create-environment --environment-name sample-env --application-
name sampleapp --option-settings file://myoptions.json
```

myoptions.json是一个定义多个设置的 JSON 对象：

```
[
  {
    "Namespace": "aws:elb:healthcheck",
    "OptionName": "Interval",
```

```
    "Value": "15"
  },
  {
    "Namespace": "aws:elb:healthcheck",
    "OptionName": "Timeout",
    "Value": "8"
  },
  {
    "Namespace": "aws:elb:healthcheck",
    "OptionName": "HealthyThreshold",
    "Value": "2"
  },
  {
    "Namespace": "aws:elb:healthcheck",
    "OptionName": "UnhealthyThreshold",
    "Value": "3"
  }
]
```

有关更多信息，请参阅《Elastic Beanstalk 开发者指南》中的期权值。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateEnvironment](#) 中的。

## create-storage-location

以下代码示例演示了如何使用 `create-storage-location`。

### AWS CLI

#### 创建存储位置

以下命令在 Amazon S3 中创建存储位置：

```
aws elasticbeanstalk create-storage-location
```

输出：

```
{
  "S3Bucket": "elasticbeanstalk-us-west-2-0123456789012"
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateStorageLocation](#) 中的。

## delete-application-version

以下代码示例演示了如何使用 delete-application-version。

### AWS CLI

删除应用程序版本

以下命令删除以名22a0-stage-150819\_182129为的应用程序命名的应用程序的应用程序版本my-app：

```
aws elasticbeanstalk delete-application-version --version-label 22a0-stage-150819_182129 --application-name my-app
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteApplicationVersion](#)中的。

## delete-application

以下代码示例演示了如何使用 delete-application。

### AWS CLI

删除应用程序

以下命令删除名为的应用程序my-app：

```
aws elasticbeanstalk delete-application --application-name my-app
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteApplication](#)中的。

## delete-configuration-template

以下代码示例演示了如何使用 delete-configuration-template。

### AWS CLI

删除配置模板

以下命令删除my-template为名为的应用程序命名的配置模板my-app：

```
aws elasticbeanstalk delete-configuration-template --template-name my-template --application-name my-app
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteConfigurationTemplate](#)中的。

## delete-environment-configuration

以下代码示例演示了如何使用 delete-environment-configuration。

### AWS CLI

#### 删除草稿配置

以下命令删除名为的环境的草稿配置my-env：

```
aws elasticbeanstalk delete-environment-configuration --environment-name my-env --application-name my-app
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteEnvironmentConfiguration](#)中的。

## describe-application-versions

以下代码示例演示了如何使用 describe-application-versions。

### AWS CLI

#### 查看有关应用程序版本的信息

以下命令检索标有以下标签v2的应用程序版本的相关信息：

```
aws elasticbeanstalk describe-application-versions --application-name my-app --version-label "v2"
```

输出：

```
{
  "ApplicationVersions": [
    {
      "ApplicationName": "my-app",
      "VersionLabel": "v2",
      "Description": "update cover page",
      "DateCreated": "2015-07-23T01:32:26.079Z",
      "DateUpdated": "2015-07-23T01:32:26.079Z",
      "SourceBundle": {
        "S3Bucket": "elasticbeanstalk-us-west-2-015321684451",
```

```

        "S3Key": "my-app/5026-stage-150723_224258.war"
      }
    },
    {
      "ApplicationName": "my-app",
      "VersionLabel": "v1",
      "Description": "initial version",
      "DateCreated": "2015-07-23T22:26:10.816Z",
      "DateUpdated": "2015-07-23T22:26:10.816Z",
      "SourceBundle": {
        "S3Bucket": "elasticbeanstalk-us-west-2-015321684451",
        "S3Key": "my-app/5026-stage-150723_222618.war"
      }
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeApplicationVersions](#)中的。

## describe-applications

以下代码示例演示了如何使用 describe-applications。

### AWS CLI

查看应用程序列表

以下命令检索有关当前区域中应用程序的信息：

```
aws elasticbeanstalk describe-applications
```

输出：

```

{
  "Applications": [
    {
      "ApplicationName": "ruby",
      "ConfigurationTemplates": [],
      "DateUpdated": "2015-08-13T21:05:44.376Z",
      "Versions": [
        "Sample Application"
      ],
      "DateCreated": "2015-08-13T21:05:44.376Z"
    }
  ]
}

```

```

    },
    {
      "ApplicationName": "pythonsample",
      "Description": "Application created from the EB CLI using \"eb init\"",
      "Versions": [
        "Sample Application"
      ],
      "DateCreated": "2015-08-13T19:05:43.637Z",
      "ConfigurationTemplates": [],
      "DateUpdated": "2015-08-13T19:05:43.637Z"
    },
    {
      "ApplicationName": "nodejs-example",
      "ConfigurationTemplates": [],
      "DateUpdated": "2015-08-06T17:50:02.486Z",
      "Versions": [
        "add elasticache",
        "First Release"
      ],
      "DateCreated": "2015-08-06T17:50:02.486Z"
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeApplications](#)中的。

## describe-configuration-options

以下代码示例演示了如何使用 describe-configuration-options。

### AWS CLI

#### 查看环境的配置选项

以下命令检索名my-env为的环境的所有可用配置选项的描述：

```
aws elasticbeanstalk describe-configuration-options --environment-name my-env --
application-name my-app
```

输出 ( 缩写 ) ：

```
{
```

```
"Options": [  
  {  
    "Name": "JVMOptions",  
    "UserDefined": false,  
    "DefaultValue": "Xms=256m,Xmx=256m,XX:MaxPermSize=64m,JVM Options=",  
    "ChangeSeverity": "RestartApplicationServer",  
    "Namespace": "aws:cloudformation:template:parameter",  
    "ValueType": "KeyValueList"  
  },  
  {  
    "Name": "Interval",  
    "UserDefined": false,  
    "DefaultValue": "30",  
    "ChangeSeverity": "NoInterruption",  
    "Namespace": "aws:elb:healthcheck",  
    "MaxValue": 300,  
    "MinValue": 5,  
    "ValueType": "Scalar"  
  },  
  ...  
  {  
    "Name": "LowerThreshold",  
    "UserDefined": false,  
    "DefaultValue": "2000000",  
    "ChangeSeverity": "NoInterruption",  
    "Namespace": "aws:autoscaling:trigger",  
    "MinValue": 0,  
    "ValueType": "Scalar"  
  },  
  {  
    "Name": "ListenerEnabled",  
    "UserDefined": false,  
    "DefaultValue": "true",  
    "ChangeSeverity": "Unknown",  
    "Namespace": "aws:elb:listener",  
    "ValueType": "Boolean"  
  }  
]  
}
```

可用的配置选项因平台和配置版本而异。有关命名空间和支持的选项的更多信息，请参阅《Elastic Be AWS anstalk 开发者指南》中的选项值。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeConfigurationOptions](#)中的。



## describe-configuration-settings

以下代码示例演示了如何使用 describe-configuration-settings。

### AWS CLI

#### 查看环境的配置设置

以下命令检索名my-env为的环境的配置设置：

```
aws elasticbeanstalk describe-configuration-settings --environment-name my-env --
application-name my-app
```

输出 ( 缩写 ) ：

```
{
  "ConfigurationSettings": [
    {
      "ApplicationName": "my-app",
      "EnvironmentName": "my-env",
      "Description": "Environment created from the EB CLI using \"eb create
      \",
      "DeploymentStatus": "deployed",
      "DateCreated": "2015-08-13T19:16:25Z",
      "OptionSettings": [
        {
          "OptionName": "Availability Zones",
          "ResourceName": "AWSEBAutoScalingGroup",
          "Namespace": "aws:autoscaling:asg",
          "Value": "Any"
        },
        {
          "OptionName": "Cooldown",
          "ResourceName": "AWSEBAutoScalingGroup",
          "Namespace": "aws:autoscaling:asg",
          "Value": "360"
        },
        ...
        {
          "OptionName": "ConnectionDrainingTimeout",
          "ResourceName": "AWSEBLoadBalancer",
          "Namespace": "aws:elb:policies",
          "Value": "20"
        }
      ]
    }
  ]
}
```

```

        },
        {
            "OptionName": "ConnectionSettingIdleTimeout",
            "ResourceName": "AWSEBLoadBalancer",
            "Namespace": "aws:elb:policies",
            "Value": "60"
        }
    ],
    "DateUpdated": "2015-08-13T23:30:07Z",
    "SolutionStackName": "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 8
Java 8"
    }
]
}

```

有关命名空间和支持的选项的更多信息，请参阅《Elastic Be AWS anstalk 开发者指南》中的选项值。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeConfigurationSettings](#)中的。

## describe-environment-health

以下代码示例演示了如何使用 describe-environment-health。

### AWS CLI

#### 查看环境运行状况

以下命令检索名my-env为的环境的整体运行状况信息：

```
aws elasticbeanstalk describe-environment-health --environment-name my-env --
attribute-names All
```

输出：

```
{
  "Status": "Ready",
  "EnvironmentName": "my-env",
  "Color": "Green",
  "ApplicationMetrics": {
    "Duration": 10,
    "Latency": {
      "P99": 0.004,
```

```
        "P75": 0.002,  
        "P90": 0.003,  
        "P95": 0.004,  
        "P85": 0.003,  
        "P10": 0.001,  
        "P999": 0.004,  
        "P50": 0.001  
    },  
    "RequestCount": 45,  
    "StatusCodes": {  
        "Status3xx": 0,  
        "Status2xx": 45,  
        "Status5xx": 0,  
        "Status4xx": 0  
    }  
},  
"RefreshedAt": "2015-08-20T21:09:18Z",  
"HealthStatus": "Ok",  
"InstancesHealth": {  
    "Info": 0,  
    "Ok": 1,  
    "Unknown": 0,  
    "Severe": 0,  
    "Warning": 0,  
    "Degraded": 0,  
    "NoData": 0,  
    "Pending": 0  
},  
"Causes": []  
}
```

运行状况信息仅适用于启用了增强型运行状况报告的环境。有关更多信息，请参阅《Elasti AWS c Beanstalk 开发者指南》中的增强型运行状况报告和监控。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEnvironmentHealth](#)中的。

## describe-environment-resources

以下代码示例演示了如何使用 describe-environment-resources。

### AWS CLI

查看有关您的环境中 AWS 资源的信息

以下命令检索名my-env为的环境中有关资源的信息：

```
aws elasticbeanstalk describe-environment-resources --environment-name my-env
```

输出：

```
{
  "EnvironmentResources": {
    "EnvironmentName": "my-env",
    "AutoScalingGroups": [
      {
        "Name": "awseb-e-qu3fyyjyjs-stack-AWSEBAutoScalingGroup-
QSB2Z088SXZT"
      }
    ],
    "Triggers": [],
    "LoadBalancers": [
      {
        "Name": "awseb-e-q-AWSEBLoa-1EEPZ0K98BIF0"
      }
    ],
    "Queues": [],
    "Instances": [
      {
        "Id": "i-0c91c786"
      }
    ],
    "LaunchConfigurations": [
      {
        "Name": "awseb-e-qu3fyyjyjs-stack-
AWSEBAutoScalingLaunchConfiguration-1UUVQIBC96TQ2"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEnvironmentResources](#)中的。

## describe-environments

以下代码示例演示了如何使用 describe-environments。

## AWS CLI

查看有关环境的信息

以下命令检索名my-env为的环境的相关信息：

```
aws elasticbeanstalk describe-environments --environment-names my-env
```

输出：

```
{
  "Environments": [
    {
      "ApplicationName": "my-app",
      "EnvironmentName": "my-env",
      "VersionLabel": "7f58-stage-150812_025409",
      "Status": "Ready",
      "EnvironmentId": "e-rpqsewtp2j",
      "EndpointURL": "awseb-e-w-AWSEBLoa-1483140XB0Q4L-109QXY8121.us-
west-2.elb.amazonaws.com",
      "SolutionStackName": "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 8
Java 8",
      "CNAME": "my-env.elasticbeanstalk.com",
      "Health": "Green",
      "AbortableOperationInProgress": false,
      "Tier": {
        "Version": " ",
        "Type": "Standard",
        "Name": "WebServer"
      },
      "DateUpdated": "2015-08-12T18:16:55.019Z",
      "DateCreated": "2015-08-07T20:48:49.599Z"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEnvironments](#)中的。

## describe-events

以下代码示例演示了如何使用 describe-events。

## AWS CLI

### 查看环境的事件

以下命令检索名my-env为的环境的事件：

```
aws elasticbeanstalk describe-events --environment-name my-env
```

输出 ( 缩写 )：

```
{
  "Events": [
    {
      "ApplicationName": "my-app",
      "EnvironmentName": "my-env",
      "Message": "Environment health has transitioned from Info to Ok.",
      "EventDate": "2015-08-20T07:06:53.535Z",
      "Severity": "INFO"
    },
    {
      "ApplicationName": "my-app",
      "EnvironmentName": "my-env",
      "Severity": "INFO",
      "RequestId": "b7f3960b-4709-11e5-ba1e-07e16200da41",
      "Message": "Environment update completed successfully.",
      "EventDate": "2015-08-20T07:06:02.049Z"
    },
    ...
    {
      "ApplicationName": "my-app",
      "EnvironmentName": "my-env",
      "Severity": "INFO",
      "RequestId": "ca8dfbf6-41ef-11e5-988b-651aa638f46b",
      "Message": "Using elasticbeanstalk-us-west-2-012445113685 as Amazon S3
storage bucket for environment data.",
      "EventDate": "2015-08-13T19:16:27.561Z"
    },
    {
      "ApplicationName": "my-app",
      "EnvironmentName": "my-env",
      "Severity": "INFO",
      "RequestId": "cdfba8f6-41ef-11e5-988b-65638f41aa6b",
      "Message": "createEnvironment is starting.",

```

```
        "EventDate": "2015-08-13T19:16:26.581Z"
      }
    ]
  }
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEvents](#)中的。

## describe-instances-health

以下代码示例演示了如何使用 describe-instances-health。

### AWS CLI

#### 查看环境运行状况

以下命令检索名my-env为的环境中实例的运行状况信息：

```
aws elasticbeanstalk describe-instances-health --environment-name my-env --
attribute-names All
```

输出：

```
{
  "InstanceHealthList": [
    {
      "InstanceId": "i-08691cc7",
      "ApplicationMetrics": {
        "Duration": 10,
        "Latency": {
          "P99": 0.006,
          "P75": 0.002,
          "P90": 0.004,
          "P95": 0.005,
          "P85": 0.003,
          "P10": 0.0,
          "P999": 0.006,
          "P50": 0.001
        },
        "RequestCount": 48,
        "StatusCodes": {
          "Status3xx": 0,
          "Status2xx": 47,

```

```

        "Status5xx": 0,
        "Status4xx": 1
      }
    },
    "System": {
      "LoadAverage": [
        0.0,
        0.02,
        0.05
      ],
      "CPUUtilization": {
        "SoftIRQ": 0.1,
        "IOWait": 0.2,
        "System": 0.3,
        "Idle": 97.8,
        "User": 1.5,
        "IRQ": 0.0,
        "Nice": 0.1
      }
    },
    "Color": "Green",
    "HealthStatus": "Ok",
    "LaunchedAt": "2015-08-13T19:17:09Z",
    "Causes": []
  }
],
"RefreshedAt": "2015-08-20T21:09:08Z"
}

```

运行状况信息仅适用于启用了增强型运行状况报告的环境。有关更多信息，请参阅《Elasti AWS c Beanstalk 开发者指南》中的增强型运行状况报告和监控。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeInstancesHealth](#) 中的。

## list-available-solution-stacks

以下代码示例演示了如何使用 `list-available-solution-stacks`。

### AWS CLI

#### 查看解决方案堆栈

以下命令列出了所有当前可用平台配置以及您过去使用的任何平台配置的解决方案堆栈：



```
aws elasticbeanstalk list-available-solution-stacks
```

输出 ( 缩写 ) :

```
{
  "SolutionStacks": [
    "64bit Amazon Linux 2015.03 v2.0.0 running Node.js",
    "64bit Amazon Linux 2015.03 v2.0.0 running PHP 5.6",
    "64bit Amazon Linux 2015.03 v2.0.0 running PHP 5.5",
    "64bit Amazon Linux 2015.03 v2.0.0 running PHP 5.4",
    "64bit Amazon Linux 2015.03 v2.0.0 running Python 3.4",
    "64bit Amazon Linux 2015.03 v2.0.0 running Python 2.7",
    "64bit Amazon Linux 2015.03 v2.0.0 running Python",
    "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.2 (Puma)",
    "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.2 (Passenger Standalone)",
    "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.1 (Puma)",
    "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.1 (Passenger Standalone)",
    "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.0 (Puma)",
    "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.0 (Passenger Standalone)",
    "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 1.9.3",
    "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 8 Java 8",
    "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 7 Java 7",
    "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 7 Java 6",
    "64bit Windows Server Core 2012 R2 running IIS 8.5",
    "64bit Windows Server 2012 R2 running IIS 8.5",
    "64bit Windows Server 2012 running IIS 8",
    "64bit Windows Server 2008 R2 running IIS 7.5",
    "64bit Amazon Linux 2015.03 v2.0.0 running Docker 1.6.2",
    "64bit Amazon Linux 2015.03 v2.0.0 running Multi-container Docker 1.6.2
(Generic)",
    "64bit Debian jessie v2.0.0 running GlassFish 4.1 Java 8 (Preconfigured -
Docker)",
    "64bit Debian jessie v2.0.0 running GlassFish 4.0 Java 7 (Preconfigured -
Docker)",
    "64bit Debian jessie v2.0.0 running Go 1.4 (Preconfigured - Docker)",
    "64bit Debian jessie v2.0.0 running Go 1.3 (Preconfigured - Docker)",
    "64bit Debian jessie v2.0.0 running Python 3.4 (Preconfigured - Docker)",
  ],
  "SolutionStackDetails": [
    {
      "PermittedFileTypes": [
        "zip"
      ],
    },
  ],
}
```

```
        "SolutionStackName": "64bit Amazon Linux 2015.03 v2.0.0 running Node.js"  
    },  
    ...  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAvailableSolutionStacks](#)中的。

## rebuild-environment

以下代码示例演示了如何使用 `rebuild-environment`。

### AWS CLI

#### 重建环境

以下命令终止并在名为的环境中重新创建资源：`my-env`

```
aws elasticbeanstalk rebuild-environment --environment-name my-env
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RebuildEnvironment](#)中的。

## request-environment-info

以下代码示例演示了如何使用 `request-environment-info`。

### AWS CLI

#### 请求尾部日志

以下命令从名为的环境请求日志`my-env`：

```
aws elasticbeanstalk request-environment-info --environment-name my-env --info-type  
tail
```

请求日志后，使用检索其位置 `retrieve-environment-info`。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RequestEnvironmentInfo](#)中的。

## restart-app-server

以下代码示例演示了如何使用 `restart-app-server`。

## AWS CLI

### 重新启动应用程序服务器

以下命令在名为my-env为的环境中重新启动所有实例上的应用程序服务器：

```
aws elasticbeanstalk restart-app-server --environment-name my-env
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RestartAppServer](#)中的。

## retrieve-environment-info

以下代码示例演示了如何使用 retrieve-environment-info。

## AWS CLI

### 检索尾部日志

以下命令从名为my-env为的环境中检索指向日志的链接：

```
aws elasticbeanstalk retrieve-environment-info --environment-name my-env --info-type tail
```

输出：

```
{
  "EnvironmentInfo": [
    {
      "SampleTimestamp": "2015-08-20T22:23:17.703Z",
      "Message": "https://elasticbeanstalk-us-west-2-0123456789012.s3.amazonaws.com/resources/environments/logs/tail/e-fyqyju3yjs/i-09c1c867/TailLogs-1440109397703.out?AWSAccessKeyId=AKGPT4J56IAJ2EUBL5CQ&Expires=1440195891&Signature=n%2BEa10V6A2HI0x4Rcfb7LT16bBM%3D",
      "InfoType": "tail",
      "Ec2InstanceId": "i-09c1c867"
    }
  ]
}
```

在浏览器中查看链接。在检索之前，必须使用请求日志 request-environment-info。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RetrieveEnvironmentInfo](#)中的。

## swap-environment-cnames

以下代码示例演示了如何使用 swap-environment-cnames。

### AWS CLI

#### 交换环境 CNAMEs

以下命令交换两个环境的已分配子域：

```
aws elasticbeanstalk swap-environment-cnames --source-environment-name my-env-blue
--destination-environment-name my-env-green
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SwapEnvironmentCnames](#)中的。

## terminate-environment

以下代码示例演示了如何使用 terminate-environment。

### AWS CLI

#### 终止环境

以下命令终止名为的 Elastic Beanstalk 环境：my-env

```
aws elasticbeanstalk terminate-environment --environment-name my-env
```

输出：

```
{
  "ApplicationName": "my-app",
  "EnvironmentName": "my-env",
  "Status": "Terminating",
  "EnvironmentId": "e-fh2eravpns",
  "EndpointURL": "awseb-e-f-AWSEBLoa-1I9XUMP4-8492WNUP202574.us-
west-2.elb.amazonaws.com",
  "SolutionStackName": "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 8 Java
8",
  "CNAME": "my-env.elasticbeanstalk.com",
  "Health": "Grey",
```

```
"AbortableOperationInProgress": false,
  "Tier": {
    "Version": " ",
    "Type": "Standard",
    "Name": "WebServer"
  },
  "DateUpdated": "2015-08-12T19:05:54.744Z",
  "DateCreated": "2015-08-12T18:52:53.622Z"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TerminateEnvironment](#)中的。

## update-application-version

以下代码示例演示了如何使用 `update-application-version`。

### AWS CLI

#### 更改应用程序版本的描述

以下命令更新名为的应用程序版本的描述22a0-stage-150819\_185942：

```
aws elasticbeanstalk update-application-version --version-label 22a0-
stage-150819_185942 --application-name my-app --description "new description"
```

输出：

```
{
  "ApplicationVersion": {
    "ApplicationName": "my-app",
    "VersionLabel": "22a0-stage-150819_185942",
    "Description": "new description",
    "DateCreated": "2015-08-19T18:59:17.646Z",
    "DateUpdated": "2015-08-20T22:53:28.871Z",
    "SourceBundle": {
      "S3Bucket": "elasticbeanstalk-us-west-2-0123456789012",
      "S3Key": "my-app/22a0-stage-150819_185942.war"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateApplicationVersion](#)中的。

## update-application

以下代码示例演示了如何使用 `update-application`。

### AWS CLI

#### 更改应用程序的描述

以下命令更新名为 `my-app` 的应用程序的描述：

```
aws elasticbeanstalk update-application --application-name my-app --description "my Elastic Beanstalk application"
```

输出：

```
{
  "Application": {
    "ApplicationName": "my-app",
    "Description": "my Elastic Beanstalk application",
    "Versions": [
      "2fba-stage-150819_234450",
      "bf07-stage-150820_214945",
      "93f8",
      "fd7c-stage-150820_000431",
      "22a0-stage-150819_185942"
    ],
    "DateCreated": "2015-08-13T19:15:50.449Z",
    "ConfigurationTemplates": [],
    "DateUpdated": "2015-08-20T22:34:56.195Z"
  }
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateApplication](#) 中的。

## update-configuration-template

以下代码示例演示了如何使用 `update-configuration-template`。

### AWS CLI

#### 更新配置模板

以下命令从名为 `my-template` 的已保存配置的配置模板中删除已配置的 CloudWatch 自定义运行状况指标配置 `my-template`：

```
aws elasticbeanstalk update-configuration-template --template-name my-template --application-name my-app --options-to-remove Namespace=aws:elasticbeanstalk:healthreporting:system,OptionName=ConfigDocument
```

输出：

```
{
  "ApplicationName": "my-app",
  "TemplateName": "my-template",
  "DateCreated": "2015-08-20T22:39:31Z",
  "DateUpdated": "2015-08-20T22:43:11Z",
  "SolutionStackName": "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 8 Java 8"
}
```

有关命名空间和支持的选项的更多信息，请参阅《Elastic Be AWS anstalk 开发者指南》中的选项值。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateConfigurationTemplate](#) 中的。

## update-environment

以下代码示例演示了如何使用 `update-environment`。

### AWS CLI

将环境更新到新版本

以下命令将名为 `my-env` 的环境更新为其所属应用程序的 `v2` 版本：

```
aws elasticbeanstalk update-environment --environment-name my-env --version-label v2
```

此命令要求 `my-env` 环境已经存在，并且属于标签为 `v2` 的有效应用程序版本的应用程序。

输出：

```
{
  "ApplicationName": "my-app",
  "EnvironmentName": "my-env",
}
```

```
"VersionLabel": "v2",
"Status": "Updating",
"EnvironmentId": "e-szqipays4h",
"EndpointURL": "awseb-e-i-AWSEBLoa-1RD LX6TC9VUA0-0123456789.us-
west-2.elb.amazonaws.com",
"SolutionStackName": "64bit Amazon Linux running Tomcat 7",
"CNAME": "my-env.elasticbeanstalk.com",
"Health": "Grey",
"Tier": {
  "Version": " ",
  "Type": "Standard",
  "Name": "WebServer"
},
"DateUpdated": "2015-02-03T23:12:29.119Z",
"DateCreated": "2015-02-03T23:04:54.453Z"
}
```

## 设置环境变量

以下命令将“my-env”环境中“PARAM1”变量的值设置为“”：ParamValue

```
aws elasticbeanstalk update-environment --environment-name my-env --option-settings
Namespace=aws:elasticbeanstalk:application:environment,OptionName=PARAM1,Value=ParamValue
```

除了变量的名称和值外，该option-settings参数还需要一个命名空间。除了环境变量外，Elastic Beanstalk 还支持多个用于选项的命名空间。

## 从文件配置选项设置

以下命令通过文件在aws:elb:loadbalancer命名空间中配置多个选项：

```
aws elasticbeanstalk update-environment --environment-name my-env --option-settings
file://options.json
```

options.json是一个定义多个设置的 JSON 对象：

```
[
  {
    "Namespace": "aws:elb:healthcheck",
    "OptionName": "Interval",
    "Value": "15"
  },
]
```



```
{
  "Namespace": "aws:elb:healthcheck",
  "OptionName": "Timeout",
  "Value": "8"
},
{
  "Namespace": "aws:elb:healthcheck",
  "OptionName": "HealthyThreshold",
  "Value": "2"
},
{
  "Namespace": "aws:elb:healthcheck",
  "OptionName": "UnhealthyThreshold",
  "Value": "3"
}
]
```

输出：

```
{
  "ApplicationName": "my-app",
  "EnvironmentName": "my-env",
  "VersionLabel": "7f58-stage-150812_025409",
  "Status": "Updating",
  "EnvironmentId": "e-wtp2rqpqsej",
  "EndpointURL": "awseb-e-w-AWSEBLoa-14XB83101Q4L-104QXY80921.sa-
east-1.elb.amazonaws.com",
  "SolutionStackName": "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 8 Java
8",
  "CNAME": "my-env.elasticbeanstalk.com",
  "Health": "Grey",
  "AbortableOperationInProgress": true,
  "Tier": {
    "Version": " ",
    "Type": "Standard",
    "Name": "WebServer"
  },
  "DateUpdated": "2015-08-12T18:15:23.804Z",
  "DateCreated": "2015-08-07T20:48:49.599Z"
}
```

有关命名空间和支持的选项的更多信息，请参阅《Elastic Be AWS anstalk 开发者指南》中的选项值。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateEnvironment](#)中的。

## validate-configuration-settings

以下代码示例演示了如何使用 validate-configuration-settings。

### AWS CLI

#### 验证配置设置

以下命令验证 CloudWatch 自定义指标配置文档：

```
aws elasticbeanstalk validate-configuration-settings --application-name my-app --
environment-name my-env --option-settings file://options.json
```

options.json是一个 JSON 文档，其中包含一个或多个要验证的配置设置：

```
[
  {
    "Namespace": "aws:elasticbeanstalk:healthreporting:system",
    "OptionName": "ConfigDocument",
    "Value": "{\"CloudWatchMetrics\": {\"Environment\":
  {\"ApplicationLatencyP99.9\": null,\"InstancesSevere\": 60,
  \\\"ApplicationLatencyP90\\\": 60,\"ApplicationLatencyP99\": null,
  \\\"ApplicationLatencyP95\\\": 60,\"InstancesUnknown\": 60,\"ApplicationLatencyP85\\\":
  60,\"InstancesInfo\": null,\"ApplicationRequests2xx\": null,\"InstancesDegraded
  \": null,\"InstancesWarning\": 60,\"ApplicationLatencyP50\": 60,
  \\\"ApplicationRequestsTotal\": null,\"InstancesNoData\": null,\"InstancesPending
  \": 60,\"ApplicationLatencyP10\": null,\"ApplicationRequests5xx\": null,
  \\\"ApplicationLatencyP75\\\": null,\"InstancesOk\": 60,\"ApplicationRequests3xx\":
  null,\"ApplicationRequests4xx\": null},\"Instance\": {\"ApplicationLatencyP99.9\":
  null,\"ApplicationLatencyP90\": 60,\"ApplicationLatencyP99\": null,
  \\\"ApplicationLatencyP95\\\": null,\"ApplicationLatencyP85\": null,\"CPUUser\": 60,
  \\\"ApplicationRequests2xx\": null,\"CPUIdle\": null,\"ApplicationLatencyP50\":
  null,\"ApplicationRequestsTotal\": 60,\"RootFilesystemUtil\": null,
  \\\"LoadAverage1min\": null,\"CPUIrq\": null,\"CPUNice\": 60,\"CPUiowait\": 60,
  \\\"ApplicationLatencyP10\": null,\"LoadAverage5min\": null,\"ApplicationRequests5xx
  \": null,\"ApplicationLatencyP75\": 60,\"CPUSystem\": 60,\"ApplicationRequests3xx\":
  60,\"ApplicationRequests4xx\": null,\"InstanceHealth\": null,\"CPUsoftirq\": 60}},
  \\\"Version\": 1}"
  }
]
```

如果您指定的选项对指定环境有效，则 Elastic Beanstalk 会返回一个空的消息数组：

```
{
  "Messages": []
}
```

如果验证失败，则响应将包含有关错误的信息：

```
{
  "Messages": [
    {
      "OptionName": "ConfigDocumet",
      "Message": "Invalid option specification (Namespace:
'aws:elasticbeanstalk:healthreporting:system', OptionName: 'ConfigDocumet'):
Unknown configuration setting.",
      "Namespace": "aws:elasticbeanstalk:healthreporting:system",
      "Severity": "error"
    }
  ]
}
```

有关命名空间和支持的选项的更多信息，请参阅《Elastic Be AWS anstalk 开发者指南》中的选项值。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ValidateConfigurationSettings](#) 中的。

## Elastic Load Balancing-版本 1 示例使用 AWS CLI

以下代码示例向您展示了如何在 Elastic Load Balancing-版本 1 中 AWS Command Line Interface 用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### add-tags

以下代码示例演示了如何使用 `add-tags`。

#### AWS CLI

向负载均衡器添加标签

此示例向指定的负载均衡器添加标签。

命令:

```
aws elb add-tags --load-balancer-name my-load-balancer --tags
"Key=project,Value=lima" "Key=department,Value=digital-media"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddTags](#)中的。

### apply-security-groups-to-load-balancer

以下代码示例演示了如何使用 `apply-security-groups-to-load-balancer`。

#### AWS CLI

将安全组与 VPC 中的负载均衡器关联

此示例将安全组与 VPC 中的指定负载均衡器相关联。

命令:

```
aws elb apply-security-groups-to-load-balancer --load-balancer-name my-load-balancer
--security-groups sg-fc448899
```

输出:

```
{
  "SecurityGroups": [
    "sg-fc448899"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ApplySecurityGroupsToLoadBalancer](#)中的。

## attach-load-balancer-to-subnets

以下代码示例演示了如何使用 `attach-load-balancer-to-subnets`。

### AWS CLI

将子网连接到负载均衡器

此示例将指定的子网添加到指定负载均衡器的已配置子网集中。

命令:

```
aws elb attach-load-balancer-to-subnets --load-balancer-name my-load-balancer --
subnets subnet-0ecac448
```

输出:

```
{
  "Subnets": [
    "subnet-15aaab61",
    "subnet-0ecac448"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachLoadBalancerToSubnets](#)中的。

## configure-health-check

以下代码示例演示了如何使用 `configure-health-check`。

### AWS CLI

为您的后端 EC2 实例指定运行状况检查设置

此示例指定了用于评估后端 EC2 实例运行状况的运行状况检查设置。

命令:

```
aws elb configure-health-check --load-balancer-name my-load-balancer --health-check
Target=HTTP:80/png,Interval=30,UnhealthyThreshold=2,HealthyThreshold=2,Timeout=3
```

输出：

```
{
  "HealthCheck": {
    "HealthyThreshold": 2,
    "Interval": 30,
    "Target": "HTTP:80/png",
    "Timeout": 3,
    "UnhealthyThreshold": 2
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ConfigureHealthCheck](#)中的。

### **create-app-cookie-stickness-policy**

以下代码示例演示了如何使用 `create-app-cookie-stickness-policy`。

#### AWS CLI

为您的 HTTPS 负载均衡器生成粘性策略

此示例生成粘性策略，该策略遵循应用程序生成的 Cookie 的粘性会话生命周期。

命令：

```
aws elb create-app-cookie-stickness-policy --load-balancer-name my-load-balancer --
policy-name my-app-cookie-policy --cookie-name my-app-cookie
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAppCookieStickinessPolicy](#)中的。

### **create-lb-cookie-stickness-policy**

以下代码示例演示了如何使用 `create-lb-cookie-stickness-policy`。

#### AWS CLI

为您的 HTTPS 负载均衡器生成基于持续时间的粘性策略

此示例生成粘性策略，其粘性会话生命周期由指定的到期时间控制。

命令：

```
aws elb create-lb-cookie-stickness-policy --load-balancer-name my-load-balancer --  
policy-name my-duration-cookie-policy --cookie-expiration-period 60
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLbCookieStickinessPolicy](#)中的。

## create-load-balancer-listeners

以下代码示例演示了如何使用 create-load-balancer-listeners。

### AWS CLI

为负载均衡器创建 HTTP 侦听器

此示例使用 HTTP 协议在端口 80 上为您的负载均衡器创建侦听器。

命令:

```
aws elb create-load-balancer-listeners --load-balancer-name my-load-balancer --  
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"
```

为负载均衡器创建 HTTPS 侦听器

此示例使用 HTTPS 协议在端口 443 上为您的负载均衡器创建侦听器。

命令:

```
aws elb create-load-balancer-listeners --load-balancer-name my-load-balancer --  
listeners  
"Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTP,InstancePort=80"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLoadBalancerListeners](#)中的。

## create-load-balancer-policy

以下代码示例演示了如何使用 create-load-balancer-policy。

### AWS CLI

创建在负载均衡器上启用代理协议的策略

此示例创建了一个在指定负载均衡器上启用代理协议的策略。

**命令:**

```
aws elb create-load-balancer-policy --load-balancer-name my-load-balancer --policy-name my-ProxyProtocol-policy --policy-type-name ProxyProtocolPolicyType --policy-attributes AttributeName=ProxyProtocol,AttributeValue=true
```

**使用推荐的安全策略创建 SSL 协商策略**

此示例使用推荐的安全策略为指定的 HTTPS 负载均衡器创建 SSL 协商策略。

**命令:**

```
aws elb create-load-balancer-policy --load-balancer-name my-load-balancer --policy-name my-SSLNegotiation-policy --policy-type-name SSLNegotiationPolicyType --policy-attributes AttributeName=Reference-Security-Policy,AttributeValue=ELBSecurityPolicy-2015-03
```

**使用自定义安全策略创建 SSL 协商策略**

此示例通过启用协议和密码使用自定义安全策略为您的 HTTPS 负载均衡器创建 SSL 协商策略。

**命令:**

```
aws elb create-load-balancer-policy --load-balancer-name my-load-balancer --policy-name my-SSLNegotiation-policy --policy-type-name SSLNegotiationPolicyType --policy-attributes AttributeName=Protocol-SSLv3,AttributeValue=true AttributeName=Protocol-TLSv1.1,AttributeValue=true AttributeName=DHE-RSA-AES256-SHA256,AttributeValue=true AttributeName=Server-Defined-Cipher-Order,AttributeValue=true
```

**创建公钥策略**

此示例创建了一个公钥策略。

**命令:**

```
aws elb create-load-balancer-policy --load-balancer-name my-load-balancer --policy-name my-PublicKey-policy --policy-type-name PublicKeyPolicyType --policy-attributes AttributeName=PublicKey,AttributeValue=MIIBIjANBgkqhkiG9w0BAQEFAA0CAQ8AMIIBCgKCAQEAWYUjnfy+dS74kj//c6x7R0tusUaeQCTgIUkayttRDWchuqo1pHC1u+n5xxXnBBE2ejbb2WRsKIQ5rXEeixsjFpFsojpsQKkzhVGI6mJVZBJDVKSHmswnwLBdofLhzv1lpovBPTHe+o4haAWvDBALJU0pkSI1FecPHcs2hwxf14zHoXy1e2k36A64nXW43wtfx5qcVSIxtCE0jnYRg7RPvybaGfQ
```



```
+v6Iaxb/+7J5kEvZhTFQId+bSiJImF1FSUT1W1xwzBZPUBcUkkXDj45vC2s3Z8E  
+Lk7a3uZhvsQHLZnrFuWjBWGWvZ/MhZYgEXAMPLE
```

### 创建后端服务器身份验证策略

此示例创建了一个后端服务器身份验证策略，该策略允许使用公钥策略对您的后端实例进行身份验证。

命令:

```
aws elb create-load-balancer-policy --load-balancer-name my-load-  
balancer --policy-name my-authentication-policy --policy-type-  
name BackendServerAuthenticationPolicyType --policy-attributes  
AttributeName=PublicKeyPolicyName,AttributeValue=my-PublicKey-policy
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLoadBalancerPolicy](#)中的。

## create-load-balancer

以下代码示例演示了如何使用 create-load-balancer。

### AWS CLI

#### 创建 HTTP 负载均衡器

此示例在 VPC 中创建带有 HTTP 侦听器的负载均衡器。

命令:

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --listeners  
"Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80" --subnets  
subnet-15aaab61 --security-groups sg-a61988c3
```

输出:

```
{  
  "DNSName": "my-load-balancer-1234567890.us-west-2.elb.amazonaws.com"  
}
```

此示例在 EC2-Classical 中创建了一个带有 HTTP 侦听器的负载均衡器。

命令:

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --listeners
"Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80" --
availability-zones us-west-2a us-west-2b
```

输出：

```
{
  "DNSName": "my-load-balancer-123456789.us-west-2.elb.amazonaws.com"
}
```

### 创建 HTTPS 负载均衡器

此示例在 VPC 中创建带有 HTTPS 侦听器的负载均衡器。

命令：

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --listeners
"Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"
"Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTP,InstancePort=80,SSLCertificateId
certificate/my-server-cert" --subnets subnet-15aaab61 --security-groups sg-a61988c3
```

输出：

```
{
  "DNSName": "my-load-balancer-1234567890.us-west-2.elb.amazonaws.com"
}
```

此示例在 EC2-Classical 中创建了一个带有 HTTPS 侦听器的负载均衡器。

命令：

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --listeners
"Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"
"Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTP,InstancePort=80,SSLCertificateId
certificate/my-server-cert" --availability-zones us-west-2a us-west-2b
```

输出：

```
{
  "DNSName": "my-load-balancer-123456789.us-west-2.elb.amazonaws.com"
}
```

## 创建内部负载均衡器

此示例在 VPC 中创建带有 HTTP 侦听器的内部负载均衡器。

命令:

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --listeners
"Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80" --scheme
internal --subnets subnet-a85db0df --security-groups sg-a61988c3
```

输出:

```
{
  "DNSName": "internal-my-load-balancer-123456789.us-west-2.elb.amazonaws.com"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLoadBalancer](#)中的。

## delete-load-balancer-listeners

以下代码示例演示了如何使用 delete-load-balancer-listeners。

AWS CLI

从负载均衡器中删除侦听器

此示例从指定的负载均衡器中删除指定端口的监听器。

命令:

```
aws elb delete-load-balancer-listeners --load-balancer-name my-load-balancer --load-
balancer-ports 80
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLoadBalancerListeners](#)中的。

## delete-load-balancer-policy

以下代码示例演示了如何使用 delete-load-balancer-policy。

AWS CLI

从您的负载均衡器中删除策略

此示例从指定的负载均衡器中删除指定的策略。不得在任何监听器上启用该策略。

命令:

```
aws elb delete-load-balancer-policy --load-balancer-name my-load-balancer --policy-name my-duration-cookie-policy
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLoadBalancerPolicy](#)中的。

## **delete-load-balancer**

以下代码示例演示了如何使用 `delete-load-balancer`。

AWS CLI

删除负载均衡器

此示例删除了指定的负载均衡器。

命令:

```
aws elb delete-load-balancer --load-balancer-name my-load-balancer
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLoadBalancer](#)中的。

## **deregister-instances-from-load-balancer**

以下代码示例演示了如何使用 `deregister-instances-from-load-balancer`。

AWS CLI

从负载均衡器注销实例

此示例从指定的负载均衡器中取消注册指定实例。

命令:

```
aws elb deregister-instances-from-load-balancer --load-balancer-name my-load-balancer --instances i-d6f6fae3
```

输出:

```
{
  "Instances": [
    {
      "InstanceId": "i-207d9717"
    },
    {
      "InstanceId": "i-afefb49b"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterInstancesFromLoadBalancer](#)中的。

## describe-account-limits

以下代码示例演示了如何使用 describe-account-limits。

### AWS CLI

描述您的 Classic Load Balancer 限制

以下describe-account-limits示例显示了有关您的 AWS 账户的 Classic Load Balancer 限制的详细信息。

```
aws elb describe-account-limits
```

输出：

```
{
  "Limits": [
    {
      "Name": "classic-load-balancers",
      "Max": "20"
    },
    {
      "Name": "classic-listeners",
      "Max": "100"
    },
    {
      "Name": "classic-registered-instances",
      "Max": "1000"
    }
  ]
}
```

```
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAccountLimits](#)中的。

## describe-instance-health

以下代码示例演示了如何使用 `describe-instance-health`。

### AWS CLI

描述负载均衡器实例的运行状况

此示例描述了指定负载均衡器实例的运行状况。

命令:

```
aws elb describe-instance-health --load-balancer-name my-load-balancer
```

输出:

```
{  
  "InstanceStates": [  
    {  
      "InstanceId": "i-207d9717",  
      "ReasonCode": "N/A",  
      "State": "InService",  
      "Description": "N/A"  
    },  
    {  
      "InstanceId": "i-afefb49b",  
      "ReasonCode": "N/A",  
      "State": "InService",  
      "Description": "N/A"  
    }  
  ]  
}
```

描述负载均衡器实例的运行状况

此示例描述了指定负载均衡器的指定实例的运行状况。

命令:

```
aws elb describe-instance-health --load-balancer-name my-load-balancer --instances
i-7299c809
```

以下是正在注册的实例的响应示例。

输出 :

```
{
  "InstanceStates": [
    {
      "InstanceId": "i-7299c809",
      "ReasonCode": "ELB",
      "State": "OutOfService",
      "Description": "Instance registration is still in progress."
    }
  ]
}
```

以下是运行状况不佳的实例的响应示例。

输出 :

```
{
  "InstanceStates": [
    {
      "InstanceId": "i-7299c809",
      "ReasonCode": "Instance",
      "State": "OutOfService",
      "Description": "Instance has failed at least the UnhealthyThreshold number
of health checks consecutively."
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInstanceHealth](#)中的。

## describe-load-balancer-attributes

以下代码示例演示了如何使用 describe-load-balancer-attributes。

## AWS CLI

### 描述负载均衡器的属性

此示例描述了指定负载均衡器的属性。

命令:

```
aws elb describe-load-balancer-attributes --load-balancer-name my-load-balancer
```

输出:

```
{
  "LoadBalancerAttributes": {
    "ConnectionDraining": {
      "Enabled": false,
      "Timeout": 300
    },
    "CrossZoneLoadBalancing": {
      "Enabled": true
    },
    "ConnectionSettings": {
      "IdleTimeout": 30
    },
    "AccessLog": {
      "Enabled": false
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoadBalancerAttributes](#)中的。

## describe-load-balancer-policies

以下代码示例演示了如何使用 describe-load-balancer-policies。

## AWS CLI

### 描述与负载均衡器关联的所有策略

此示例描述了与指定负载均衡器关联的所有策略。



## 命令:

```
aws elb describe-load-balancer-policies --load-balancer-name my-load-balancer
```

## 输出:

```
{
  "PolicyDescriptions": [
    {
      "PolicyAttributeDescriptions": [
        {
          "AttributeName": "ProxyProtocol",
          "AttributeValue": "true"
        }
      ],
      "PolicyName": "my-ProxyProtocol-policy",
      "PolicyTypeName": "ProxyProtocolPolicyType"
    },
    {
      "PolicyAttributeDescriptions": [
        {
          "AttributeName": "CookieName",
          "AttributeValue": "my-app-cookie"
        }
      ],
      "PolicyName": "my-app-cookie-policy",
      "PolicyTypeName": "AppCookieStickinessPolicyType"
    },
    {
      "PolicyAttributeDescriptions": [
        {
          "AttributeName": "CookieExpirationPeriod",
          "AttributeValue": "60"
        }
      ],
      "PolicyName": "my-duration-cookie-policy",
      "PolicyTypeName": "LBCookieStickinessPolicyType"
    },
    .
    .
    .
  ]
}
```

## 描述与负载均衡器关联的特定策略

此示例描述了与指定负载均衡器关联的指定策略。

命令:

```
aws elb describe-load-balancer-policies --load-balancer-name my-load-balancer --policy-name my-authentication-policy
```

输出:

```
{
  "PolicyDescriptions": [
    {
      "PolicyAttributeDescriptions": [
        {
          "AttributeName": "PublicKeyPolicyName",
          "AttributeValue": "my-PublicKey-policy"
        }
      ],
      "PolicyName": "my-authentication-policy",
      "PolicyTypeName": "BackendServerAuthenticationPolicyType"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoadBalancerPolicies](#)中的。

## describe-load-balancer-policy-types

以下代码示例演示了如何使用 describe-load-balancer-policy-types。

AWS CLI

描述 Elastic Load Balancing 定义的负载均衡器策略类型

此示例描述了可用于为负载均衡器创建策略配置的负载均衡器策略类型。

命令:

```
aws elb describe-load-balancer-policy-types
```

输出：

```
{
  "PolicyTypeDescriptions": [
    {
      "PolicyAttributeTypeDescriptions": [
        {
          "Cardinality": "ONE",
          "AttributeName": "ProxyProtocol",
          "AttributeType": "Boolean"
        }
      ],
      "PolicyTypeName": "ProxyProtocolPolicyType",
      "Description": "Policy that controls whether to include the IP address and
port of the originating request for TCP messages. This policy operates on TCP/SSL
listeners only"
    },
    {
      "PolicyAttributeTypeDescriptions": [
        {
          "Cardinality": "ONE",
          "AttributeName": "PublicKey",
          "AttributeType": "String"
        }
      ],
      "PolicyTypeName": "PublicKeyPolicyType",
      "Description": "Policy containing a list of public keys to
accept when authenticating the back-end server(s). This policy cannot be
applied directly to back-end servers or listeners but must be part of a
BackendServerAuthenticationPolicyType."
    },
    {
      "PolicyAttributeTypeDescriptions": [
        {
          "Cardinality": "ONE",
          "AttributeName": "CookieName",
          "AttributeType": "String"
        }
      ],
      "PolicyTypeName": "AppCookieStickinessPolicyType",
      "Description": "Stickiness policy with session lifetimes controlled by the
lifetime of the application-generated cookie. This policy can be associated only
with HTTP/HTTPS listeners."
    },
  ],
}
```

```

{
  "PolicyAttributeTypeDescriptions": [
    {
      "Cardinality": "ZERO_OR_ONE",
      "AttributeName": "CookieExpirationPeriod",
      "AttributeType": "Long"
    }
  ],
  "PolicyTypeName": "LBCookieStickinessPolicyType",
  "Description": "Stickiness policy with session lifetimes controlled by
the browser (user-agent) or a specified expiration period. This policy can be
associated only with HTTP/HTTPS listeners."
},
{
  "PolicyAttributeTypeDescriptions": [
    .
    .
    .
  ],
  "PolicyTypeName": "SSLNegotiationPolicyType",
  "Description": "Listener policy that defines the ciphers and protocols
that will be accepted by the load balancer. This policy can be associated only with
HTTPS/SSL listeners."
},
{
  "PolicyAttributeTypeDescriptions": [
    {
      "Cardinality": "ONE_OR_MORE",
      "AttributeName": "PublicKeyPolicyName",
      "AttributeType": "PolicyName"
    }
  ],
  "PolicyTypeName": "BackendServerAuthenticationPolicyType",
  "Description": "Policy that controls authentication to back-end server(s)
and contains one or more policies, such as an instance of a PublicKeyPolicyType.
This policy can be associated only with back-end servers that are using HTTPS/SSL."
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoadBalancerPolicyTypes](#)中的。

## describe-load-balancers

以下代码示例演示了如何使用 describe-load-balancers。

### AWS CLI

描述您的负载均衡器

此示例描述您的所有负载均衡器。

命令:

```
aws elb describe-load-balancers
```

描述您的一个负载均衡器

此示例描述指定的负载均衡器。

命令:

```
aws elb describe-load-balancers --load-balancer-name my-load-balancer
```

以下示例响应适用于 VPC 中的 HTTPS 负载均衡器。

输出:

```
{
  "LoadBalancerDescriptions": [
    {
      "Subnets": [
        "subnet-15aaab61"
      ],
      "CanonicalHostedZoneNameID": "Z3DZXE0EXAMPLE",
      "CanonicalHostedZoneName": "my-load-balancer-1234567890.us-west-2.elb.amazonaws.com",
      "ListenerDescriptions": [
        {
          "Listener": {
            "InstancePort": 80,
            "LoadBalancerPort": 80,
            "Protocol": "HTTP",
            "InstanceProtocol": "HTTP"
          },
          "PolicyNames": []
        }
      ]
    }
  ]
}
```

```
    },
    {
      "Listener": {
        "InstancePort": 443,
        "SSLCertificateId": "arn:aws:iam::123456789012:server-certificate/
my-server-cert",
        "LoadBalancerPort": 443,
        "Protocol": "HTTPS",
        "InstanceProtocol": "HTTPS"
      },
      "PolicyNames": [
        "ELBSecurityPolicy-2015-03"
      ]
    }
  ],
  "HealthCheck": {
    "HealthyThreshold": 2,
    "Interval": 30,
    "Target": "HTTP:80/png",
    "Timeout": 3,
    "UnhealthyThreshold": 2
  },
  "VPCId": "vpc-a01106c2",
  "BackendServerDescriptions": [
    {
      "InstancePort": 80,
      "PolicyNames": [
        "my-ProxyProtocol-policy"
      ]
    }
  ],
  "Instances": [
    {
      "InstanceId": "i-207d9717"
    },
    {
      "InstanceId": "i-afefb49b"
    }
  ],
  "DNSName": "my-load-balancer-1234567890.us-west-2.elb.amazonaws.com",
  "SecurityGroups": [
    "sg-a61988c3"
  ],
  "Policies": {
```

```
    "LBCookieStickinessPolicies": [
      {
        "PolicyName": "my-duration-cookie-policy",
        "CookieExpirationPeriod": 60
      }
    ],
    "AppCookieStickinessPolicies": [],
    "OtherPolicies": [
      "my-PublicKey-policy",
      "my-authentication-policy",
      "my-SSLNegotiation-policy",
      "my-ProxyProtocol-policy",
      "ELBSecurityPolicy-2015-03"
    ]
  },
  "LoadBalancerName": "my-load-balancer",
  "CreatedTime": "2015-03-19T03:24:02.650Z",
  "AvailabilityZones": [
    "us-west-2a"
  ],
  "Scheme": "internet-facing",
  "SourceSecurityGroup": {
    "OwnerAlias": "123456789012",
    "GroupName": "my-elb-sg"
  }
}
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoadBalancers](#)中的。

## describe-tags

以下代码示例演示了如何使用 describe-tags。

### AWS CLI

描述分配给负载均衡器的标签

此示例描述了分配给指定负载均衡器的标签。

命令:

```
aws elb describe-tags --load-balancer-name my-load-balancer
```

输出：

```
{
  "TagDescriptions": [
    {
      "Tags": [
        {
          "Value": "lima",
          "Key": "project"
        },
        {
          "Value": "digital-media",
          "Key": "department"
        }
      ],
      "LoadBalancerName": "my-load-balancer"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTags](#)中的。

## detach-load-balancer-from-subnets

以下代码示例演示了如何使用 detach-load-balancer-from-subnets。

AWS CLI

将负载均衡器与子网分离

此示例将指定的负载均衡器与指定子网分离。

命令：

```
aws elb detach-load-balancer-from-subnets --load-balancer-name my-load-balancer --
subnets subnet-0ecac448
```

输出：



```
{
  "Subnets": [
    "subnet-15aaab61"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachLoadBalancerFromSubnets](#)中的。

## **disable-availability-zones-for-load-balancer**

以下代码示例演示了如何使用 `disable-availability-zones-for-load-balancer`。

### AWS CLI

为负载均衡器禁用可用区

此示例将指定的可用区域从指定负载均衡器的可用区域集中移除。

命令:

```
aws elb disable-availability-zones-for-load-balancer --load-balancer-name my-load-balancer --availability-zones us-west-2a
```

输出:

```
{
  "AvailabilityZones": [
    "us-west-2b"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableAvailabilityZonesForLoadBalancer](#)中的。

## **enable-availability-zones-for-load-balancer**

以下代码示例演示了如何使用 `enable-availability-zones-for-load-balancer`。

### AWS CLI

为负载均衡器启用可用区

此示例将指定的可用区添加到指定的负载均衡器。

命令:

```
aws elb enable-availability-zones-for-load-balancer --load-balancer-name my-load-balancer --availability-zones us-west-2b
```

输出 :

```
{
  "AvailabilityZones": [
    "us-west-2a",
    "us-west-2b"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableAvailabilityZonesForLoadBalancer](#)中的。

## modify-load-balancer-attributes

以下代码示例演示了如何使用 modify-load-balancer-attributes。

### AWS CLI

修改负载均衡器的属性

此示例修改了指定负载均衡器的CrossZoneLoadBalancing属性。

命令:

```
aws elb modify-load-balancer-attributes --load-balancer-name my-load-balancer --load-balancer-attributes "{\"CrossZoneLoadBalancing\":{\"Enabled\":true}}"
```

输出 :

```
{
  "LoadBalancerAttributes": {
    "CrossZoneLoadBalancing": {
      "Enabled": true
    }
  }
}
```

```
    }
  },
  "LoadBalancerName": "my-load-balancer"
}
```

此示例修改了指定负载均衡器的ConnectionDraining属性。

命令:

```
aws elb modify-load-balancer-attributes --load-balancer-name my-load-balancer
--load-balancer-attributes "{\"ConnectionDraining\":{\"Enabled\":true,\"Timeout
\":300}}"
```

输出:

```
{
  "LoadBalancerAttributes": {
    "ConnectionDraining": {
      "Enabled": true,
      "Timeout": 300
    }
  },
  "LoadBalancerName": "my-load-balancer"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyLoadBalancerAttributes](#)中的。

## register-instances-with-load-balancer

以下代码示例演示了如何使用 register-instances-with-load-balancer。

### AWS CLI

向负载均衡器注册实例

此示例将指定的实例注册到指定的负载均衡器。

命令:

```
aws elb register-instances-with-load-balancer --load-balancer-name my-load-balancer
--instances i-d6f6fae3
```

输出：

```
{
  "Instances": [
    {
      "InstanceId": "i-d6f6fae3"
    },
    {
      "InstanceId": "i-207d9717"
    },
    {
      "InstanceId": "i-afefb49b"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterInstancesWithLoadBalancer](#)中的。

## remove-tags

以下代码示例演示了如何使用 `remove-tags`。

### AWS CLI

从负载均衡器中移除标签

此示例从指定的负载均衡器中移除标签。

命令：

```
aws elb remove-tags --load-balancer-name my-load-balancer --tags project
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveTags](#)中的。

## set-load-balancer-listener-ssl-certificate

以下代码示例演示了如何使用 `set-load-balancer-listener-ssl-certificate`。

### AWS CLI

更新 HTTPS 负载均衡器的 SSL 证书

此示例替换了指定 HTTPS 负载均衡器的现有 SSL 证书。

命令:

```
aws elb set-load-balancer-listener-ssl-certificate --load-balancer-name my-load-balancer --load-balancer-port 443 --ssl-certificate-id arn:aws:iam::123456789012:server-certificate/new-server-cert
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetLoadBalancerListenerSslCertificate](#)中的。

## set-load-balancer-policies-for-backend-server

以下代码示例演示了如何使用 `set-load-balancer-policies-for-backend-server`。

AWS CLI

替换与后端实例的端口关联的策略

此示例替换了当前与指定端口关联的策略。

命令:

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-load-balancer --instance-port 80 --policy-names my-ProxyProtocol-policy
```

移除当前与您的后端实例上的某个端口关联的所有策略

此示例删除了与指定端口关联的所有策略。

命令:

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-load-balancer --instance-port 80 --policy-names []
```

要确认策略已删除，请使用 `describe-load-balancer-policies` 命令。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetLoadBalancerPoliciesForBackendServer](#)中的。

## set-load-balancer-policies-of-listener

以下代码示例演示了如何使用 `set-load-balancer-policies-of-listener`。

## AWS CLI

### 替换与监听器关联的策略

此示例替换了当前与指定侦听器关联的策略。

命令:

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-load-balancer
--load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

### 删除与您的监听器关联的所有策略

此示例删除了当前与指定侦听器关联的所有策略。

命令:

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-load-balancer
--load-balancer-port 443 --policy-names []
```

要确认策略已从负载均衡器中删除，请使用 `describe-load-balancer-policies` 命令。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SetLoadBalancerPoliciesOfListener](#) 中的。

## Elastic Load Balancing-版本 2 示例使用 AWS CLI

以下代码示例向您展示了如何在 Elastic Load Balancing-版本 2 中 AWS Command Line Interface 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

## 操作

### add-listener-certificates

以下代码示例演示了如何使用 `add-listener-certificates`。

#### AWS CLI

向安全侦听器添加证书

此示例将指定的证书添加到指定的安全侦听器。

命令:

```
aws elbv2 add-listener-certificates --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 --certificates CertificateArn=arn:aws:acm:us-west-2:123456789012:certificate/5cc54884-f4a3-4072-80be-05b9ba72f705
```

输出:

```
{
  "Certificates": [
    {
      "CertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/5cc54884-f4a3-4072-80be-05b9ba72f705",
      "IsDefault": false
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddListenerCertificates](#)中的。

### add-tags

以下代码示例演示了如何使用 `add-tags`。

#### AWS CLI

向负载均衡器添加标签

以下`add-tags`示例将`project`和`department`标签添加到指定的负载均衡器。

```
aws elbv2 add-tags \  
  --resource-arns arn:aws:elasticloadbalancing:us-  
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \  
  --tags "Key=project,Value=lima" "Key=department,Value=digital-media"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddTags](#)中的。

## create-listener

以下代码示例演示了如何使用 `create-listener`。

### AWS CLI

#### 示例 1：创建 HTTP 侦听器

以下 `create-listener` 示例为指定的应用程序负载均衡器创建一个 HTTP 侦听器，用于将请求转发到指定的目标组。

```
aws elbv2 create-listener \  
  --load-balancer-arn arn:aws:elasticloadbalancing:us-  
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \  
  --protocol HTTP \  
  --port 80 \  
  --default-actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

有关更多信息，请参阅 [《应用程序负载均衡器用户指南》中的教程：使用 AWS CLI 创建应用程序负载均衡器](#)。

#### 示例 1：创建 HTTPS 侦听器

以下 `create-listener` 示例为指定的应用程序负载均衡器创建一个 HTTPS 侦听器，用于将请求转发到指定的目标组。您必须指定 HTTPS 侦听器的 SSL 证书。您可以使用 Certificate Manager (ACM) 创建和管理 AWS 证书。或者，您可以使用 SSL/TLS 工具创建证书，获取由证书颁发机构 (CA) 签名的证书，然后将证书上传到 AWS 身份与访问管理 (IAM) and Access Management。

```
aws elbv2 create-listener \  
  --load-balancer-arn arn:aws:elasticloadbalancing:us-  
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \  
  --protocol HTTPS \  
  --port 443 \  
  --ssl-certificates arn:aws:iam::123456789012:certificate/12345678-1234-1234-1234-123456789012
```



```
--certificates CertificateArn=arn:aws:acm:us-west-2:123456789012:certificate/3dcb0a41-bd72-4774-9ad9-756919c40557 \  
--ssl-policy ELBSecurityPolicy-2016-08 \  
--default-actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

有关更多信息，请参阅《应用程序负载均衡器用户指南》中的[添加 HTTPS 侦听器](#)。

### 示例 3：创建 TCP 侦听器

以下 `create-listener` 示例为指定的网络负载均衡器创建一个 TCP 侦听器，用于将请求转发到指定的目标组。

```
aws elbv2 create-listener \  
  --load-balancer-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/net/my-network-load-balancer/5d1b75f4f1cee11e \  
  --protocol TCP \  
  --port 80 \  
  --default-actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-tcp-targets/b6bba954d1361c78
```

有关更多信息，请参阅[网络负载均衡器用户指南中的教程：使用 AWS CLI 创建网络负载均衡器](#)。

### 示例 4：创建 TLS 侦听器

以下 `create-listener` 示例为指定的网络负载均衡器创建一个 TLS 侦听器，用于将请求转发到指定的目标组。您必须指定 TLS 侦听器的 SSL 证书。

```
aws elbv2 create-listener \  
  --load-balancer-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \  
  --protocol TLS \  
  --port 443 \  
  --certificates CertificateArn=arn:aws:acm:us-west-2:123456789012:certificate/3dcb0a41-bd72-4774-9ad9-756919c40557 \  
  --ssl-policy ELBSecurityPolicy-2016-08 \  
  --default-actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

有关更多信息，请参阅《网络负载均衡器用户指南》中的[网络负载均衡器的 TLS 侦听器](#)。

### 示例 5：创建 UDP 侦听器

以下 `create-listener` 示例为指定的网络负载均衡器创建一个 UDP 侦听器，用于将请求转发到指定的目标组。

```
aws elbv2 create-listener \
  --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/net/my-network-load-balancer/5d1b75f4f1cee11e \
  --protocol UDP \
  --port 53 \
  --default-actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-tcp-targets/b6bba954d1361c78
```

有关更多信息，请参阅[网络负载均衡器用户指南中的教程：使用 AWS CLI 创建网络负载均衡器](#)。

示例 6：为指定的网关和转发创建侦听器

以下 `create-listener` 示例为指定的网关负载均衡器创建一个侦听器，用于将请求转发到指定的目标组。

```
aws elbv2 create-listener \
  --load-balancer-arn arn:aws:elasticloadbalancing:us-
east-1:850631746142:loadbalancer/gwy/my-gateway-load-balancer/e0f9b3d5c7f7d3d6 \
  --default-actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
east-1:850631746142:targetgroup/my-glb-targets/007ca469fae3bb1615
```

输出：

```
{
  "Listeners": [
    {
      "ListenerArn": "arn:aws:elasticloadbalancing:us-
east-1:850631746142:listener/gwy/my-agw-lb-example2/e0f9b3d5c7f7d3d6/
afc127db15f925de",
      "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
east-1:850631746142:loadbalancer/gwy/my-agw-lb-example2/e0f9b3d5c7f7d3d6",
      "DefaultActions": [
        {
          "Type": "forward",
          "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
east-1:850631746142:targetgroup/test-tg-agw-2/007ca469fae3bb1615",
          "ForwardConfig": {
            "TargetGroups": [
              {
```

```
                                "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
                                east-1:850631746142:targetgroup/test-tg-agw-2/007ca469fae3bb1615"
                                }
                                ]
                                }
                                ]
                                }
                                ]
                                }
```

有关更多信息，请参阅[网关负载均衡器用户指南中的使用 AWS CLI 使用网关负载均衡器入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateListener](#) 中的。

## create-load-balancer

以下代码示例演示了如何使用 create-load-balancer。

### AWS CLI

示例 1：创建面向 Internet 的负载均衡器

以下 create-load-balancer 示例创建一个面向 Internet 的应用程序负载均衡器，并为指定的子网启用可用区。

```
aws elbv2 create-load-balancer \
  --name my-load-balancer \
  --subnets subnet-b7d581c0 subnet-8360a9e7
```

输出：

```
{
  "LoadBalancers": [
    {
      "Type": "application",
      "Scheme": "internet-facing",
      "IpAddressType": "ipv4",
      "VpcId": "vpc-3ac0fb5f",
      "AvailabilityZones": [
        {
          "ZoneName": "us-west-2a",
          "SubnetId": "subnet-8360a9e7"
        }
      ]
    }
  ]
}
```

```

    },
    {
      "ZoneName": "us-west-2b",
      "SubnetId": "subnet-b7d581c0"
    }
  ],
  "CreatedTime": "2017-08-25T21:26:12.920Z",
  "CanonicalHostedZoneId": "Z2P70J7EXAMPLE",
  "DNSName": "my-load-balancer-424835706.us-west-2.elb.amazonaws.com",
  "SecurityGroups": [
    "sg-5943793c"
  ],
  "LoadBalancerName": "my-load-balancer",
  "State": {
    "Code": "provisioning"
  },
  "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188"
}
]
}

```

有关更多信息，请参阅 [《应用程序负载均衡器用户指南》](#) 中的教程：[使用 AWS CLI 创建应用程序负载均衡器](#)。

#### 示例 2：创建内部负载均衡器

以下 `create-load-balancer` 示例创建一个内部应用程序负载均衡器，并为指定的子网启用可用区。

```

aws elbv2 create-load-balancer \
  --name my-internal-load-balancer \
  --scheme internal \
  --subnets subnet-b7d581c0 subnet-8360a9e7

```

输出：

```

{
  "LoadBalancers": [
    {
      "Type": "application",
      "Scheme": "internal",

```

```
    "IpAddressType": "ipv4",
    "VpcId": "vpc-3ac0fb5f",
    "AvailabilityZones": [
      {
        "ZoneName": "us-west-2a",
        "SubnetId": "subnet-8360a9e7"
      },
      {
        "ZoneName": "us-west-2b",
        "SubnetId": "subnet-b7d581c0"
      }
    ],
    "CreatedTime": "2016-03-25T21:29:48.850Z",
    "CanonicalHostedZoneId": "Z2P70J7EXAMPLE",
    "DNSName": "internal-my-internal-load-balancer-1529930873.us-
west-2.elb.amazonaws.com",
    "SecurityGroups": [
      "sg-5943793c"
    ],
    "LoadBalancerName": "my-internal-load-balancer",
    "State": {
      "Code": "provisioning"
    },
    "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-internal-load-balancer/5b49b8d4303115c2"
  }
]
```

有关更多信息，请参阅 [《应用程序负载均衡器用户指南》](#) 中的教程：[使用 AWS CLI 创建应用程序负载均衡器](#)。

### 示例 3：创建网络负载均衡器

以下 `create-load-balancer` 示例创建一个面向 Internet 的网络负载均衡器，并为指定的子网启用可用区。它使用子网映射，将指定的弹性 IP 地址与可用区的负载均衡器节点使用的网络接口相关联。

```
aws elbv2 create-load-balancer \
  --name my-network-load-balancer \
  --type network \
  --subnet-mappings SubnetId=subnet-b7d581c0,AllocationId=eipalloc-64d5890a
```

输出：

```
{
  "LoadBalancers": [
    {
      "Type": "network",
      "Scheme": "internet-facing",
      "IpAddressType": "ipv4",
      "VpcId": "vpc-3ac0fb5f",
      "AvailabilityZones": [
        {
          "LoadBalancerAddresses": [
            {
              "IpAddress": "35.161.207.171",
              "AllocationId": "eipalloc-64d5890a"
            }
          ],
          "ZoneName": "us-west-2b",
          "SubnetId": "subnet-5264e837"
        }
      ],
      "CreatedTime": "2017-10-15T22:41:25.657Z",
      "CanonicalHostedZoneId": "Z2P70J7EXAMPLE",
      "DNSName": "my-network-load-balancer-5d1b75f4f1cee11e.elb.us-
west-2.amazonaws.com",
      "LoadBalancerName": "my-network-load-balancer",
      "State": {
        "Code": "provisioning"
      },
      "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/net/my-network-load-balancer/5d1b75f4f1cee11e"
    }
  ]
}
```

有关更多信息，请参阅[网络负载均衡器用户指南中的教程：使用 AWS CLI 创建网络负载均衡器](#)。

#### 示例 4：创建网关负载均衡器

以下 `create-load-balancer` 示例创建一个网关负载均衡器，并为指定的子网启用可用区。

```
aws elbv2 create-load-balancer \
  --name my-gateway-load-balancer \
```

```
--type gateway \  
--subnets subnet-dc83f691 subnet-a62583f9
```

输出：

```
{  
  "LoadBalancers": [  
    {  
      "Type": "gateway",  
      "VpcId": "vpc-838475fe",  
      "AvailabilityZones": [  
        {  
          "ZoneName": "us-east-1b",  
          "SubnetId": "subnet-a62583f9"  
        },  
        {  
          "ZoneName": "us-east-1a",  
          "SubnetId": "subnet-dc83f691"  
        }  
      ],  
      "CreatedTime": "2021-07-14T19:33:43.324000+00:00",  
      "LoadBalancerName": "my-gateway-load-balancer",  
      "State": {  
        "Code": "provisioning"  
      },  
      "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-  
east-1:850631746142:loadbalancer/gwy/my-gateway-load-balancer/dfbb5a7d32cdee79"  
    }  
  ]  
}
```

有关更多信息，请参阅[网关负载均衡器用户指南中的使用 AWS CLI 使用网关负载均衡器入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateLoadBalancer](#) 中的。

## create-rule

以下代码示例演示了如何使用 create-rule。

### AWS CLI

示例 1：使用路径条件和转发操作创建规则

以下create-rule示例创建了一个规则，如果 URL 包含指定的模式，则该规则将请求转发到指定的目标组。

```
aws elbv2 create-rule \  
  --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/  
my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \  
  --priority 5 \  
  --conditions file://conditions-pattern.json \  
  --actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

conditions-pattern.json 的内容：

```
[  
  {  
    "Field": "path-pattern",  
    "PathPatternConfig": {  
      "Values": ["/images/*"]  
    }  
  }  
]
```

示例 2：使用主机条件和固定响应创建规则

以下create-rule示例创建了一个规则，如果主机标头中的主机名与指定的主机名匹配，则提供固定响应。

```
aws elbv2 create-rule \  
  --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/  
my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \  
  --priority 10 \  
  --conditions file://conditions-host.json \  
  --actions file://actions-fixed-response.json
```

conditions-host.json 的内容

```
[  
  {  
    "Field": "host-header",  
    "HostHeaderConfig": {  
      "Values": ["*.example.com"]  
    }  
  }  
]
```



```
    }  
  }  
]
```

### actions-fixed-response.json 的内容

```
[  
  {  
    "Type": "fixed-response",  
    "FixedResponseConfig": {  
      "MessageBody": "Hello world",  
      "StatusCode": "200",  
      "ContentType": "text/plain"  
    }  
  }  
]
```

### 示例 3：使用源 IP 地址条件、身份验证操作和转发操作创建规则

以下 `create-rule` 示例创建了一个规则，该规则用于在源 IP 地址与指定的 IP 地址匹配时对用户进行身份验证，如果身份验证成功，则将请求转发到指定的目标组。

```
aws elbv2 create-rule \  
  --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/  
my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \  
  --priority 20 \  
  --conditions file://conditions-source-ip.json \  
  --actions file://actions-authenticate.json
```

### conditions-source-ip.json 的内容

```
[  
  {  
    "Field": "source-ip",  
    "SourceIpConfig": {  
      "Values": ["192.0.2.0/24", "198.51.100.10/32"]  
    }  
  }  
]
```

### actions-authenticate.json 的内容

```
[
  {
    "Type": "authenticate-oidc",
    "AuthenticateOidcConfig": {
      "Issuer": "https://idp-issuer.com",
      "AuthorizationEndpoint": "https://authorization-endpoint.com",
      "TokenEndpoint": "https://token-endpoint.com",
      "UserInfoEndpoint": "https://user-info-endpoint.com",
      "ClientId": "abcdefghijklmnopqrstuvwxy123456789",
      "ClientSecret": "123456789012345678901234567890",
      "SessionCookieName": "my-cookie",
      "SessionTimeout": 3600,
      "Scope": "email",
      "AuthenticationRequestExtraParams": {
        "display": "page",
        "prompt": "login"
      },
      "OnUnauthenticatedRequest": "deny"
    },
    "Order": 1
  },
  {
    "Type": "forward",
    "TargetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:880185128111:targetgroup/cli-test/642a97ecb0e0f26b",
    "Order": 2
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRule](#)中的。

## create-target-group

以下代码示例演示了如何使用 create-target-group。

### AWS CLI

示例 1：为 Application Load Balancer 创建目标组

以下 create-target-group 示例为应用程序负载均衡器创建目标组，以便您按实例 ID（目标类型为 instance）注册目标。此目标组使用 HTTP 协议、端口 80 和 HTTP 目标组的默认运行状况检查设置。

```
aws elbv2 create-target-group \  
  --name my-targets \  
  --protocol HTTP \  
  --port 80 \  
  --target-type instance \  
  --vpc-id vpc-3ac0fb5f
```

输出：

```
{  
  "TargetGroups": [  
    {  
      "TargetGroupArn": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",  
      "TargetGroupName": "my-targets",  
      "Protocol": "HTTP",  
      "Port": 80,  
      "VpcId": "vpc-3ac0fb5f",  
      "HealthCheckProtocol": "HTTP",  
      "HealthCheckPort": "traffic-port",  
      "HealthCheckEnabled": true,  
      "HealthCheckIntervalSeconds": 30,  
      "HealthCheckTimeoutSeconds": 5,  
      "HealthyThresholdCount": 5,  
      "UnhealthyThresholdCount": 2,  
      "HealthCheckPath": "/",  
      "Matcher": {  
        "HttpCode": "200"  
      },  
      "TargetType": "instance",  
      "ProtocolVersion": "HTTP1",  
      "IpAddressType": "ipv4"  
    }  
  ]  
}
```

有关更多信息，请参阅《应用程序负载均衡器用户指南》中的[创建目标组](#)。

示例 2：创建目标组以将流量从 Application Load Balancer 路由到 Lambda 函数

以下 create-target-group 示例为应用程序负载均衡器创建目标组，在其中目标为 Lambda 函数（目标类型为 lambda）。默认情况下，为此目标组禁用运行状况检查。

```
aws elbv2 create-target-group \  
  --name my-lambda-target \  
  --target-type lambda
```

输出：

```
{  
  "TargetGroups": [  
    {  
      "TargetGroupArn": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-lambda-target/a3003e085dbb8ddc",  
      "TargetGroupName": "my-lambda-target",  
      "HealthCheckEnabled": false,  
      "HealthCheckIntervalSeconds": 35,  
      "HealthCheckTimeoutSeconds": 30,  
      "HealthyThresholdCount": 5,  
      "UnhealthyThresholdCount": 2,  
      "HealthCheckPath": "/",  
      "Matcher": {  
        "HttpCode": "200"  
      },  
      "TargetType": "lambda",  
      "IpAddressType": "ipv4"  
    }  
  ]  
}
```

有关更多信息，请参阅应用程序负载均衡器用户指南中的 [Lambda 函数作为目标](#)。

示例 3：为 Network Load Balancer 创建目标组

以下 create-target-group 示例为网络负载均衡器创建目标组，以便您按 IP 地址（目标类型为 ip）注册目标。此目标组使用 TCP 协议、端口 80 和 TCP 目标组的默认运行状况检查设置。

```
aws elbv2 create-target-group \  
  --name my-ip-targets \  
  --protocol TCP \  
  --port 80 \  
  --target-type ip \  
  --vpc-id vpc-3ac0fb5f
```

输出：

```
{
  "TargetGroups": [
    {
      "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-ip-targets/b6bba954d1361c78",
      "TargetGroupName": "my-ip-targets",
      "Protocol": "TCP",
      "Port": 80,
      "VpcId": "vpc-3ac0fb5f",
      "HealthCheckEnabled": true,
      "HealthCheckProtocol": "TCP",
      "HealthCheckPort": "traffic-port",
      "HealthCheckIntervalSeconds": 30,
      "HealthCheckTimeoutSeconds": 10,
      "HealthyThresholdCount": 5,
      "UnhealthyThresholdCount": 2,
      "TargetType": "ip",
      "IpAddressType": "ipv4"
    }
  ]
}
```

有关更多信息，请参阅网络负载均衡器用户指南中的[创建目标组](#)。

示例 4：创建目标组以将流量从网络负载均衡器路由到 Application Load Balancer

以下create-target-group示例为网络负载均衡器创建目标组，您将在其中将应用程序负载均衡器注册为目标（目标类型为alb）。

```
aws elbv2 create-target-group --name--protocol TCP--端口 80--target-type alb my-alb-target --vpc-
id vpc-3ac0fb5f
```

输出：

```
{
  "TargetGroups": [
    {
      "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-alb-target/a3003e085dbb8ddc",
      "TargetGroupName": "my-alb-target",
      "Protocol": "TCP",
      "Port": 80,
      "VpcId": "vpc-838475fe",
```

```

    "HealthCheckProtocol": "HTTP",
    "HealthCheckPort": "traffic-port",
    "HealthCheckEnabled": true,
    "HealthCheckIntervalSeconds": 30,
    "HealthCheckTimeoutSeconds": 6,
    "HealthyThresholdCount": 5,
    "UnhealthyThresholdCount": 2,
    "HealthCheckPath": "/",
    "Matcher": {
      "HttpCode": "200-399"
    },
    "TargetType": "alb",
    "IpAddressType": "ipv4"
  }
]
}

```

有关更多信息，请参阅网络负载均衡器用户指南中的创建以 Application Load Balancer 作为目标的目标组。

#### 示例 5：为 Gateway Load Balancer 创建目标组

以下 `create-target-group` 示例为 Gateway Load Balancer 创建目标组，其中目标为实例，目标组协议为 GENEVE。

```

aws elbv2 create-target-group \
  --name my-glb-targetgroup \
  --protocol GENEVE \
  --port 6081 \
  --target-type instance \
  --vpc-id vpc-838475fe

```

输出：

```

{
  "TargetGroups": [
    {
      "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-glb-targetgroup/00c3d57eacd6f40b6f",
      "TargetGroupName": "my-glb-targetgroup",
      "Protocol": "GENEVE",
      "Port": 6081,

```

```
    "VpcId": "vpc-838475fe",
    "HealthCheckProtocol": "TCP",
    "HealthCheckPort": "80",
    "HealthCheckEnabled": true,
    "HealthCheckIntervalSeconds": 10,
    "HealthCheckTimeoutSeconds": 5,
    "HealthyThresholdCount": 5,
    "UnhealthyThresholdCount": 2,
    "TargetType": "instance"
  }
]
```

有关更多信息，请参阅 Gateway Load Balancer 用户指南中的创建目标组 < <https://docs.aws.amazon.com/elasticloadbalancing/latest/gateway/create-target-group.html> > 。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateTargetGroup](#) 中的。

## delete-listener

以下代码示例演示了如何使用 delete-listener。

### AWS CLI

#### 删除监听器

以下 delete-listener 示例删除了指定的监听器。

```
aws elbv2 delete-listener \
  --listener-arn arn:aws:elasticloadbalancing:ua-west-2:123456789012:listener/app/
my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteListener](#) 中的。

## delete-load-balancer

以下代码示例演示了如何使用 delete-load-balancer。

### AWS CLI

#### 删除负载均衡器

以下 `delete-load-balancer` 示例将删除指定的负载均衡器。

```
aws elbv2 delete-load-balancer \  
  --load-balancer-arn arn:aws:elasticloadbalancing:us-  
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLoadBalancer](#)中的。

## delete-rule

以下代码示例演示了如何使用 `delete-rule`。

### AWS CLI

#### 删除规则

以下`delete-rule`示例删除了指定的规则。

```
aws elbv2 delete-rule \  
  --rule-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener-rule/  
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2/1291d13826f405c3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRule](#)中的。

## delete-target-group

以下代码示例演示了如何使用 `delete-target-group`。

### AWS CLI

#### 删除目标组

以下 `delete-target-group` 示例将删除指定的目标组。

```
aws elbv2 delete-target-group \  
  --target-group-arn arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

此命令不生成任何输出。

有关更多信息，请参阅 [Application Load Balancer 指南中的删除负载均衡器](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTargetGroup](#)中的。

## deregister-targets

以下代码示例演示了如何使用 `deregister-targets`。

### AWS CLI

示例 1：从目标组中注销目标

以下`deregister-targets`示例将指定实例从指定的目标组中移除。

```
aws elbv2 deregister-targets \  
  --target-group-arn arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067 \  
  --targets Id=i-1234567890abcdef0
```

示例 2：取消注册使用端口覆盖注册的目标

以下`deregister-targets`示例从使用端口覆盖注册的目标组中删除一个实例。

```
aws elbv2 deregister-targets \  
  --target-group-arn arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-internal-targets/3bb63f11dfb0faf9 \  
  --targets Id=i-1234567890abcdef0,Port=80 Id=i-1234567890abcdef0,Port=766
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterTargets](#)中的。

## describe-account-limits

以下代码示例演示了如何使用 `describe-account-limits`。

### AWS CLI

描述您的 Elastic Load Balancing 限制

以下`describe-account-limits`示例显示了您在当前区域的 AWS 账户的 Elastic Load Balancing 限制。

```
aws elbv2 describe-account-limits
```

输出：

```
{
  "Limits": [
    {
      "Name": "target-groups",
      "Max": "3000"
    },
    {
      "Name": "targets-per-application-load-balancer",
      "Max": "1000"
    },
    {
      "Name": "listeners-per-application-load-balancer",
      "Max": "50"
    },
    {
      "Name": "rules-per-application-load-balancer",
      "Max": "100"
    },
    {
      "Name": "network-load-balancers",
      "Max": "50"
    },
    {
      "Name": "targets-per-network-load-balancer",
      "Max": "3000"
    },
    {
      "Name": "targets-per-availability-zone-per-network-load-balancer",
      "Max": "500"
    },
    {
      "Name": "listeners-per-network-load-balancer",
      "Max": "50"
    },
    {
      "Name": "condition-values-per-alb-rule",
      "Max": "5"
    },
    {
      "Name": "condition-wildcards-per-alb-rule",
      "Max": "5"
    },
  ],
}
```

```
{
  "Name": "target-groups-per-application-load-balancer",
  "Max": "100"
},
{
  "Name": "target-groups-per-action-on-application-load-balancer",
  "Max": "5"
},
{
  "Name": "target-groups-per-action-on-network-load-balancer",
  "Max": "1"
},
{
  "Name": "certificates-per-application-load-balancer",
  "Max": "25"
},
{
  "Name": "certificates-per-network-load-balancer",
  "Max": "25"
},
{
  "Name": "targets-per-target-group",
  "Max": "1000"
},
{
  "Name": "target-id-registrations-per-application-load-balancer",
  "Max": "1000"
},
{
  "Name": "network-load-balancer-enis-per-vpc",
  "Max": "1200"
},
{
  "Name": "application-load-balancers",
  "Max": "50"
},
{
  "Name": "gateway-load-balancers",
  "Max": "100"
},
{
  "Name": "gateway-load-balancers-per-vpc",
  "Max": "100"
},
},
```

```
{
  "Name": "geneve-target-groups",
  "Max": "100"
},
{
  "Name": "targets-per-availability-zone-per-gateway-load-balancer",
  "Max": "300"
}
]
```

有关更多信息，请参阅《AWS 一般参考》中的[配额](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAccountLimits](#) 中的。

## describe-listener-certificates

以下代码示例演示了如何使用 `describe-listener-certificates`。

### AWS CLI

描述安全侦听器的证书

此示例描述了指定安全侦听器的证书。

命令:

```
aws elbv2 describe-listener-certificates --listener-arn
arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/my-load-
balancer/50dc6c495c0c9188/f2f7dc8efc522ab2
```

输出:

```
{
  "Certificates": [
    {
      "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/5cc54884-f4a3-4072-80be-05b9ba72f705",
      "IsDefault": false
    },
    {
      "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/3dcb0a41-bd72-4774-9ad9-756919c40557",
```

```
        "IsDefault": false
      },
      {
        "CertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/
fe59da96-6f58-4a22-8eed-6d0d50477e1d",
        "IsDefault": true
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeListenerCertificates](#)中的。

## describe-listeners

以下代码示例演示了如何使用 describe-listeners。

### AWS CLI

#### 描述听众

此示例描述了指定的监听器。

#### 命令:

```
aws elbv2 describe-listeners --listener-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2
```

#### 输出:

```
{
  "Listeners": [
    {
      "Port": 80,
      "Protocol": "HTTP",
      "DefaultActions": [
        {
          "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
          "Type": "forward"
        }
      ],
      "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
    }
  ]
}
```

```
    "ListenerArn": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2"  
  }  
]  
}
```

## 描述负载均衡器的监听器

此示例描述了指定负载均衡器的侦听器。

### 命令:

```
aws elbv2 describe-listeners --load-balancer-arn arn:aws:elasticloadbalancing:us-  
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188
```

### 输出:

```
{  
  "Listeners": [  
    {  
      "Port": 443,  
      "Protocol": "HTTPS",  
      "DefaultActions": [  
        {  
          "TargetGroupArn": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",  
          "Type": "forward"  
        }  
      ],  
      "SslPolicy": "ELBSecurityPolicy-2015-05",  
      "Certificates": [  
        {  
          "CertificateArn": "arn:aws:iam::123456789012:server-certificate/  
my-server-cert"  
        }  
      ],  
      "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",  
      "ListenerArn": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/0467ef3c8400ae65"  
    },  
    {  
      "Port": 80,
```

```

    "Protocol": "HTTP",
    "DefaultActions": [
      {
        "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
        "Type": "forward"
      }
    ],
    "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
    "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeListeners](#)中的。

## describe-load-balancer-attributes

以下代码示例演示了如何使用 describe-load-balancer-attributes。

### AWS CLI

描述负载均衡器属性

以下describe-load-balancer-attributes示例显示了指定负载均衡器的属性。

```

aws elbv2 describe-load-balancer-attributes \
  --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188

```

以下示例输出显示了 Application Load Balancer 的属性。

```

{
  "Attributes": [
    {
      "Value": "false",
      "Key": "access_logs.s3.enabled"
    },
    {
      "Value": "",

```

```

        "Key": "access_logs.s3.bucket"
    },
    {
        "Value": "",
        "Key": "access_logs.s3.prefix"
    },
    {
        "Value": "60",
        "Key": "idle_timeout.timeout_seconds"
    },
    {
        "Value": "false",
        "Key": "deletion_protection.enabled"
    },
    {
        "Value": "true",
        "Key": "routing.http2.enabled"
    }
]
}

```

以下示例输出包括 Network Load Balancer 的属性。

```

{
  "Attributes": [
    {
      "Value": "false",
      "Key": "access_logs.s3.enabled"
    },
    {
      "Value": "",
      "Key": "access_logs.s3.bucket"
    },
    {
      "Value": "",
      "Key": "access_logs.s3.prefix"
    },
    {
      "Value": "false",
      "Key": "deletion_protection.enabled"
    },
    {
      "Value": "false",

```



```
        "Key": "load_balancing.cross_zone.enabled"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoadBalancerAttributes](#)中的。

## describe-load-balancers

以下代码示例演示了如何使用 describe-load-balancers。

### AWS CLI

#### 描述负载均衡器

此示例描述指定的负载均衡器。

命令:

```
aws elbv2 describe-load-balancers --load-balancer-arns
arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188
```

输出:

```
{
  "LoadBalancers": [
    {
      "Type": "application",
      "Scheme": "internet-facing",
      "IpAddressType": "ipv4",
      "VpcId": "vpc-3ac0fb5f",
      "AvailabilityZones": [
        {
          "ZoneName": "us-west-2a",
          "SubnetId": "subnet-8360a9e7"
        },
        {
          "ZoneName": "us-west-2b",
          "SubnetId": "subnet-b7d581c0"
        }
      ]
    }
  ],
}
```

```
    "CreatedTime": "2016-03-25T21:26:12.920Z",
    "CanonicalHostedZoneId": "Z2P70J7EXAMPLE",
    "DNSName": "my-load-balancer-424835706.us-west-2.elb.amazonaws.com",
    "SecurityGroups": [
      "sg-5943793c"
    ],
    "LoadBalancerName": "my-load-balancer",
    "State": {
      "Code": "active"
    },
    "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188"
  }
]
}
```

## 描述所有负载均衡器

此示例描述您的所有负载均衡器。

命令:

```
aws elbv2 describe-load-balancers
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoadBalancers](#)中的。

## describe-rules

以下代码示例演示了如何使用 describe-rules。

### AWS CLI

#### 示例 1：描述规则

以下describe-rules示例显示了指定规则的详细信息。

```
aws elbv2 describe-rules \  
  --rule-arns arn:aws:elasticloadbalancing:us-west-2:123456789012:listener-rule/  
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2/9683b2d02a6cabee
```

#### 示例 2：描述监听器的规则

以下describe-rules示例显示了指定侦听器的规则的详细信息。输出包括默认规则和您添加的任何其他规则。

```
aws elbv2 describe-rules \  
  --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/  
my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeRules](#)中的。

## describe-ssl-policies

以下代码示例演示了如何使用 describe-ssl-policies。

### AWS CLI

示例 1：按负载均衡器类型列出用于 SSL 协商的策略

以下describe-ssl-policies示例显示了可用于与 Application Load Balancer 进行 SSL 协商的策略的名称。该示例使用--query参数仅显示策略的名称。

```
aws elbv2 describe-ssl-policies \  
  --load-balancer-type application \  
  --query SslPolicies[*].Name
```

输出：

```
[  
  "ELBSecurityPolicy-2016-08",  
  "ELBSecurityPolicy-TLS13-1-2-2021-06",  
  "ELBSecurityPolicy-TLS13-1-2-Res-2021-06",  
  "ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06",  
  "ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06",  
  "ELBSecurityPolicy-TLS13-1-1-2021-06",  
  "ELBSecurityPolicy-TLS13-1-0-2021-06",  
  "ELBSecurityPolicy-TLS13-1-3-2021-06",  
  "ELBSecurityPolicy-TLS-1-2-2017-01",  
  "ELBSecurityPolicy-TLS-1-1-2017-01",  
  "ELBSecurityPolicy-TLS-1-2-Ext-2018-06",  
  "ELBSecurityPolicy-FS-2018-06",  
  "ELBSecurityPolicy-2015-05",  
  "ELBSecurityPolicy-TLS-1-0-2015-04",
```

```
"ELBSecurityPolicy-FS-1-2-Res-2019-08",  
"ELBSecurityPolicy-FS-1-1-2019-08",  
"ELBSecurityPolicy-FS-1-2-2019-08",  
"ELBSecurityPolicy-FS-1-2-Res-2020-10"  
]
```

### 示例 2：列出支持特定协议的策略

以下describe-ssl-policies示例显示支持 TLS 1.3 协议的策略的名称。该示例使用--query参数仅显示策略的名称。

```
aws elbv2 describe-ssl-policies \  
  --load-balancer-type application \  
  --query SslPolicies[?contains(SslProtocols,'TLSv1.3')].Name
```

输出：

```
[  
  "ELBSecurityPolicy-TLS13-1-2-2021-06",  
  "ELBSecurityPolicy-TLS13-1-2-Res-2021-06",  
  "ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06",  
  "ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06",  
  "ELBSecurityPolicy-TLS13-1-1-2021-06",  
  "ELBSecurityPolicy-TLS13-1-0-2021-06",  
  "ELBSecurityPolicy-TLS13-1-3-2021-06"  
]
```

### 示例 3：显示策略的密码

以下describe-ssl-policies示例显示指定策略的密码名称。该示例使用--query参数仅显示密码名称。列表中第一个密码的优先级为 1，其余密码按优先级顺序排列。

```
aws elbv2 describe-ssl-policies \  
  --names ELBSecurityPolicy-TLS13-1-2-2021-06 \  
  --query SslPolicies[*].Ciphers[*].Name
```

输出：

```
[  
  "TLS_AES_128_GCM_SHA256",
```

```

    "TLS_AES_256_GCM_SHA384",
    "TLS_CHACHA20_POLY1305_SHA256",
    "ECDHE-ECDSA-AES128-GCM-SHA256",
    "ECDHE-RSA-AES128-GCM-SHA256",
    "ECDHE-ECDSA-AES128-SHA256",
    "ECDHE-RSA-AES128-SHA256",
    "ECDHE-ECDSA-AES256-GCM-SHA384",
    "ECDHE-RSA-AES256-GCM-SHA384",
    "ECDHE-ECDSA-AES256-SHA384",
    "ECDHE-RSA-AES256-SHA384"
  ]

```

有关更多信息，请参阅应用程序负载均衡器用户指南中的[安全策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSslPolicies](#)中的。

## describe-tags

以下代码示例演示了如何使用 describe-tags。

### AWS CLI

描述分配给负载均衡器的标签

此示例描述了分配给指定负载均衡器的标签。

命令：

```
aws elbv2 describe-tags --resource-arns arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188
```

输出：

```
{
  "TagDescriptions": [
    {
      "ResourceArn": "arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
      "Tags": [
        {
          "Value": "lima",
          "Key": "project"
        }
      ]
    }
  ]
}
```

```

    },
    {
      "Value": "digital-media",
      "Key": "department"
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTags](#)中的。

## describe-target-group-attributes

以下代码示例演示了如何使用 describe-target-group-attributes。

### AWS CLI

描述目标群体的属性

以下describe-target-group-attributes示例显示了指定目标组的属性。

```

aws elbv2 describe-target-group-attributes \
  --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067

```

如果协议为 HTTP 或 HTTPS 且目标类型为或，则instance输出包含属性ip。

```

{
  "Attributes": [
    {
      "Value": "false",
      "Key": "stickiness.enabled"
    },
    {
      "Value": "300",
      "Key": "deregistration_delay.timeout_seconds"
    },
    {
      "Value": "lb_cookie",
      "Key": "stickiness.type"
    }
  ],
}

```

```

    {
      "Value": "86400",
      "Key": "stickiness.lb_cookie.duration_seconds"
    },
    {
      "Value": "0",
      "Key": "slow_start.duration_seconds"
    }
  ]
}

```

如果协议为 HTTP 或 HTTPS 且目标类型为 ，则以下输出包含属性 `lambda`。

```

{
  "Attributes": [
    {
      "Value": "false",
      "Key": "lambda.multi_value_headers.enabled"
    }
  ]
}

```

如果协议是 TCP、TLS、UDP 或 TCP\_UDP ，则以下输出将包含属性。

```

{
  "Attributes": [
    {
      "Value": "false",
      "Key": "proxy_protocol_v2.enabled"
    },
    {
      "Value": "300",
      "Key": "deregistration_delay.timeout_seconds"
    }
  ]
}

```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeTargetGroupAttributes](#) 中的。

## describe-target-groups

以下代码示例演示了如何使用 `describe-target-groups`。

## AWS CLI

### 示例 1：描述目标组

以下 `describe-target-groups` 示例显示指定目标组的详细信息。

```
aws elbv2 describe-target-groups \  
  --target-group-arns arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

输出：

```
{  
  "TargetGroups": [  
    {  
      "TargetGroupArn": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",  
      "TargetGroupName": "my-targets",  
      "Protocol": "HTTP",  
      "Port": 80,  
      "VpcId": "vpc-3ac0fb5f",  
      "HealthCheckProtocol": "HTTP",  
      "HealthCheckPort": "traffic-port",  
      "HealthCheckEnabled": true,  
      "HealthCheckIntervalSeconds": 30,  
      "HealthCheckTimeoutSeconds": 5,  
      "HealthyThresholdCount": 5,  
      "UnhealthyThresholdCount": 2,  
      "HealthCheckPath": "/",  
      "Matcher": {  
        "HttpCode": "200"  
      },  
      "LoadBalancerArns": [  
        "arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/  
app/my-load-balancer/50dc6c495c0c9188"  
      ],  
      "TargetType": "instance",  
      "ProtocolVersion": "HTTP1",  
      "IpAddressType": "ipv4"  
    }  
  ]  
}
```



## 示例 2：描述负载均衡器的所有目标组

以下 `describe-target-groups` 示例显示指定负载均衡器所有目标组的详细信息。该示例使用 `--query` 参数仅显示目标组名称。

```
aws elbv2 describe-target-groups \  
  --load-balancer-arn arn:aws:elasticloadbalancing:us-  
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \  
  --query TargetGroups[*].TargetGroupName
```

输出：

```
[  
  "my-instance-targets",  
  "my-ip-targets",  
  "my-lambda-target"  
]
```

有关更多信息，请参阅《应用程序负载均衡器指南》中的[目标组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeTargetGroups](#) 中的。

## describe-target-health

以下代码示例演示了如何使用 `describe-target-health`。

### AWS CLI

#### 示例 1：描述目标组中目标的运行状况

以下 `describe-target-health` 示例显示指定目标组中目标的运行状况详细信息。这些目标运行状况良好。

```
aws elbv2 describe-target-health \  
  --target-group-arn arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

输出：

```
{
```

```
    "TargetHealthDescriptions": [
      {
        "HealthCheckPort": "80",
        "Target": {
          "Id": "i-cedddcd4d",
          "Port": 80
        },
        "TargetHealth": {
          "State": "healthy"
        }
      },
      {
        "HealthCheckPort": "80",
        "Target": {
          "Id": "i-0f76fade",
          "Port": 80
        },
        "TargetHealth": {
          "State": "healthy"
        }
      }
    ]
  }
}
```

## 示例 2：描述目标的运行状况

以下 `describe-target-health` 示例显示指定目标的运行状况详细信息。此目标运行正常。

```
aws elbv2 describe-target-health \
  --targets Id=i-0f76fade,Port=80 \
  --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

输出：

```
{
  "TargetHealthDescriptions": [
    {
      "HealthCheckPort": "80",
      "Target": {
        "Id": "i-0f76fade",
        "Port": 80
      },

```

```

        "TargetHealth": {
            "State": "healthy"
        }
    ]
}

```

以下示例输出适用于未在侦听器的操作中指定目标组的目标。此目标无法接收来自负载均衡器的流量。

```

{
  "TargetHealthDescriptions": [
    {
      "HealthCheckPort": "80",
      "Target": {
        "Id": "i-0f76fade",
        "Port": 80
      },
      "TargetHealth": {
        "State": "unused",
        "Reason": "Target.NotInUse",
        "Description": "Target group is not configured to receive traffic
from the load balancer"
      }
    }
  ]
}

```

以下示例输出适用于仅在侦听器的操作中指定目标组的目标。该目标仍在注册中。

```

{
  "TargetHealthDescriptions": [
    {
      "HealthCheckPort": "80",
      "Target": {
        "Id": "i-0f76fade",
        "Port": 80
      },
      "TargetHealth": {
        "State": "initial",
        "Reason": "Elb.RegistrationInProgress",
        "Description": "Target registration is in progress"
      }
    }
  ]
}

```

```

    }
  ]
}

```

以下示例输出适用于运行状况不佳的目标。

```

{
  "TargetHealthDescriptions": [
    {
      "HealthCheckPort": "80",
      "Target": {
        "Id": "i-0f76fade",
        "Port": 80
      },
      "TargetHealth": {
        "State": "unhealthy",
        "Reason": "Target.Timeout",
        "Description": "Connection to target timed out"
      }
    }
  ]
}

```

以下示例输出针对的目标是 Lambda 函数且运行状况检查已禁用。

```

{
  "TargetHealthDescriptions": [
    {
      "Target": {
        "Id": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
        "AvailabilityZone": "all",
      },
      "TargetHealth": {
        "State": "unavailable",
        "Reason": "Target.HealthCheckDisabled",
        "Description": "Health checks are not enabled for this target"
      }
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTargetHealth](#)中的。

## modify-listener

以下代码示例演示了如何使用 `modify-listener`。

### AWS CLI

示例 1：将默认操作更改为转发操作

以下 `modify-listener` 示例将指定侦听器的默认操作（更改为转发操作）。

```
aws elbv2 modify-listener \  
  --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/  
my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \  
  --default-actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-new-targets/2453ed029918f21f
```

输出：

```
{  
  "Listeners": [  
    {  
      "Protocol": "HTTP",  
      "DefaultActions": [  
        {  
          "TargetGroupArn": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-new-targets/2453ed029918f21f",  
          "Type": "forward"  
        }  
      ],  
      "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",  
      "Port": 80,  
      "ListenerArn": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2"  
    }  
  ]  
}
```

示例 2：将默认操作更改为重定向操作

以下 `modify-listener` 示例将指定侦听器的默认操作更改为重定向操作。

```
aws elbv2 modify-listener \  

```

```
--listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/
my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \
--default-actions Type=redirect,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-new-targets/2453ed029918f21f
```

输出：

```
{
  "Listeners": [
    {
      "Protocol": "HTTP",
      "DefaultActions": [
        {
          "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-new-targets/2453ed029918f21f",
          "Type": "redirect"
        }
      ],
      "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
      "Port": 80,
      "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2"
    }
  ]
}
```

### 示例 3：更改服务器证书

此示例更改指定 HTTPS 侦听器的服务器证书。

```
aws elbv2 modify-listener \
--listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/
my-load-balancer/50dc6c495c0c9188/0467ef3c8400ae65 \
--certificates CertificateArn=arn:aws:iam::123456789012:server-certificate/my-
new-server-cert
```

输出：

```
{
  "Listeners": [
    {
```

```
    "Protocol": "HTTPS",
    "DefaultActions": [
      {
        "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
        "Type": "forward"
      }
    ],
    "SslPolicy": "ELBSecurityPolicy-2015-05",
    "Certificates": [
      {
        "CertificateArn": "arn:aws:iam::123456789012:server-certificate/
my-new-server-cert"
      }
    ],
    "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
    "Port": 443,
    "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/0467ef3c8400ae65"
  }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyListener](#)中的。

## modify-load-balancer-attributes

以下代码示例演示了如何使用 `modify-load-balancer-attributes`。

### AWS CLI

#### 启用删除保护

此示例为指定的负载均衡器启用删除保护。

#### 命令:

```
aws elbv2 modify-load-balancer-attributes --load-balancer-arn
arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188 --attributes Key=deletion_protection.enabled,Value=true
```

#### 输出:

```
{
  "Attributes": [
    {
      "Value": "true",
      "Key": "deletion_protection.enabled"
    },
    {
      "Value": "false",
      "Key": "access_logs.s3.enabled"
    },
    {
      "Value": "60",
      "Key": "idle_timeout.timeout_seconds"
    },
    {
      "Value": "",
      "Key": "access_logs.s3.prefix"
    },
    {
      "Value": "",
      "Key": "access_logs.s3.bucket"
    }
  ]
}
```

## 更改空闲超时

此示例更改了指定负载均衡器的空闲超时值。

命令:

```
aws elbv2 modify-load-balancer-attributes --load-balancer-arn
arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188 --attributes Key=idle_timeout.timeout_seconds,Value=30
```

输出:

```
{
  "Attributes": [
    {
      "Value": "30",
      "Key": "idle_timeout.timeout_seconds"
    }
  ]
}
```



```
    },
    {
      "Value": "false",
      "Key": "access_logs.s3.enabled"
    },
    {
      "Value": "",
      "Key": "access_logs.s3.prefix"
    },
    {
      "Value": "true",
      "Key": "deletion_protection.enabled"
    },
    {
      "Value": "",
      "Key": "access_logs.s3.bucket"
    }
  ]
}
```

## 启用访问日志

此示例启用指定负载均衡器的访问日志。请注意，S3 存储桶必须与负载均衡器位于同一区域，并且必须附加授予对 Elastic Load Balancing 服务的访问权限的策略。

### 命令:

```
aws elbv2 modify-load-balancer-attributes --load-balancer-arn
arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188 --attributes Key=access_logs.s3.enabled,Value=true
Key=access_logs.s3.bucket,Value=my-loadbalancer-logs
Key=access_logs.s3.prefix,Value=myapp
```

### 输出:

```
{
  "Attributes": [
    {
      "Value": "true",
      "Key": "access_logs.s3.enabled"
    },
    {
      "Value": "my-load-balancer-logs",
```

```

    "Key": "access_logs.s3.bucket"
  },
  {
    "Value": "myapp",
    "Key": "access_logs.s3.prefix"
  },
  {
    "Value": "60",
    "Key": "idle_timeout.timeout_seconds"
  },
  {
    "Value": "false",
    "Key": "deletion_protection.enabled"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyLoadBalancerAttributes](#)中的。

## modify-rule

以下代码示例演示了如何使用 modify-rule。

### AWS CLI

#### 修改规则

以下modify-rule示例更新了指定规则的操作和条件。

```

aws elbv2 modify-rule \
  --actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067 \
  --conditions Field=path-pattern,Values='/images/*'
  --rule-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener-rule/app/
my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2/9683b2d02a6cabee

```

输出：

```

{
  "Rules": [
    {
      "Priority": "10",

```

```
    "Conditions": [
      {
        "Field": "path-pattern",
        "Values": [
          "/images/*"
        ]
      }
    ],
    "RuleArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener-rule/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2/9683b2d02a6cabee",
    "IsDefault": false,
    "Actions": [
      {
        "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
        "Type": "forward"
      }
    ]
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyRule](#)中的。

## modify-target-group-attributes

以下代码示例演示了如何使用 `modify-target-group-attributes`。

### AWS CLI

#### 修改取消注册延迟超时

此示例将指定目标组的取消注册延迟超时设置为指定值。

#### 命令:

```
aws elbv2 modify-target-group-attributes --target-group-arn
arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067 --attributes
Key=deregistration_delay.timeout_seconds,Value=600
```

#### 输出:

```
{
  "Attributes": [
    {
      "Value": "false",
      "Key": "stickiness.enabled"
    },
    {
      "Value": "600",
      "Key": "deregistration_delay.timeout_seconds"
    },
    {
      "Value": "lb_cookie",
      "Key": "stickiness.type"
    },
    {
      "Value": "86400",
      "Key": "stickiness.lb_cookie.duration_seconds"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyTargetGroupAttributes](#)中的。

## modify-target-group

以下代码示例演示了如何使用 modify-target-group。

### AWS CLI

#### 修改目标组的运行状况检查配置

以下modify-target-group示例更改了用于评估指定目标组目标运行状况的运行状况检查的配置。请注意，由于 CLI 解析逗号的方式，您必须使用单引号而不是双引号将--matcher选项的范围括起来。

```
aws elbv2 modify-target-group \
  --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-https-targets/2453ed029918f21f \
  --health-check-protocol HTTPS \
  --health-check-port 443 \
  --matcher HttpCode='200,299'
```

输出：

```
{
  "TargetGroups": [
    {
      "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-https-targets/2453ed029918f21f",
      "TargetGroupName": "my-https-targets",
      "Protocol": "HTTPS",
      "Port": 443,
      "VpcId": "vpc-3ac0fb5f",
      "HealthCheckProtocol": "HTTPS",
      "HealthCheckPort": "443",
      "HealthCheckEnabled": true,
      "HealthCheckIntervalSeconds": 30,
      "HealthCheckTimeoutSeconds": 5,
      "HealthyThresholdCount": 5,
      "UnhealthyThresholdCount": 2,
      "Matcher": {
        "HttpCode": "200,299"
      },
      "LoadBalancerArns": [
        "arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/
app/my-load-balancer/50dc6c495c0c9188"
      ],
      "TargetType": "instance",
      "ProtocolVersion": "HTTP1",
      "IpAddressType": "ipv4"
    }
  ]
}
```

有关更多信息，请参阅《应用程序负载均衡器指南》中的[目标组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyTargetGroup](#)中的。

## register-targets

以下代码示例演示了如何使用 register-targets。

### AWS CLI

示例 1：通过实例 ID 向目标组注册目标

以下 `register-targets` 示例将指定的实例注册到目标组。目标组的目标类型必须为 `instance`。

```
aws elbv2 register-targets \  
  --target-group-arn arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067 \  
  --targets Id=i-1234567890abcdef0 Id=i-0abcdef1234567890
```

示例 2：使用端口覆盖向目标组注册目标

以下 `register-targets` 示例使用多个端口向目标组注册指定实例。这使您能够在与目标组中的目标相同的实例上注册容器。

```
aws elbv2 register-targets \  
  --target-group-arn arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-internal-targets/3bb63f11dfb0faf9 \  
  --targets Id=i-0598c7d356eba48d7,Port=80 Id=i-0598c7d356eba48d7,Port=766
```

示例 3：通过 IP 地址向目标组注册目标

以下 `register-targets` 示例将指定的 IP 地址注册到目标组。目标组的目标类型必须为 `ip`。

```
aws elbv2 register-targets \  
  --target-group-arn arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-tcp-ip-targets/8518e899d173178f \  
  --targets Id=10.0.1.15 Id=10.0.1.23
```

示例 4：将 Lambda 函数注册为目标

以下 `register-targets` 示例将指定的 IP 地址注册到目标组。目标组的目标类型必须为 `lambda`。您必须授予 Elastic Load Balancing 权限才能调用 Lambda 函数。

```
aws elbv2 register-targets \  
  --target-group-arn arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-tcp-ip-targets/8518e899d173178f \  
  --targets Id=arn:aws:lambda:us-west-2:123456789012:function:my-function
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterTargets](#) 中的。

## **remove-listener-certificates**

以下代码示例演示了如何使用 `remove-listener-certificates`。

## AWS CLI

### 从安全侦听器中删除证书

此示例从指定的安全侦听器中删除指定的证书。

命令:

```
aws elbv2 remove-listener-certificates --listener-arn
arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/
my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 --certificates
CertificateArn=arn:aws:acm:us-west-2:123456789012:certificate/5cc54884-
f4a3-4072-80be-05b9ba72f705
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveListenerCertificates](#)中的。

## remove-tags

以下代码示例演示了如何使用 remove-tags。

## AWS CLI

### 从负载均衡器中移除标签

以下remove-tags示例从指定的负载均衡器中移除project和department标签。

```
aws elbv2 remove-tags \
--resource-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \
--tag-keys project department
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveTags](#)中的。

## set-ip-address-type

以下代码示例演示了如何使用 set-ip-address-type。

## AWS CLI

### 设置负载均衡器的地址类型

此示例将指定负载均衡器的地址类型设置为dualstack。负载均衡器子网必须具有关联的 IPv6 CIDR 块。

**命令:**

```
aws elbv2 set-ip-address-type --load-balancer-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 --ip-address-type dualstack
```

**输出:**

```
{
  "IpAddressType": "dualstack"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetIpAddressType](#)中的。

**set-rule-priorities**

以下代码示例演示了如何使用 set-rule-priorities。

**AWS CLI****设置规则优先级**

此示例设置指定规则的优先级。

**命令:**

```
aws elbv2 set-rule-priorities --rule-priorities
RuleArn=arn:aws:elasticloadbalancing:us-west-2:123456789012:listener-rule/app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2/1291d13826f405c3,Priority=5
```

**输出:**

```
{
  "Rules": [
    {
      "Priority": "5",
      "Conditions": [
        {
          "Field": "path-pattern",
          "Values": [
            "/img/*"
          ]
        }
      ]
    }
  ]
}
```



```

    }
  ],
  "RuleArn": "arn:aws:elasticloadbalancing:us-west-2:123456789012:listener-
rule/app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2/1291d13826f405c3",
  "IsDefault": false,
  "Actions": [
    {
      "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
      "Type": "forward"
    }
  ]
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetRulePriorities](#)中的。

## set-security-groups

以下代码示例演示了如何使用 set-security-groups。

### AWS CLI

将安全组与负载均衡器关联

此示例将指定的安全组与指定的负载均衡器相关联。

命令:

```
aws elbv2 set-security-groups --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 --security-
groups sg-5943793c
```

输出：

```
{
  "SecurityGroupIds": [
    "sg-5943793c"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetSecurityGroups](#)中的。

## set-subnets

以下代码示例演示了如何使用 set-subnets。

### AWS CLI

为负载均衡器启用可用区

此示例为指定负载均衡器的指定子网启用可用区。

命令:

```
aws elbv2 set-subnets --load-balancer-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 --subnets subnet-8360a9e7 subnet-b7d581c0
```

输出:

```
{
  "AvailabilityZones": [
    {
      "SubnetId": "subnet-8360a9e7",
      "ZoneName": "us-west-2a"
    },
    {
      "SubnetId": "subnet-b7d581c0",
      "ZoneName": "us-west-2b"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetSubnets](#)中的。

## 使用 Elastic Transcoder 示例 AWS CLI

以下代码示例向您展示了如何使用与 Elastic Transcoder AWS Command Line Interface 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## cancel-job

以下代码示例演示了如何使用 `cancel-job`。

AWS CLI

要取消某项任务 ElasticTranscoder

这将取消的指定任务。 ElasticTranscoder

命令:

```
aws elastictranscoder cancel-job --id 3333333333333-abcde3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelJob](#)中的。

## create-job

以下代码示例演示了如何使用 `create-job`。

AWS CLI

为以下对象创建工作 ElasticTranscoder

以下`create-job`示例为创建了作业 ElasticTranscoder。

```
aws elastictranscoder create-job \  
  --pipeline-id 1111111111111-abcde1 \  
  --inputs file://inputs.json \  
  --outputs file://outputs.json \  
  --output-key-prefix "recipes/" \  
  --user-metadata file://user-metadata.json
```

inputs.json 的内容：

```
[{
  "Key": "ETS_example_file.mp4",
  "FrameRate": "auto",
  "Resolution": "auto",
  "AspectRatio": "auto",
  "Interlaced": "auto",
  "Container": "mp4"
}]
```

outputs.json 的内容：

```
[
  {
    "Key": "webm/ETS_example_file-kindlefirehd.webm",
    "Rotate": "0",
    "PresetId": "1351620000001-100250"
  }
]
```

user-metadata.json 的内容：

```
{
  "Food type": "Italian",
  "Cook book": "recipe notebook"
}
```

输出：

```
{
  "Job": {
    "Status": "Submitted",
    "Inputs": [
      {
        "Container": "mp4",
        "FrameRate": "auto",
        "Key": "ETS_example_file.mp4",
        "AspectRatio": "auto",
        "Resolution": "auto",
        "Interlaced": "auto"
      }
    ]
  }
}
```

```
],
  "Playlists": [],
  "Outputs": [
    {
      "Status": "Submitted",
      "Rotate": "0",
      "PresetId": "1351620000001-100250",
      "Watermarks": [],
      "Key": "webm/ETS_example_file-kindlefirehd.webm",
      "Id": "1"
    }
  ],
  "PipelineId": "3333333333333-abcde3",
  "OutputKeyPrefix": "recipes/",
  "UserMetadata": {
    "Cook book": "recipe notebook",
    "Food type": "Italian"
  },
  "Output": {
    "Status": "Submitted",
    "Rotate": "0",
    "PresetId": "1351620000001-100250",
    "Watermarks": [],
    "Key": "webm/ETS_example_file-kindlefirehd.webm",
    "Id": "1"
  },
  "Timing": {
    "SubmitTimeMillis": 1533838012298
  },
  "Input": {
    "Container": "mp4",
    "FrameRate": "auto",
    "Key": "ETS_example_file.mp4",
    "AspectRatio": "auto",
    "Resolution": "auto",
    "Interlaced": "auto"
  },
  "Id": "1533838012294-example",
  "Arn": "arn:aws:elastictranscoder:us-west-2:123456789012:job/1533838012294-example"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateJob](#)中的。

## create-pipeline

以下代码示例演示了如何使用 create-pipeline。

### AWS CLI

为创建管道 ElasticTranscoder

以下create-pipeline示例为创建了一个管道 ElasticTranscoder。

```
aws elastictranscoder create-pipeline \  
  --name Default \  
  --input-bucket salesoffice.example.com-source \  
  --role arn:aws:iam::123456789012:role/Elastic_Transcoder_Default_Role \  
  --notifications Progressing="",Completed="",Warning="",Error=arn:aws:sns:us-  
east-1:111222333444:ETS_Errors \  
  --content-config file://content-config.json \  
  --thumbnail-config file://thumbnail-config.json
```

content-config.json 的内容：

```
{  
  "Bucket":"salesoffice.example.com-public-promos",  
  "Permissions":[  
    {  
      "GranteeType":"Email",  
      "Grantee":"marketing-promos@example.com",  
      "Access":[  
        "FullControl"  
      ]  
    }  
  ],  
  "StorageClass":"Standard"  
}
```

thumbnail-config.json 的内容：

```
{  
  "Bucket":"salesoffice.example.com-public-promos-thumbnails",  
  "Permissions":[  
    {  
      "GranteeType":"Email",
```

```
        "Grantee": "marketing-promos@example.com",
        "Access": [
            "FullControl"
        ]
    },
],
"StorageClass": "ReducedRedundancy"
}
```

输出：

```
{
  "Pipeline": {
    "Status": "Active",
    "ContentConfig": {
      "Bucket": "salesoffice.example.com-public-promos",
      "StorageClass": "Standard",
      "Permissions": [
        {
          "Access": [
            "FullControl"
          ],
          "Grantee": "marketing-promos@example.com",
          "GranteeType": "Email"
        }
      ]
    },
    "Name": "Default",
    "ThumbnailConfig": {
      "Bucket": "salesoffice.example.com-public-promos-thumbnails",
      "StorageClass": "ReducedRedundancy",
      "Permissions": [
        {
          "Access": [
            "FullControl"
          ],
          "Grantee": "marketing-promos@example.com",
          "GranteeType": "Email"
        }
      ]
    },
    "Notifications": {
      "Completed": "",

```

```

        "Warning": "",
        "Progressing": "",
        "Error": "arn:aws:sns:us-east-1:123456789012:ETS_Errors"
    },
    "Role": "arn:aws:iam::123456789012:role/Elastic_Transcoder_Default_Role",
    "InputBucket": "salesoffice.example.com-source",
    "Id": "1533765810590-example",
    "Arn": "arn:aws:elastictranscoder:us-
west-2:123456789012:pipeline/1533765810590-example"
    },
    "Warnings": [
        {
            "Message": "The SNS notification topic for Error events and the pipeline
are in different regions, which increases processing time for jobs in the pipeline
and can incur additional charges. To decrease processing time and prevent cross-
regional charges, use the same region for the SNS notification topic and the
pipeline.",
            "Code": "6006"
        }
    ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePipeline](#)中的。

## create-preset

以下代码示例演示了如何使用 create-preset。

### AWS CLI

为创建预设 ElasticTranscoder

以下create-preset示例为创建预设 ElasticTranscoder。

```

aws elastictranscoder create-preset \
  --name DefaultPreset \
  --description "Use for published videos" \
  --container mp4 \
  --video file://video.json \
  --audio file://audio.json \
  --thumbnails file://thumbnails.json

```

video.json 的内容：



```
{
  "Codec": "H.264",
  "CodecOptions": {
    "Profile": "main",
    "Level": "2.2",
    "MaxReferenceFrames": "3",
    "MaxBitRate": "",
    "BufferSize": "",
    "InterlacedMode": "Progressive",
    "ColorSpaceConversionMode": "None"
  },
  "KeyframesMaxDist": "240",
  "FixedGOP": "false",
  "BitRate": "1600",
  "FrameRate": "auto",
  "MaxFrameRate": "30",
  "MaxWidth": "auto",
  "MaxHeight": "auto",
  "SizingPolicy": "Fit",
  "PaddingPolicy": "Pad",
  "DisplayAspectRatio": "auto",
  "Watermarks": [
    {
      "Id": "company logo",
      "MaxWidth": "20%",
      "MaxHeight": "20%",
      "SizingPolicy": "ShrinkToFit",
      "HorizontalAlign": "Right",
      "HorizontalOffset": "10px",
      "VerticalAlign": "Bottom",
      "VerticalOffset": "10px",
      "Opacity": "55.5",
      "Target": "Content"
    }
  ]
}
```

audio.json 的内容：

```
{
  "Codec": "AAC",
  "CodecOptions": {
    "Profile": "AAC-LC"
  }
}
```

```
  },  
  "SampleRate": "44100",  
  "BitRate": "96",  
  "Channels": "2"  
}
```

thumbnails.json 的内容：

```
{  
  "Format": "png",  
  "Interval": "120",  
  "MaxWidth": "auto",  
  "MaxHeight": "auto",  
  "SizingPolicy": "Fit",  
  "PaddingPolicy": "Pad"  
}
```

输出：

```
{  
  "Preset": {  
    "Thumbnails": {  
      "SizingPolicy": "Fit",  
      "MaxWidth": "auto",  
      "Format": "png",  
      "PaddingPolicy": "Pad",  
      "Interval": "120",  
      "MaxHeight": "auto"  
    },  
    "Container": "mp4",  
    "Description": "Use for published videos",  
    "Video": {  
      "SizingPolicy": "Fit",  
      "MaxWidth": "auto",  
      "PaddingPolicy": "Pad",  
      "MaxFrameRate": "30",  
      "FrameRate": "auto",  
      "MaxHeight": "auto",  
      "KeyframesMaxDist": "240",  
      "FixedGOP": "false",  
      "Codec": "H.264",  
      "Watermarks": [  
        {  
          "SizingPolicy": "Fit",  
          "MaxWidth": "auto",  
          "MaxHeight": "auto",  
          "KeyframesMaxDist": "240",  
          "FixedGOP": "false",  
          "Codec": "H.264",  
          "Watermarks": [  
            {  
              "SizingPolicy": "Fit",  
              "MaxWidth": "auto",  
              "MaxHeight": "auto",  
              "KeyframesMaxDist": "240",  
              "FixedGOP": "false",  
              "Codec": "H.264",  
              "Watermarks": [  
                {  
                  "SizingPolicy": "Fit",  
                  "MaxWidth": "auto",  
                  "MaxHeight": "auto",  
                  "KeyframesMaxDist": "240",  
                  "FixedGOP": "false",  
                  "Codec": "H.264",  
                  "Watermarks": [  
                    {  
                      "SizingPolicy": "Fit",  
                      "MaxWidth": "auto",  
                      "MaxHeight": "auto",  
                      "KeyframesMaxDist": "240",  
                      "FixedGOP": "false",  
                      "Codec": "H.264",  
                      "Watermarks": [  
                        {  
                          "SizingPolicy": "Fit",  
                          "MaxWidth": "auto",  
                          "MaxHeight": "auto",  
                          "KeyframesMaxDist": "240",  
                          "FixedGOP": "false",  
                          "Codec": "H.264",  
                          "Watermarks": [  
                            {  
                              "SizingPolicy": "Fit",  
                              "MaxWidth": "auto",  
                              "MaxHeight": "auto",  
                              "KeyframesMaxDist": "240",  
                              "FixedGOP": "false",  
                              "Codec": "H.264",  
                              "Watermarks": [  
                                {  
                                  "SizingPolicy": "Fit",  
                                  "MaxWidth": "auto",  
                                  "MaxHeight": "auto",  
                                  "KeyframesMaxDist": "240",  
                                  "FixedGOP": "false",  
                                  "Codec": "H.264",  
                                  "Watermarks": [  
                                    {  
                                      "SizingPolicy": "Fit",  
                                      "MaxWidth": "auto",  
                                      "MaxHeight": "auto",  
                                      "KeyframesMaxDist": "240",  
                                      "FixedGOP": "false",  
                                      "Codec": "H.264",  
                                      "Watermarks": [  
                                        {  
                                          "SizingPolicy": "Fit",  
                                          "MaxWidth": "auto",  
                                          "MaxHeight": "auto",  
                                          "KeyframesMaxDist": "240",  
                                          "FixedGOP": "false",  
                                          "Codec": "H.264",  
                                          "Watermarks": [  
                                            {  
                                              "SizingPolicy": "Fit",  
                                              "MaxWidth": "auto",  
                                              "MaxHeight": "auto",  
                                              "KeyframesMaxDist": "240",  
                                              "FixedGOP": "false",  
                                              "Codec": "H.264",  
                                              "Watermarks": [  
                                                {  
                                                  "SizingPolicy": "Fit",  
                                                  "MaxWidth": "auto",  
                                                  "MaxHeight": "auto",  
                                                  "KeyframesMaxDist": "240",  
                                                  "FixedGOP": "false",  
                                                  "Codec": "H.264",  
                                                  "Watermarks": [  
                                                    {  
                                                      "SizingPolicy": "Fit",  
                                                      "MaxWidth": "auto",  
                                                      "MaxHeight": "auto",  
                                                      "KeyframesMaxDist": "240",  
                                                      "FixedGOP": "false",  
                                                      "Codec": "H.264",  
                                                      "Watermarks": [  
                                                        {  
                                                          "SizingPolicy": "Fit",  
                                                          "MaxWidth": "auto",  
                                                          "MaxHeight": "auto",  
                                                          "KeyframesMaxDist": "240",  
                                                          "FixedGOP": "false",  
                                                          "Codec": "H.264",  
                                                          "Watermarks": [  
                                                            {  
                                                              "SizingPolicy": "Fit",  
                                                              "MaxWidth": "auto",  
                                                              "MaxHeight": "auto",  
                                                              "KeyframesMaxDist": "240",  
                                                              "FixedGOP": "false",  
                                                              "Codec": "H.264",  
                                                              "Watermarks": [  
                                                                {  
                                                                  "SizingPolicy": "Fit",  
                                                                  "MaxWidth": "auto",  
                                                                  "MaxHeight": "auto",  
                                                                  "KeyframesMaxDist": "240",  
                                                                  "FixedGOP": "false",  
                                                                  "Codec": "H.264",  
                                                                  "Watermarks": [  
                                                                    {  
                                                                      "SizingPolicy": "Fit",  
                                                                      "MaxWidth": "auto",  
                                                                      "MaxHeight": "auto",  
                                                                      "KeyframesMaxDist": "240",  
                                                                      "FixedGOP": "false",  
                                                                      "Codec": "H.264",  
                                                                      "Watermarks": [  
                                                                      ]  
                                                                    ]  
                                                                  ]  
                                                                ]  
                                                              ]  
                                                            ]  
                                                          ]  
                                                        ]  
                                                      ]  
                                                    ]  
                                                  ]  
                                                ]  
                                              ]  
                                            ]  
                                          ]  
                                        ]  
                                      ]  
                                    ]  
                                  ]  
                                ]  
                              ]  
                            ]  
                          ]  
                        ]  
                      ]  
                    ]  
                  ]  
                ]  
              ]  
            ]  
          ]  
        ]  
      ]  
    ]  
  ]  
}
```

```

        "SizingPolicy": "ShrinkToFit",
        "VerticalOffset": "10px",
        "VerticalAlign": "Bottom",
        "Target": "Content",
        "MaxWidth": "20%",
        "MaxHeight": "20%",
        "HorizontalAlign": "Right",
        "HorizontalOffset": "10px",
        "Opacity": "55.5",
        "Id": "company logo"
    }
],
"CodecOptions": {
    "Profile": "main",
    "MaxBitRate": "32",
    "InterlacedMode": "Progressive",
    "Level": "2.2",
    "ColorSpaceConversionMode": "None",
    "MaxReferenceFrames": "3",
    "BufferSize": "5"
},
"BitRate": "1600",
"DisplayAspectRatio": "auto"
},
"Audio": {
    "Channels": "2",
    "CodecOptions": {
        "Profile": "AAC-LC"
    },
    "SampleRate": "44100",
    "Codec": "AAC",
    "BitRate": "96"
},
"Type": "Custom",
"Id": "1533765290724-example"
"Arn": "arn:aws:elastictranscoder:us-
west-2:123456789012:preset/1533765290724-example",
"Name": "DefaultPreset"
},
"Warning": ""
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePreset](#)中的。

## delete-pipeline

以下代码示例演示了如何使用 delete-pipeline。

### AWS CLI

删除指定的 ElasticTranscoder 管道

这将删除指定的 ElasticTranscoder 管道。

命令:

```
aws elastictranscoder delete-pipeline --id 111111111111-abcde1
```

输出:

```
{
  "Success": "true"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePipeline](#)中的。

## delete-preset

以下代码示例演示了如何使用 delete-preset。

### AWS CLI

删除指定的 ElasticTranscoder 预设

这将删除指定的 ElasticTranscoder 预设。

命令:

```
aws elastictranscoder delete-preset --id 555555555555-abcde5
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePreset](#)中的。

## list-jobs-by-pipeline

以下代码示例演示了如何使用 list-jobs-by-pipeline。

## AWS CLI

检索指定管道中的 ElasticTranscoder 任务列表

此示例检索指定管道中的 ElasticTranscoder 任务列表。

命令:

```
aws elastictranscoder list-jobs-by-pipeline --pipeline-id 11111111111111-abcde1
```

输出:

```
{
  "Jobs": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListJobsByPipeline](#)中的。

## list-jobs-by-status

以下代码示例演示了如何使用 list-jobs-by-status。

### AWS CLI

检索状态为“完成”的 ElasticTranscoder 任务列表

此示例检索状态为“完成”的 ElasticTranscoder 任务列表。

命令:

```
aws elastictranscoder list-jobs-by-status --status Complete
```

输出:

```
{
  "Jobs": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListJobsByStatus](#)中的。

## list-pipelines

以下代码示例演示了如何使用 `list-pipelines`。

### AWS CLI

检索 ElasticTranscoder 管道列表

此示例检索 ElasticTranscoder 管道列表。

命令:

```
aws elastictranscoder list-pipelines
```

输出:

```
{
  "Pipelines": [
    {
      "Status": "Active",
      "ContentConfig": {
        "Bucket": "ets-example",
        "Permissions": []
      },
      "Name": "example-pipeline",
      "ThumbnailConfig": {
        "Bucket": "ets-example",
        "Permissions": []
      },
      "Notifications": {
        "Completed": "arn:aws:sns:us-west-2:123456789012:ets_example",
        "Warning": "",
        "Progressing": "",
        "Error": ""
      },
      "Role": "arn:aws:iam::123456789012:role/Elastic_Transcoder_Default_Role",
      "InputBucket": "ets-example",
      "OutputBucket": "ets-example",
      "Id": "3333333333333-abcde3",
      "Arn": "arn:aws:elastictranscoder:us-west-2:123456789012:pipeline/3333333333333-abcde3"
    },
    {
      "Status": "Paused",
```

```
    "ContentConfig": {
      "Bucket": "ets-example",
      "Permissions": []
    },
    "Name": "example-php-test",
    "ThumbnailConfig": {
      "Bucket": "ets-example",
      "Permissions": []
    },
    "Notifications": {
      "Completed": "",
      "Warning": "",
      "Progressing": "",
      "Error": ""
    },
    "Role": "arn:aws:iam::123456789012:role/Elastic_Transcoder_Default_Role",
    "InputBucket": "ets-example",
    "OutputBucket": "ets-example",
    "Id": "333333333333-abcde2",
    "Arn": "arn:aws:elastictranscoder:us-
west-2:123456789012:pipeline/333333333333-abcde2"
  },
  {
    "Status": "Active",
    "ContentConfig": {
      "Bucket": "ets-west-output",
      "Permissions": []
    },
    "Name": "pipeline-west",
    "ThumbnailConfig": {
      "Bucket": "ets-west-output",
      "Permissions": []
    },
    "Notifications": {
      "Completed": "arn:aws:sns:us-west-2:123456789012:ets-notifications",
      "Warning": "",
      "Progressing": "",
      "Error": ""
    },
    "Role": "arn:aws:iam::123456789012:role/Elastic_Transcoder_Default_Role",
    "InputBucket": "ets-west-input",
    "OutputBucket": "ets-west-output",
    "Id": "333333333333-abcde1",
```

```
        "Arn": "arn:aws:elastictranscoder:us-  
west-2:123456789012:pipeline/333333333333-abcde1"  
      }  
    ]  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPipelines](#)中的。

## list-presets

以下代码示例演示了如何使用 list-presets。

### AWS CLI

检索 ElasticTranscoder 预设列表

此示例检索 ElasticTranscoder 预设列表。

命令:

```
aws elastictranscoder list-presets --max-items 2
```

输出:

```
{  
  "Presets": [  
    {  
      "Container": "mp4",  
      "Name": "KindleFireHD-preset",  
      "Video": {  
        "Resolution": "1280x720",  
        "FrameRate": "30",  
        "KeyframesMaxDist": "90",  
        "FixedGOP": "false",  
        "Codec": "H.264",  
        "Watermarks": [],  
        "CodecOptions": {  
          "Profile": "main",  
          "MaxReferenceFrames": "3",  
          "ColorSpaceConversionMode": "None",  
          "InterlacedMode": "Progressive",  
          "Level": "4"  
        }  
      }  
    },  
  ],  
}
```



```
        "AspectRatio": "16:9",
        "BitRate": "2200"
    },
    "Audio": {
        "Channels": "2",
        "CodecOptions": {
            "Profile": "AAC-LC"
        },
        "SampleRate": "48000",
        "Codec": "AAC",
        "BitRate": "160"
    },
    "Type": "Custom",
    "Id": "333333333333-abcde2",
    "Arn": "arn:aws:elastictranscoder:us-
west-2:123456789012:preset/333333333333-abcde2",
    "Thumbnails": {
        "AspectRatio": "16:9",
        "Interval": "60",
        "Resolution": "192x108",
        "Format": "png"
    }
},
{
    "Thumbnails": {
        "AspectRatio": "16:9",
        "Interval": "60",
        "Resolution": "192x108",
        "Format": "png"
    },
    "Container": "mp4",
    "Description": "Custom preset for transcoding jobs",
    "Video": {
        "Resolution": "1280x720",
        "FrameRate": "30",
        "KeyframesMaxDist": "90",
        "FixedGOP": "false",
        "Codec": "H.264",
        "Watermarks": [],
        "CodecOptions": {
            "Profile": "main",
            "MaxReferenceFrames": "3",
            "ColorSpaceConversionMode": "None",
            "InterlacedMode": "Progressive",
```

```

        "Level": "3.1"
      },
      "AspectRatio": "16:9",
      "BitRate": "2200"
    },
    "Audio": {
      "Channels": "2",
      "CodecOptions": {
        "Profile": "AAC-LC"
      },
      "SampleRate": "44100",
      "Codec": "AAC",
      "BitRate": "160"
    },
    "Type": "Custom",
    "Id": "333333333333-abcde3",
    "Arn": "arn:aws:elastictranscoder:us-
west-2:123456789012:preset/333333333333-abcde3",
    "Name": "Roman's Preset"
  }
],
"NextToken": "eyJQYWdlVG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ=="
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPresets](#)中的。

## read-job

以下代码示例演示了如何使用 read-job。

### AWS CLI

检索作 ElasticTranscoder 业

此示例检索指定的作 ElasticTranscoder 业。

命令:

```
aws elastictranscoder read-job --id 1533838012294-example
```

输出:

```
{
```

```
"Job": {
  "Status": "Progressing",
  "Inputs": [
    {
      "Container": "mp4",
      "FrameRate": "auto",
      "Key": "ETS_example_file.mp4",
      "AspectRatio": "auto",
      "Resolution": "auto",
      "Interlaced": "auto"
    }
  ],
  "Playlists": [],
  "Outputs": [
    {
      "Status": "Progressing",
      "Rotate": "0",
      "PresetId": "1351620000001-100250",
      "Watermarks": [],
      "Key": "webm/ETS_example_file-kindlefirehd.webm",
      "Id": "1"
    }
  ],
  "PipelineId": "333333333333-abcde3",
  "OutputKeyPrefix": "recipes/",
  "UserMetadata": {
    "Cook book": "recipe notebook",
    "Food type": "Italian"
  },
  "Output": {
    "Status": "Progressing",
    "Rotate": "0",
    "PresetId": "1351620000001-100250",
    "Watermarks": [],
    "Key": "webm/ETS_example_file-kindlefirehd.webm",
    "Id": "1"
  },
  "Timing": {
    "SubmitTimeMillis": 1533838012298,
    "StartTimeMillis": 1533838013786
  },
  "Input": {
    "Container": "mp4",
    "FrameRate": "auto",
```

```
        "Key": "ETS_example_file.mp4",
        "AspectRatio": "auto",
        "Resolution": "auto",
        "Interlaced": "auto"
    },
    "Id": "1533838012294-example",
    "Arn": "arn:aws:elastictranscoder:us-west-2:123456789012:job/1533838012294-example"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReadJob](#)中的。

## read-pipeline

以下代码示例演示了如何使用 read-pipeline。

### AWS CLI

检索 ElasticTranscoder 管道

此示例检索指定的 ElasticTranscoder 管道。

命令:

```
aws elastictranscoder read-pipeline --id 333333333333-abcde3
```

输出:

```
{
  "Pipeline": {
    "Status": "Active",
    "ContentConfig": {
      "Bucket": "ets-example",
      "StorageClass": "Standard",
      "Permissions": [
        {
          "Access": [
            "FullControl"
          ],
          "Grantee": "marketing-promos@example.com",
        }
      ]
    }
  }
}
```

```

        "GranteeType": "Email"
      }
    ]
  },
  "Name": "Default",
  "ThumbnailConfig": {
    "Bucket": "ets-example",
    "StorageClass": "ReducedRedundancy",
    "Permissions": [
      {
        "Access": [
          "FullControl"
        ],
        "Grantee": "marketing-promos@example.com",
        "GranteeType": "Email"
      }
    ]
  },
  "Notifications": {
    "Completed": "",
    "Warning": "",
    "Progressing": "",
    "Error": "arn:aws:sns:us-east-1:123456789012:ETS_Errors"
  },
  "Role": "arn:aws:iam::123456789012:role/Elastic_Transcoder_Default_Role",
  "InputBucket": "ets-example",
  "Id": "333333333333-abcde3",
  "Arn": "arn:aws:elastictranscoder:us-
west-2:123456789012:pipeline/333333333333-abcde3"
},
"Warnings": [
  {
    "Message": "The SNS notification topic for Error events and the pipeline
are in different regions, which increases processing time for jobs in the pipeline
and can incur additional charges. To decrease processing time and prevent cross-
regional charges, use the same region for the SNS notification topic and the
pipeline.",
    "Code": "6006"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReadPipeline](#)中的。

## read-preset

以下代码示例演示了如何使用 read-preset。

### AWS CLI

检索预 ElasticTranscoder 设

此示例检索指定的 ElasticTranscoder 预设。

命令:

```
aws elastictranscoder read-preset --id 1351620000001-500020
```

输出:

```
{
  "Preset": {
    "Thumbnails": {
      "SizingPolicy": "ShrinkToFit",
      "MaxWidth": "192",
      "Format": "png",
      "PaddingPolicy": "NoPad",
      "Interval": "300",
      "MaxHeight": "108"
    },
    "Container": "fmp4",
    "Description": "System preset: MPEG-Dash Video - 4.8M",
    "Video": {
      "SizingPolicy": "ShrinkToFit",
      "MaxWidth": "1280",
      "PaddingPolicy": "NoPad",
      "FrameRate": "30",
      "MaxHeight": "720",
      "KeyframesMaxDist": "60",
      "FixedGOP": "true",
      "Codec": "H.264",
      "Watermarks": [
        {
          "SizingPolicy": "ShrinkToFit",
          "VerticalOffset": "10%",
          "VerticalAlign": "Top",
          "Target": "Content",
          "MaxWidth": "10%",
```

```
        "MaxHeight": "10%",
        "HorizontalAlign": "Left",
        "HorizontalOffset": "10%",
        "Opacity": "100",
        "Id": "TopLeft"
    },
    {
        "SizingPolicy": "ShrinkToFit",
        "VerticalOffset": "10%",
        "VerticalAlign": "Top",
        "Target": "Content",
        "MaxWidth": "10%",
        "MaxHeight": "10%",
        "HorizontalAlign": "Right",
        "HorizontalOffset": "10%",
        "Opacity": "100",
        "Id": "TopRight"
    },
    {
        "SizingPolicy": "ShrinkToFit",
        "VerticalOffset": "10%",
        "VerticalAlign": "Bottom",
        "Target": "Content",
        "MaxWidth": "10%",
        "MaxHeight": "10%",
        "HorizontalAlign": "Left",
        "HorizontalOffset": "10%",
        "Opacity": "100",
        "Id": "BottomLeft"
    },
    {
        "SizingPolicy": "ShrinkToFit",
        "VerticalOffset": "10%",
        "VerticalAlign": "Bottom",
        "Target": "Content",
        "MaxWidth": "10%",
        "MaxHeight": "10%",
        "HorizontalAlign": "Right",
        "HorizontalOffset": "10%",
        "Opacity": "100",
        "Id": "BottomRight"
    }
],
"CodecOptions": {
```

```

        "Profile": "main",
        "MaxBitRate": "4800",
        "InterlacedMode": "Progressive",
        "Level": "3.1",
        "ColorSpaceConversionMode": "None",
        "MaxReferenceFrames": "3",
        "BufferSize": "9600"
    },
    "BitRate": "4800",
    "DisplayAspectRatio": "auto"
},
"Type": "System",
"Id": "1351620000001-500020",
"Arn": "arn:aws:elastictranscoder:us-
west-2:123456789012:preset/1351620000001-500020",
"Name": "System preset: MPEG-Dash Video - 4.8M"
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReadPreset](#)中的。

## update-pipeline-notifications

以下代码示例演示了如何使用 update-pipeline-notifications。

### AWS CLI

#### 更新 ElasticTranscoder 管道的通知

此示例更新指定 ElasticTranscoder 管道的通知。

#### 命令:

```

aws elastictranscoder update-pipeline-notifications --id 111111111111-
abcde1 --notifications Progressing=arn:aws:sns:us-west-2:0123456789012:my-
topic,Completed=arn:aws:sns:us-west-2:0123456789012:my-topic,Warning=arn:aws:sns:us-
west-2:0123456789012:my-topic,Error=arn:aws:sns:us-east-1:111222333444:ETS_Errors

```

#### 输出:

```
{
```



```
"Pipeline": {
  "Status": "Active",
  "ContentConfig": {
    "Bucket": "ets-example",
    "StorageClass": "Standard",
    "Permissions": [
      {
        "Access": [
          "FullControl"
        ],
        "Grantee": "marketing-promos@example.com",
        "GranteeType": "Email"
      }
    ]
  },
  "Name": "Default",
  "ThumbnailConfig": {
    "Bucket": "ets-example",
    "StorageClass": "ReducedRedundancy",
    "Permissions": [
      {
        "Access": [
          "FullControl"
        ],
        "Grantee": "marketing-promos@example.com",
        "GranteeType": "Email"
      }
    ]
  },
  "Notifications": {
    "Completed": "arn:aws:sns:us-west-2:0123456789012:my-topic",
    "Warning": "arn:aws:sns:us-west-2:0123456789012:my-topic",
    "Progressing": "arn:aws:sns:us-west-2:0123456789012:my-topic",
    "Error": "arn:aws:sns:us-east-1:111222333444:ETS_Errors"
  },
  "Role": "arn:aws:iam::123456789012:role/Elastic_Transcoder_Default_Role",
  "InputBucket": "ets-example",
  "Id": "111111111111-abcde1",
  "Arn": "arn:aws:elastictranscoder:us-
west-2:123456789012:pipeline/111111111111-abcde1"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePipelineNotifications](#)中的。

## update-pipeline-status

以下代码示例演示了如何使用 update-pipeline-status。

### AWS CLI

更新 ElasticTranscoder 管道的状态

此示例更新了指定 ElasticTranscoder 管道的状态。

命令:

```
aws elastictranscoder update-pipeline-status --id 111111111111-abcde1 --status Paused
```

输出:

```
{
  "Pipeline": {
    "Status": "Paused",
    "ContentConfig": {
      "Bucket": "ets-example",
      "StorageClass": "Standard",
      "Permissions": [
        {
          "Access": [
            "FullControl"
          ],
          "Grantee": "marketing-promos@example.com",
          "GranteeType": "Email"
        }
      ]
    },
    "Name": "Default",
    "ThumbnailConfig": {
      "Bucket": "ets-example",
      "StorageClass": "ReducedRedundancy",
      "Permissions": [
        {
          "Access": [
            "FullControl"
          ],
          "Grantee": "marketing-promos@example.com",
          "GranteeType": "Email"
        }
      ]
    }
  }
}
```

```

    }
  ]
},
"Notifications": {
  "Completed": "",
  "Warning": "",
  "Progressing": "",
  "Error": "arn:aws:sns:us-east-1:803981987763:ETS_Errors"
},
"Role": "arn:aws:iam::123456789012:role/Elastic_Transcoder_Default_Role",
"InputBucket": "ets-example",
"Id": "111111111111-abcde1",
"Arn": "arn:aws:elastictranscoder:us-
west-2:123456789012:pipeline/111111111111-abcde1"
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePipelineStatus](#)中的。

## update-pipeline

以下代码示例演示了如何使用 update-pipeline。

### AWS CLI

#### 更新 ElasticTranscoder 管道

以下update-pipeline示例更新了指定的 ElasticTranscoder 管道。

```

aws elastictranscoder update-pipeline \
  --id 111111111111-abcde1
  --name DefaultExample \
  --input-bucket salesoffice.example.com-source \
  --role arn:aws:iam::123456789012:role/Elastic_Transcoder_Default_Role \
  --notifications Progressing="",Completed="",Warning="",Error=arn:aws:sns:us-
east-1:111222333444:ETS_Errors \
  --content-config file://content-config.json \
  --thumbnail-config file://thumbnail-config.json

```

content-config.json 的内容：

```
{
```

```
"Bucket": "salesoffice.example.com-public-promos",
"Permissions": [
  {
    "GranteeType": "Email",
    "Grantee": "marketing-promos@example.com",
    "Access": [
      "FullControl"
    ]
  }
],
"StorageClass": "Standard"
}
```

thumbnail-config.json 的内容：

```
{
  "Bucket": "salesoffice.example.com-public-promos-thumbnails",
  "Permissions": [
    {
      "GranteeType": "Email",
      "Grantee": "marketing-promos@example.com",
      "Access": [
        "FullControl"
      ]
    }
  ],
  "StorageClass": "ReducedRedundancy"
}
```

输出：

```
{
  "Pipeline": {
    "Status": "Active",
    "ContentConfig": {
      "Bucket": "ets-example",
      "StorageClass": "Standard",
      "Permissions": [
        {
          "Access": [
            "FullControl"
          ],
          "Grantee": "marketing-promos@example.com",

```

```

        "GranteeType": "Email"
      }
    ]
  },
  "Name": "DefaultExample",
  "ThumbnailConfig": {
    "Bucket": "ets-example",
    "StorageClass": "ReducedRedundancy",
    "Permissions": [
      {
        "Access": [
          "FullControl"
        ],
        "Grantee": "marketing-promos@example.com",
        "GranteeType": "Email"
      }
    ]
  },
  "Notifications": {
    "Completed": "",
    "Warning": "",
    "Progressing": "",
    "Error": "arn:aws:sns:us-east-1:111222333444:ETS_Errors"
  },
  "Role": "arn:aws:iam::123456789012:role/Elastic_Transcoder_Default_Role",
  "InputBucket": "ets-example",
  "Id": "333333333333-abcde3",
  "Arn": "arn:aws:elastictranscoder:us-
west-2:123456789012:pipeline/333333333333-abcde3"
},
"Warnings": [
  {
    "Message": "The SNS notification topic for Error events and the pipeline
are in different regions, which increases processing time for jobs in the pipeline
and can incur additional charges. To decrease processing time and prevent cross-
regional charges, use the same region for the SNS notification topic and the
pipeline.",
    "Code": "6006"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePipeline](#)中的。

## ElastiCache 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 ElastiCache。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### add-tags-to-resource

以下代码示例演示了如何使用 add-tags-to-resource。

AWS CLI

为资源添加标签

以下add-tags-to-resource示例向集群或快照资源添加最多 10 个标签（键值对）。

```
aws elasticache add-tags-to-resource \  
  --resource-name "arn:aws:elasticache:us-east-1:1234567890:cluster:my-mem-  
cluster" \  
  --tags '{"20150202":15, "ElastiCache":"Service"}
```

输出：

```
{  
  "TagList": [  
    {  
      "Value": "20150202",  
      "Key": "APIVersion"  
    },  
    {
```

```
        "Value": "ElastiCache",
        "Key": "Service"
    }
]
}
```

有关更多信息，请参阅 Elasticache 用户指南中的[使用成本分配标签监控成本](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddTagsToResource](#)中的。

## authorize-cache-security-group-ingress

以下代码示例演示了如何使用 authorize-cache-security-group-ingress。

### AWS CLI

为入口授权缓存安全组

以下authorize-cache-security-group-ingress示例允许网络进入缓存安全组。

```
aws elasticache authorize-cache-security-group-ingress \
  --cache-security-group-name "my-sec-grp" \
  --ec2-security-group-name "my-ec2-sec-grp" \
  --ec2-security-group-owner-id "1234567890"
```

该命令不产生任何输出。

有关更多信息，请参阅 Elasticache 用户指南 [ElastiCache中的亚马逊自助服务更新](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AuthorizeCacheSecurityGroupIngress](#)中的。

## batch-apply-update-action

以下代码示例演示了如何使用 batch-apply-update-action。

### AWS CLI

应用服务更新

以下batch-apply-update-action示例将服务更新应用于 Redis 集群。

```
aws elasticache batch-apply-update-action \
  --service-update-name elc-xxxxx406-xxx \
```

```
--replication-group-ids test-cluster
```

输出：

```
{
  "ProcessedUpdateActions": [
    {
      "ReplicationGroupId": "pat-cluster",
      "ServiceUpdateName": "elc-xxxxx406-xxx",
      "UpdateActionStatus": "waiting-to-start"
    }
  ],
  "UnprocessedUpdateActions": []
}
```

有关更多信息，请参阅 Elasticache 用户指南 [ElastiCache 中的亚马逊自助服务更新](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchApplyUpdateAction](#) 中的。

## batch-stop-update-action

以下代码示例演示了如何使用 batch-stop-update-action。

### AWS CLI

#### 停止服务更新

以下 batch-stop-update-action 示例将服务更新应用于 Redis 集群。

```
aws elasticache batch-stop-update-action \
  --service-update-name elc-xxxxx406-xxx \
  --replication-group-ids test-cluster
```

输出：

```
{
  "ProcessedUpdateActions": [
    {
      "ReplicationGroupId": "pat-cluster",
      "ServiceUpdateName": "elc-xxxxx406-xxx",
      "UpdateActionStatus": "stopping"
    }
  ]
}
```



```
    ],  
    "UnprocessedUpdateActions": []  
  }  
}
```

有关更多信息，请参阅 Elasticache 用户指南 [ElastiCache 中的亚马逊自助服务更新](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchStopUpdateAction](#) 中的。

## copy-snapshot

以下代码示例演示了如何使用 copy-snapshot。

### AWS CLI

#### 复制快照

以下 copy-snapshot 示例创建现有快照的副本。

```
aws elasticache copy-snapshot \  
  --source-snapshot-name "my-snapshot" \  
  --target-snapshot-name "my-snapshot-copy"
```

输出：

```
{  
  "Snapshot": {  
    "Engine": "redis",  
    "CacheParameterGroupName": "default.redis3.2",  
    "VpcId": "vpc-3820329f3",  
    "CacheClusterId": "my-redis4",  
    "SnapshotRetentionLimit": 7,  
    "NumCacheNodes": 1,  
    "SnapshotName": "my-snapshot-copy",  
    "CacheClusterCreateTime": "2016-12-21T22:24:04.955Z",  
    "AutoMinorVersionUpgrade": true,  
    "PreferredAvailabilityZone": "us-east-1c",  
    "SnapshotStatus": "creating",  
    "SnapshotSource": "manual",  
    "SnapshotWindow": "07:00-08:00",  
    "EngineVersion": "3.2.4",  
    "NodeSnapshots": [  
      {  
        "CacheSize": "3 MB",  
      }  
    ]  
  }  
}
```

```

        "SnapshotCreateTime": "2016-12-28T07:00:52Z",
        "CacheNodeId": "0001",
        "CacheNodeCreateTime": "2016-12-21T22:24:04.955Z"
    }
],
"CacheSubnetGroupName": "default",
"Port": 6379,
"PreferredMaintenanceWindow": "tue:09:30-tue:10:30",
"CacheNodeType": "cache.m3.large"
}
}

```

有关更多信息，请参阅《Elasticache 用户指南》中的[导出备份](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CopySnapshot](#)中的。

## create-cache-cluster

以下代码示例演示了如何使用 `create-cache-cluster`。

### AWS CLI

#### 创建缓存集群

以下`create-cache-cluster`示例使用 Redis 引擎创建缓存集群。

```

aws elasticache create-cache-cluster \
  --cache-cluster-id "cluster-test" \
  --engine redis \
  --cache-node-type cache.m5.large \
  --num-cache-nodes 1

```

输出：

```

{
  "CacheCluster": {
    "CacheClusterId": "cluster-test",
    "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
    "CacheNodeType": "cache.m5.large",
    "Engine": "redis",
    "EngineVersion": "5.0.5",
    "CacheClusterStatus": "creating",

```

```
    "NumCacheNodes": 1,
    "PreferredMaintenanceWindow": "sat:13:00-sat:14:00",
    "PendingModifiedValues": {},
    "CacheSecurityGroups": [],
    "CacheParameterGroup": {
      "CacheParameterGroupName": "default.redis5.0",
      "ParameterApplyStatus": "in-sync",
      "CacheNodeIdsToReboot": []
    },
    "CacheSubnetGroupName": "default",
    "AutoMinorVersionUpgrade": true,
    "SnapshotRetentionLimit": 0,
    "SnapshotWindow": "06:30-07:30",
    "TransitEncryptionEnabled": false,
    "AtRestEncryptionEnabled": false
  }
}
```

有关更多信息，请参阅《Elasticache 用户指南》中的[创建集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCacheCluster](#)中的。

## create-cache-parameter-group

以下代码示例演示了如何使用 create-cache-parameter-group。

### AWS CLI

#### 创建缓存参数组

以下create-cache-parameter-group示例创建了一个新的 Amazon ElastiCache 缓存参数组。

```
aws elasticache create-cache-parameter-group \
  --cache-parameter-group-family "redis5.0" \
  --cache-parameter-group-name "mygroup" \
  --description "mygroup"
```

输出：

```
{
  "CacheParameterGroup": {
```

```
    "CacheParameterGroupName": "mygroup",
    "CacheParameterGroupFamily": "redis5.0",
    "Description": "my group"
  }
}
```

有关更多信息，请参阅《Elasticache 用户指南》中的[创建参数组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCacheParameterGroup](#)中的。

## create-cache-subnet-group

以下代码示例演示了如何使用 create-cache-subnet-group。

### AWS CLI

#### 创建缓存子网组

以下create-cache-subnet-group示例创建了一个新的缓存子网组。

```
aws elasticache create-cache-subnet-group \
  --cache-subnet-group-name "mygroup" \
  --cache-subnet-group-description "my subnet group" \
  --subnet-ids "subnet-xxxxec4f"
```

输出：

```
{
  "CacheSubnetGroup": {
    "CacheSubnetGroupName": "mygroup",
    "CacheSubnetGroupDescription": "my subnet group",
    "VpcId": "vpc-a3e97cdb",
    "Subnets": [
      {
        "SubnetIdentifier": "subnet-xxxxec4f",
        "SubnetAvailabilityZone": {
          "Name": "us-west-2d"
        }
      }
    ]
  }
}
```

有关更多信息，请参阅《Elasticache 用户指南》中的[创建缓存子网组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCacheSubnetGroup](#)中的。

## create-global-replication-group

以下代码示例演示了如何使用 create-global-replication-group。

### AWS CLI

#### 创建全局复制组

以下create-global-replication-group示例创建了一个新的全局复制组。

```
aws elasticache create-global-replication-group \  
  --global-replication-group-id-suffix my-global-replication-group \  
  --primary-replication-group-id my-primary-cluster
```

输出：

```
{  
  "GlobalReplicationGroup": {  
    "GlobalReplicationGroupId": "sgaui-my-global-replication-group",  
    "GlobalReplicationGroupDescription": " ",  
    "Status": "creating",  
    "CacheNodeType": "cache.r5.large",  
    "Engine": "redis",  
    "EngineVersion": "5.0.6",  
    "Members": [  
      {  
        "ReplicationGroupId": "my-primary-cluster",  
        "ReplicationGroupRegion": "us-west-2",  
        "Role": "PRIMARY",  
        "AutomaticFailover": "enabled",  
        "Status": "associating"  
      }  
    ],  
    "ClusterEnabled": true,  
    "GlobalNodeGroups": [  
      {  
        "GlobalNodeGroupId": "sgaui-my-global-replication-group-0001",  
        "Slots": "0-16383"  
      }  
    ]  
  }  
}
```

```

    ],
    "AuthTokenEnabled": false,
    "TransitEncryptionEnabled": false,
    "AtRestEncryptionEnabled": false
  }
}

```

有关更多信息，请参阅《Elasticache 用户指南》中的[使用全球数据存储跨 AWS 区域复制](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateGlobalReplicationGroup](#)中的。

## create-replication-group

以下代码示例演示了如何使用 create-replication-group。

### AWS CLI

#### 创建复制组

以下 create-replication-group 示例创建 Redis (已禁用集群模式) 或 Redis (已启用集群模式) 复制组。此操作仅对 Redis 有效。

```

aws elasticache create-replication-group \
  --replication-group-id "mygroup" \
  --replication-group-description "my group" \
  --engine "redis" \
  --cache-node-type "cache.m5.large"

```

输出：

```

{
  "ReplicationGroup": {
    "ReplicationGroupId": "mygroup",
    "Description": "my group",
    "Status": "creating",
    "PendingModifiedValues": {},
    "MemberClusters": [
      "mygroup-001"
    ],
    "AutomaticFailover": "disabled",
    "SnapshotRetentionLimit": 0,
    "SnapshotWindow": "06:00-07:00",
  }
}

```

```
    "ClusterEnabled": false,  
    "CacheNodeType": "cache.m5.large",  
    "TransitEncryptionEnabled": false,  
    "AtRestEncryptionEnabled": false  
  }  
}
```

有关更多信息，请参阅 Elasticache [用户指南中的创建 Redis 复制组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateReplicationGroup](#) 中的。

## create-snapshot

以下代码示例演示了如何使用 create-snapshot。

### AWS CLI

#### 创建快照

以下 create-snapshot 示例使用 Redis 引擎创建快照。

```
aws elasticache create-snapshot \  
  --snapshot-name mysnapshot \  
  --cache-cluster-id cluster-test
```

输出：

```
{  
  "Snapshot": {  
    "SnapshotName": "mysnapshot",  
    "CacheClusterId": "cluster-test",  
    "SnapshotStatus": "creating",  
    "SnapshotSource": "manual",  
    "CacheNodeType": "cache.m5.large",  
    "Engine": "redis",  
    "EngineVersion": "5.0.5",  
    "NumCacheNodes": 1,  
    "PreferredAvailabilityZone": "us-west-2b",  
    "CacheClusterCreateTime": "2020-03-19T03:12:01.483Z",  
    "PreferredMaintenanceWindow": "sat:13:00-sat:14:00",  
    "Port": 6379,  
    "CacheParameterGroupName": "default.redis5.0",  
    "CacheSubnetGroupName": "default",
```

```

    "VpcId": "vpc-a3e97cdb",
    "AutoMinorVersionUpgrade": true,
    "SnapshotRetentionLimit": 0,
    "SnapshotWindow": "06:30-07:30",
    "NodeSnapshots": [
      {
        "CacheNodeId": "0001",
        "CacheSize": "",
        "CacheNodeCreateTime": "2020-03-19T03:12:01.483Z"
      }
    ]
  }
}

```

有关更多信息，请参阅《Elasticache [用户 ElastiCache 指南](#)》中的 [Redis 备份和恢复](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSnapshot](#)中的。

## create-user-group

以下代码示例演示了如何使用 create-user-group。

### AWS CLI

#### 创建用户组

以下create-user-group示例创建了一个新的用户组。

```

aws elasticache create-user-group \
  --user-group-id myusergroup \
  --engine redis \
  --user-ids default

```

#### 输出：

```

{
  "UserGroupId": "myusergroup",
  "Status": "creating",
  "Engine": "redis",
  "UserIds": [
    "default"
  ],
  "ReplicationGroups": [],
}

```



```
"ARN": "arn:aws:elasticache:us-west-2:xxxxxxxxxx52:usergroup:myusergroup"
}
```

有关更多信息，请参阅《Elasticache [用户指南](#)》中的[使用基于角色的访问控制 \(RBAC\) 对用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUserGroup](#)中的。

## create-user

以下代码示例演示了如何使用 create-user。

### AWS CLI

#### 创建用户

以下create-user示例创建了一个新用户。

```
aws elasticache create-user \  
  --user-id user1 \  
  --user-name myUser \  
  --passwords mYnuUzrpAxXw2rdzx \  
  --engine redis \  
  --access-string "on ~app:* -@all +@read"
```

输出：

```
{  
  "UserId": "user2",  
  "UserName": "myUser",  
  "Status": "active",  
  "Engine": "redis",  
  "AccessString": "on ~app:* -@all +@read +@hash +@bitmap +@geo -setbit -bitfield  
-hset -hsetnx -hmset -hincrby -hincrbyfloat -hdel -bitop -geoadd -georadius -  
georadiusbymember",  
  "UserGroupIds": [],  
  "Authentication": {  
    "Type": "password",  
    "PasswordCount": 1  
  },  
  "ARN": "arn:aws:elasticache:us-west-2:xxxxxxxxxx52:user:user2"  
}
```

有关更多信息，请参阅《Elasticache [用户指南](#)》中的[使用基于角色的访问控制 \(RBAC\) 对用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUser](#)中的。

## decrease-node-groups-in-global-replication-group

以下代码示例演示了如何使用 decrease-node-groups-in-global-replication-group。

### AWS CLI

减少全局复制组中的节点组数量

以下方法使用 Redis 引擎decrease-node-groups-in-global-replication-group减少了节点组数量。

```
aws elasticache decrease-node-groups-in-global-replication-group \  
  --global-replication-group-id sgau-test \  
  --node-group-count 1 \  
  --apply-immediately \  
  --global-node-groups-to-retain sgau-test-0003
```

输出：

```
{  
  "GlobalReplicationGroup":  
  {  
    "GlobalReplicationGroupId": "sgau-test",  
    "GlobalReplicationGroupDescription": "test",  
    "Status": "modifying",  
    "CacheNodeType": "cache.r5.large",  
    "Engine": "redis",  
    "EngineVersion": "5.0.6",  
    "Members": [  
      {  
        "ReplicationGroupId": "test-2",  
        "ReplicationGroupRegion": "us-east-1",  
        "Role": "SECONDARY",  
        "AutomaticFailover": "enabled",  
        "Status": "associated"  
      },  
      {  
        "ReplicationGroupId": "test-1",
```

```

        "ReplicationGroupRegion": "us-west-2",
        "Role": "PRIMARY",
        "AutomaticFailover": "enabled",
        "Status": "associated"
    }
],
"ClusterEnabled": true,
"GlobalNodeGroups": [
    {
        "GlobalNodeGroupId": "sgaui-test-0001",
        "Slots": "0-449,1816-5461"
    },
    {
        "GlobalNodeGroupId": "sgaui-test-0002",
        "Slots": "6827-10922"
    },
    {
        "GlobalNodeGroupId": "sgaui-test-0003",
        "Slots": "10923-14052,15418-16383"
    },
    {
        "GlobalNodeGroupId": "sgaui-test-0004",
        "Slots": "450-1815,5462-6826,14053-15417"
    }
],
"AuthTokenEnabled": false,
"TransitEncryptionEnabled": false,
"AtRestEncryptionEnabled": false
}
}

```

有关更多信息，请参阅《Elasticache 用户指南》中的[使用全球数据存储跨 AWS 区域复制](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DecreaseNodeGroupsInGlobalReplicationGroup](#) 中的。

## decrease-replica-count

以下代码示例演示了如何使用 decrease-replica-count。

### AWS CLI

#### 减少副本数量

以下decrease-replica-count示例动态减少 Redis ( 已禁用集群模式 ) 复制组中的副本数量或 Redis ( 已启用集群模式 ) 复制组的一个或多个节点组 ( 分片 ) 中的副本节点数。此操作是在没有集群停机时间的情况下执行的。

```
aws elasticache decrease-replica-count \  
  --replication-group-id my-cluster \  
  --apply-immediately \  
  --new-replica-count 2
```

输出：

```
{  
  "ReplicationGroup": {  
    "ReplicationGroupId": "my-cluster",  
    "Description": " ",  
    "Status": "modifying",  
    "PendingModifiedValues": {},  
    "MemberClusters": [  
      "myrepliac",  
      "my-cluster-001",  
      "my-cluster-002",  
      "my-cluster-003"  
    ],  
    "NodeGroups": [  
      {  
        "NodeGroupId": "0001",  
        "Status": "modifying",  
        "PrimaryEndpoint": {  
          "Address": "my-cluster.xxxxx.ng.0001.usw2.cache.amazonaws.com",  
          "Port": 6379  
        },  
        "ReaderEndpoint": {  
          "Address": "my-cluster-  
ro.xxxxx.ng.0001.usw2.cache.amazonaws.com",  
          "Port": 6379  
        },  
        "NodeGroupMembers": [  
          {  
            "CacheClusterId": "myrepliac",  
            "CacheNodeId": "0001",  
            "ReadEndpoint": {  
              "Address":  
"myrepliac.xxxxx.0001.usw2.cache.amazonaws.com",
```

```
        "Port": 6379
      },
      "PreferredAvailabilityZone": "us-west-2a",
      "CurrentRole": "replica"
    },
    {
      "CacheClusterId": "my-cluster-001",
      "CacheNodeId": "0001",
      "ReadEndpoint": {
        "Address": "my-
cluster-001.xxxxx.0001.usw2.cache.amazonaws.com",
        "Port": 6379
      },
      "PreferredAvailabilityZone": "us-west-2a",
      "CurrentRole": "primary"
    },
    {
      "CacheClusterId": "my-cluster-002",
      "CacheNodeId": "0001",
      "ReadEndpoint": {
        "Address": "my-
cluster-002.xxxxx.0001.usw2.cache.amazonaws.com",
        "Port": 6379
      },
      "PreferredAvailabilityZone": "us-west-2a",
      "CurrentRole": "replica"
    },
    {
      "CacheClusterId": "my-cluster-003",
      "CacheNodeId": "0001",
      "ReadEndpoint": {
        "Address": "my-
cluster-003.xxxxx.0001.usw2.cache.amazonaws.com",
        "Port": 6379
      },
      "PreferredAvailabilityZone": "us-west-2a",
      "CurrentRole": "replica"
    }
  ]
}
],
"AutomaticFailover": "disabled",
"SnapshotRetentionLimit": 0,
"SnapshotWindow": "07:30-08:30",
```

```
    "ClusterEnabled": false,  
    "CacheNodeType": "cache.r5.xlarge",  
    "TransitEncryptionEnabled": false,  
    "AtRestEncryptionEnabled": false  
  }  
}
```

有关更多信息，请参阅 Elasticache [用户指南中的更改副本数量](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DecreaseReplicaCount](#)中的。

## delete-cache-cluster

以下代码示例演示了如何使用 delete-cache-cluster。

### AWS CLI

#### 删除缓存集群

以下delete-cache-cluster示例删除了之前配置的指定集群。该命令将删除所有关联的缓存节点、节点端点和集群本身。当您收到此操作的成功响应后，Amazon 会 ElastiCache 立即开始删除集群；您无法取消或恢复此操作。

此操作对以下情况无效：

Redis ( 已启用集群模式 ) 集群作为复制组的最后一个只读副本的集群启用了多可用区模式的节点组 ( 分片 ) 来自 Redis ( 已启用集群模式 ) 复制组的集群未处于可用状态的集群

```
aws elasticache delete-cache-cluster \  
  --cache-cluster-id "my-cluster-002"
```

输出：

```
{  
  "CacheCluster": {  
    "CacheClusterId": "my-cluster-002",  
    "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/  
home#client-download:",  
    "CacheNodeType": "cache.r5.xlarge",  
    "Engine": "redis",  
    "EngineVersion": "5.0.5",
```

```

    "CacheClusterStatus": "deleting",
    "NumCacheNodes": 1,
    "PreferredAvailabilityZone": "us-west-2a",
    "CacheClusterCreateTime": "2019-11-26T03:35:04.546Z",
    "PreferredMaintenanceWindow": "mon:04:05-mon:05:05",
    "PendingModifiedValues": {},
    "NotificationConfiguration": {
      "TopicArn": "arn:aws:sns:us-west-x:xxxxxxx4152:My_Topic",
      "TopicStatus": "active"
    },
    "CacheSecurityGroups": [],
    "CacheParameterGroup": {
      "CacheParameterGroupName": "mygroup",
      "ParameterApplyStatus": "in-sync",
      "CacheNodeIdsToReboot": []
    },
    "CacheSubnetGroupName": "kxkxk",
    "AutoMinorVersionUpgrade": true,
    "SecurityGroups": [
      {
        "SecurityGroupId": "sg-xxxxxxxxxx9836",
        "Status": "active"
      },
      {
        "SecurityGroupId": "sg-xxxxxxxxxxx7b",
        "Status": "active"
      }
    ],
    "ReplicationGroupId": "my-cluster",
    "SnapshotRetentionLimit": 0,
    "SnapshotWindow": "07:30-08:30",
    "TransitEncryptionEnabled": false,
    "AtRestEncryptionEnabled": false
  }
}

```

有关更多信息，请参阅 Elasticache 用户指南中的[删除集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCacheCluster](#)中的。

## delete-cache-parameter-group

以下代码示例演示了如何使用 delete-cache-parameter-group。

## AWS CLI

### 删除缓存参数组

以下delete-cache-parameter-group示例删除了指定的缓存参数组。如果缓存参数组与任何缓存集群相关联，则无法将其删除。

```
aws elasticache delete-cache-parameter-group \  
  --cache-parameter-group-name myparamgroup
```

此命令不生成任何输出。

有关更多信息，请参阅Elasticache用户指南中的[删除参数组](#)。

- 有关API的详细信息，请参阅AWS CLI命令参考[DeleteCacheParameterGroup](#)中的。

### delete-cache-subnet-group

以下代码示例演示了如何使用delete-cache-subnet-group。

## AWS CLI

### 删除缓存子网组

以下delete-cache-subnet-group示例删除了指定的缓存子网组。如果缓存子网组与任何集群关联，则无法将其删除。

```
aws elasticache delete-cache-subnet-group \  
  --cache-subnet-group-name "mygroup"
```

此命令不生成任何输出。

有关更多信息，请参阅《Elasticache用户指南》中的[删除子网组](#)。

- 有关API的详细信息，请参阅AWS CLI命令参考[DeleteCacheSubnetGroup](#)中的。

### delete-global-replication-group

以下代码示例演示了如何使用delete-global-replication-group。

## AWS CLI

### 删除全局复制组



以下delete-global-replication-group示例删除了一个新的全局复制组。

```
aws elasticache delete-global-replication-group \  
  --global-replication-group-id my-global-replication-group \  
  --retain-primary-replication-group
```

输出：

```
{  
  "GlobalReplicationGroup": {  
    "GlobalReplicationGroupId": "sgaui-my-grg",  
    "GlobalReplicationGroupDescription": "my-grg",  
    "Status": "deleting",  
    "CacheNodeType": "cache.r5.large",  
    "Engine": "redis",  
    "EngineVersion": "5.0.6",  
    "Members": [  
      {  
        "ReplicationGroupId": "my-cluster-grg",  
        "ReplicationGroupRegion": "us-west-2",  
        "Role": "PRIMARY",  
        "AutomaticFailover": "enabled",  
        "Status": "associated"  
      }  
    ],  
    "ClusterEnabled": false,  
    "AuthTokenEnabled": false,  
    "TransitEncryptionEnabled": false,  
    "AtRestEncryptionEnabled": false  
  }  
}
```

有关更多信息，请参阅《Elasticache 用户指南》中的[使用全球数据存储跨 AWS 区域复制](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteGlobalReplicationGroup](#)中的。

## delete-replication-group

以下代码示例演示了如何使用 delete-replication-group。

### AWS CLI

#### 删除复制组

以下delete-replication-group示例删除现有的复制组。默认情况下，此操作会删除整个复制组，包括主副本/主副本和所有只读副本。如果复制组只有一个主副本，则可以选择只删除只读副本，同时通过设置 RetainPrimaryCluster =true 保留主副本。

当您收到此操作的成功响应后，Amazon 会 ElastiCache 立即开始删除所选资源；您无法取消或恢复此操作。仅适用于 Redis。

```
aws elasticache delete-replication-group \  
  --replication-group-id "mygroup"
```

输出：

```
{  
  "ReplicationGroup": {  
    "ReplicationGroupId": "mygroup",  
    "Description": "my group",  
    "Status": "deleting",  
    "PendingModifiedValues": {},  
    "AutomaticFailover": "disabled",  
    "SnapshotRetentionLimit": 0,  
    "SnapshotWindow": "06:00-07:00",  
    "TransitEncryptionEnabled": false,  
    "AtRestEncryptionEnabled": false  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteReplicationGroup](#)中的。

## delete-snapshot

以下代码示例演示了如何使用 delete-snapshot。

### AWS CLI

#### 删除快照

以下delete-snapshot示例使用 Redis 引擎删除了快照。

```
aws elasticache delete-snapshot \  
  --snapshot-name mysnapshot
```

输出：

```
{
  "Snapshot": {
    "SnapshotName": "my-cluster-snapshot",
    "ReplicationGroupId": "mycluster",
    "ReplicationGroupDescription": "mycluster",
    "SnapshotStatus": "deleting",
    "SnapshotSource": "manual",
    "CacheNodeType": "cache.r5.xlarge",
    "Engine": "redis",
    "EngineVersion": "5.0.5",
    "PreferredMaintenanceWindow": "thu:12:00-thu:13:00",
    "TopicArn": "arn:aws:sns:us-west-2:xxxxxxxxxxxxx152:My_Topic",
    "Port": 6379,
    "CacheParameterGroupName": "default.redis5.0.cluster.on",
    "CacheSubnetGroupName": "default",
    "VpcId": "vpc-a3e97cdb",
    "AutoMinorVersionUpgrade": true,
    "SnapshotRetentionLimit": 1,
    "SnapshotWindow": "13:00-14:00",
    "NumNodeGroups": 4,
    "AutomaticFailover": "enabled",
    "NodeSnapshots": [
      {
        "CacheClusterId": "mycluster-0002-003",
        "NodeGroupId": "0002",
        "CacheNodeId": "0001",
        "CacheSize": "6 MB",
        "CacheNodeCreateTime": "2020-06-18T00:05:44.719000+00:00",
        "SnapshotCreateTime": "2020-06-25T20:34:30+00:00"
      },
      {
        "CacheClusterId": "mycluster-0003-003",
        "NodeGroupId": "0003",
        "CacheNodeId": "0001",
        "CacheSize": "6 MB",
        "CacheNodeCreateTime": "2019-12-05T19:13:15.912000+00:00",
        "SnapshotCreateTime": "2020-06-25T20:34:30+00:00"
      },
      {
        "CacheClusterId": "mycluster-0004-002",
        "NodeGroupId": "0004",
        "CacheNodeId": "0001",
```

```

        "CacheSize": "6 MB",
        "CacheNodeCreateTime": "2019-12-09T19:44:34.324000+00:00",
        "SnapshotCreateTime": "2020-06-25T20:34:30+00:00"
    },
    {
        "CacheClusterId": "mycluster-0005-003",
        "NodeGroupId": "0005",
        "CacheNodeId": "0001",
        "CacheSize": "6 MB",
        "CacheNodeCreateTime": "2020-06-18T00:05:44.775000+00:00",
        "SnapshotCreateTime": "2020-06-25T20:34:30+00:00"
    }
]
}
}

```

有关更多信息，请参阅《Elasticache [用户 ElastiCache 指南](#)》中的 [Redis 备份和恢复](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSnapshot](#)中的。

## delete-user-group

以下代码示例演示了如何使用 delete-user-group。

### AWS CLI

#### 删除用户组

以下delete-user-group示例删除用户组。

```
aws elasticache delete-user-group \
  --user-group-id myusergroup
```

输出：

```
{
  "UserGroupId": "myusergroup",
  "Status": "deleting",
  "Engine": "redis",
  "UserIds": [
    "default"
  ],
  "ReplicationGroups": [],
}
```

```
"ARN": "arn:aws:elasticache:us-west-2:xxxxxxxxxx52:usergroup:myusergroup"
}
```

有关更多信息，请参阅《Elasticache [用户指南](#)》中的[使用基于角色的访问控制 \(RBAC\) 对用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUserGroup](#)中的。

## delete-user

以下代码示例演示了如何使用 delete-user。

### AWS CLI

删除用户

以下delete-user示例删除用户。

```
aws elasticache delete-user \  
  --user-id user2
```

输出：

```
{  
  "UserId": "user1",  
  "UserName": "myUser",  
  "Status": "deleting",  
  "Engine": "redis",  
  "AccessString": "on ~* +@all",  
  "UserGroupIds": [  
    "myusergroup"  
  ],  
  "Authentication": {  
    "Type": "password",  
    "PasswordCount": 1  
  },  
  "ARN": "arn:aws:elasticache:us-west-2:xxxxxxxxxx52:user:user1"  
}
```

有关更多信息，请参阅《Elasticache [用户指南](#)》中的[使用基于角色的访问控制 \(RBAC\) 对用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUser](#)中的。

## describe-cache-clusters

以下代码示例演示了如何使用 describe-cache-clusters。

### AWS CLI

#### 描述缓存集群

以下describe-cache-clusters示例描述了一个缓存集群。

```
aws elasticache describe-cache-clusters
```

输出：

```
{
  "CacheClusters": [
    {
      "CacheClusterId": "my-cluster-003",
      "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
      "CacheNodeType": "cache.r5.large",
      "Engine": "redis",
      "EngineVersion": "5.0.5",
      "CacheClusterStatus": "available",
      "NumCacheNodes": 1,
      "PreferredAvailabilityZone": "us-west-2a",
      "CacheClusterCreateTime": "2019-11-26T01:22:52.396Z",
      "PreferredMaintenanceWindow": "mon:17:30-mon:18:30",
      "PendingModifiedValues": {},
      "NotificationConfiguration": {
        "TopicArn": "arn:aws:sns:us-west-2:xxxxxxxxxxxx152:My_Topic",
        "TopicStatus": "active"
      },
      "CacheSecurityGroups": [],
      "CacheParameterGroup": {
        "CacheParameterGroupName": "default.redis5.0",
        "ParameterApplyStatus": "in-sync",
        "CacheNodeIdsToReboot": []
      },
      "CacheSubnetGroupName": "kxkxk",
      "AutoMinorVersionUpgrade": true,
      "SecurityGroups": [
        {
```

```
        "SecurityGroupId": "sg-xxxxxd7b",
        "Status": "active"
      }
    ],
    "ReplicationGroupId": "my-cluster",
    "SnapshotRetentionLimit": 0,
    "SnapshotWindow": "06:30-07:30",
    "AuthTokenEnabled": false,
    "TransitEncryptionEnabled": false,
    "AtRestEncryptionEnabled": false,
    "ARN": "arn:aws:elasticache:us-west-2:xxxxxxxxxxxx152:cluster:my-cache-
cluster",
    "ReplicationGroupLogDeliveryEnabled": false,
    "LogDeliveryConfigurations": [
      {
        "LogType": "slow-log",
        "DestinationType": "cloudwatch-logs",
        "DestinationDetails": {
          "CloudWatchLogsDetails": {
            "LogGroup": "test-log"
          }
        },
        "LogFormat": "text",
        "Status": "active"
      }
    ]
  }
]
```

有关更多信息，请参阅《Elasticache 用户指南》中的[管理集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeCacheClusters](#) 中的。

## describe-cache-engine-versions

以下代码示例演示了如何使用 describe-cache-engine-versions。

### AWS CLI

描述缓存引擎版本

以下 describe-cache-engine-versions 示例返回可用缓存引擎及其版本的列表。

```
aws elasticache describe-cache-engine-versions \  
--engine "Redis"
```

输出：

```
{  
  "CacheEngineVersions": [  
    {  
      "Engine": "redis",  
      "EngineVersion": "2.6.13",  
      "CacheParameterGroupFamily": "redis2.6",  
      "CacheEngineDescription": "Redis",  
      "CacheEngineVersionDescription": "redis version 2.6.13"  
    },  
    {  
      "Engine": "redis",  
      "EngineVersion": "2.8.19",  
      "CacheParameterGroupFamily": "redis2.8",  
      "CacheEngineDescription": "Redis",  
      "CacheEngineVersionDescription": "redis version 2.8.19"  
    },  
    {  
      "Engine": "redis",  
      "EngineVersion": "2.8.21",  
      "CacheParameterGroupFamily": "redis2.8",  
      "CacheEngineDescription": "Redis",  
      "CacheEngineVersionDescription": "redis version 2.8.21"  
    },  
    {  
      "Engine": "redis",  
      "EngineVersion": "2.8.22",  
      "CacheParameterGroupFamily": "redis2.8",  
      "CacheEngineDescription": "Redis",  
      "CacheEngineVersionDescription": "redis version 2.8.22"  
    },  
    {  
      "Engine": "redis",  
      "EngineVersion": "2.8.23",  
      "CacheParameterGroupFamily": "redis2.8",  
      "CacheEngineDescription": "Redis",  
      "CacheEngineVersionDescription": "redis version 2.8.23"  
    }  
  ]  
}
```



```
    "Engine": "redis",
    "EngineVersion": "2.8.24",
    "CacheParameterGroupFamily": "redis2.8",
    "CacheEngineDescription": "Redis",
    "CacheEngineVersionDescription": "redis version 2.8.24"
  },
  {
    "Engine": "redis",
    "EngineVersion": "2.8.6",
    "CacheParameterGroupFamily": "redis2.8",
    "CacheEngineDescription": "Redis",
    "CacheEngineVersionDescription": "redis version 2.8.6"
  },
  {
    "Engine": "redis",
    "EngineVersion": "3.2.10",
    "CacheParameterGroupFamily": "redis3.2",
    "CacheEngineDescription": "Redis",
    "CacheEngineVersionDescription": "redis version 3.2.10"
  },
  {
    "Engine": "redis",
    "EngineVersion": "3.2.4",
    "CacheParameterGroupFamily": "redis3.2",
    "CacheEngineDescription": "Redis",
    "CacheEngineVersionDescription": "redis version 3.2.4"
  },
  {
    "Engine": "redis",
    "EngineVersion": "3.2.6",
    "CacheParameterGroupFamily": "redis3.2",
    "CacheEngineDescription": "Redis",
    "CacheEngineVersionDescription": "redis version 3.2.6"
  },
  {
    "Engine": "redis",
    "EngineVersion": "4.0.10",
    "CacheParameterGroupFamily": "redis4.0",
    "CacheEngineDescription": "Redis",
    "CacheEngineVersionDescription": "redis version 4.0.10"
  },
  {
    "Engine": "redis",
    "EngineVersion": "5.0.0",
```

```

    "CacheParameterGroupFamily": "redis5.0",
    "CacheEngineDescription": "Redis",
    "CacheEngineVersionDescription": "redis version 5.0.0"
  },
  {
    "Engine": "redis",
    "EngineVersion": "5.0.3",
    "CacheParameterGroupFamily": "redis5.0",
    "CacheEngineDescription": "Redis",
    "CacheEngineVersionDescription": "redis version 5.0.3"
  },
  {
    "Engine": "redis",
    "EngineVersion": "5.0.4",
    "CacheParameterGroupFamily": "redis5.0",
    "CacheEngineDescription": "Redis",
    "CacheEngineVersionDescription": "redis version 5.0.4"
  },
  {
    "Engine": "redis",
    "EngineVersion": "5.0.5",
    "CacheParameterGroupFamily": "redis5.0",
    "CacheEngineDescription": "Redis",
    "CacheEngineVersionDescription": "redis version 5.0.5"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCacheEngineVersions](#)中的。

## describe-cache-parameter-groups

以下代码示例演示了如何使用 describe-cache-parameter-groups。

### AWS CLI

#### 描述缓存参数组

以下describe-cache-parameter-groups示例返回缓存参数组描述的列表。

```
aws elasticache describe-cache-parameter-groups \
  --cache-parameter-group-name "mygroup"
```

输出：

```
{
  "CacheParameterGroups": [
    {
      "CacheParameterGroupName": "mygroup",
      "CacheParameterGroupFamily": "redis5.0",
      "Description": " "
    }
  ]
}
```

有关更多信息，请参阅 Elasticache 用户 [指南中的使用参数组配置引擎参数](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeCacheParameterGroups](#) 中的。

## describe-cache-parameters

以下代码示例演示了如何使用 describe-cache-parameters。

### AWS CLI

描述缓存参数

以下 “describe-cache-parameters” 示例返回指定缓存参数组的详细参数列表。

```
aws elasticache describe-cache-parameters \
  --cache-parameter-group-name "myparamgroup"
```

输出：

```
{
  "Parameters": [
    {
      "ParameterName": "activedefrag",
      "ParameterValue": "yes",
      "Description": "Enabled active memory defragmentation",
      "Source": "user",
      "DataType": "string",
      "AllowedValues": "yes,no",
      "IsModifiable": true,
      "MinimumEngineVersion": "5.0.0",
      "ChangeType": "immediate"
    }
  ]
}
```

```
    },
    {
      "ParameterName": "active-defrag-cycle-max",
      "ParameterValue": "75",
      "Description": "Maximal effort for defrag in CPU percentage",
      "Source": "user",
      "DataType": "integer",
      "AllowedValues": "1-75",
      "IsModifiable": true,
      "MinimumEngineVersion": "5.0.0",
      "ChangeType": "immediate"
    },
    {
      "ParameterName": "active-defrag-cycle-min",
      "ParameterValue": "5",
      "Description": "Minimal effort for defrag in CPU percentage",
      "Source": "user",
      "DataType": "integer",
      "AllowedValues": "1-75",
      "IsModifiable": true,
      "MinimumEngineVersion": "5.0.0",
      "ChangeType": "immediate"
    },
    {
      "ParameterName": "active-defrag-ignore-bytes",
      "ParameterValue": "104857600",
      "Description": "Minimum amount of fragmentation waste to start active
defrag",
      "Source": "user",
      "DataType": "integer",
      "AllowedValues": "1048576-",
      "IsModifiable": true,
      "MinimumEngineVersion": "5.0.0",
      "ChangeType": "immediate"
    },
    {
      "ParameterName": "active-defrag-max-scan-fields",
      "ParameterValue": "1000",
      "Description": "Maximum number of set/hash/zset/list fields that will be
processed from the main dictionary scan",
      "Source": "user",
      "DataType": "integer",
      "AllowedValues": "1-1000000",
      "IsModifiable": true,
```

```
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "active-defrag-threshold-lower",
    "ParameterValue": "10",
    "Description": "Minimum percentage of fragmentation to start active
defrag",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "1-100",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "active-defrag-threshold-upper",
    "ParameterValue": "100",
    "Description": "Maximum percentage of fragmentation at which we use
maximum effort",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "1-100",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "activeresharding",
    "ParameterValue": "yes",
    "Description": "Apply rehashing or not.",
    "Source": "user",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "requires-reboot"
  },
  {
    "ParameterName": "appendfsync",
    "ParameterValue": "everysec",
    "Description": "fsync policy for AOF persistence",
    "Source": "system",
    "DataType": "string",
```

```
    "AllowedValues": "always, everysec, no",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "appendonly",
    "ParameterValue": "no",
    "Description": "Enable Redis persistence.",
    "Source": "system",
    "DataType": "string",
    "AllowedValues": "yes, no",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-output-buffer-limit-normal-hard-limit",
    "ParameterValue": "0",
    "Description": "Normal client output buffer hard limit in bytes.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-output-buffer-limit-normal-soft-limit",
    "ParameterValue": "0",
    "Description": "Normal client output buffer soft limit in bytes.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-output-buffer-limit-normal-soft-seconds",
    "ParameterValue": "0",
    "Description": "Normal client output buffer soft limit in seconds.",
    "Source": "user",
    "DataType": "integer",
```

```
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-output-buffer-limit-pubsub-hard-limit",
    "ParameterValue": "33554432",
    "Description": "Pubsub client output buffer hard limit in bytes.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-output-buffer-limit-pubsub-soft-limit",
    "ParameterValue": "8388608",
    "Description": "Pubsub client output buffer soft limit in bytes.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-output-buffer-limit-pubsub-soft-seconds",
    "ParameterValue": "60",
    "Description": "Pubsub client output buffer soft limit in seconds.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-output-buffer-limit-replica-soft-seconds",
    "ParameterValue": "60",
    "Description": "Replica client output buffer soft limit in seconds.",
    "Source": "system",
    "DataType": "integer",
```

```
    "AllowedValues": "0-",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-query-buffer-limit",
    "ParameterValue": "1073741824",
    "Description": "Max size of a single client query buffer",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "1048576-1073741824",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "close-on-replica-write",
    "ParameterValue": "yes",
    "Description": "If enabled, clients who attempt to write to a read-only
replica will be disconnected. Applicable to 2.8.23 and higher.",
    "Source": "user",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "cluster-enabled",
    "ParameterValue": "no",
    "Description": "Enable cluster mode",
    "Source": "user",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "requires-reboot"
  },
  {
    "ParameterName": "cluster-require-full-coverage",
    "ParameterValue": "no",
    "Description": "Whether cluster becomes unavailable if one or more slots
are not covered",
```



```
    "Source": "user",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "databases",
    "ParameterValue": "16",
    "Description": "Set the number of databases.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "1-1200000",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "requires-reboot"
  },
  {
    "ParameterName": "hash-max-ziplist-entries",
    "ParameterValue": "512",
    "Description": "The maximum number of hash entries in order for the
dataset to be compressed.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "hash-max-ziplist-value",
    "ParameterValue": "64",
    "Description": "The threshold of biggest hash entries in order for the
dataset to be compressed.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "hll-sparse-max-bytes",
```

```
    "ParameterValue": "3000",
    "Description": "HyperLogLog sparse representation bytes limit",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "1-16000",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "lazyfree-lazy-eviction",
    "ParameterValue": "no",
    "Description": "Perform an asynchronous delete on evictions",
    "Source": "user",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "lazyfree-lazy-expire",
    "ParameterValue": "no",
    "Description": "Perform an asynchronous delete on expired keys",
    "Source": "user",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "lazyfree-lazy-server-del",
    "ParameterValue": "no",
    "Description": "Perform an asynchronous delete on key updates",
    "Source": "user",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "lfu-decay-time",
```

```
    "ParameterValue": "1",
    "Description": "The amount of time in minutes to decrement the key
counter for LFU eviction policy",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "lfu-log-factor",
    "ParameterValue": "10",
    "Description": "The log factor for incrementing key counter for LFU
eviction policy",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "1-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "list-compress-depth",
    "ParameterValue": "0",
    "Description": "Number of quicklist ziplist nodes from each side of
the list to exclude from compression. The head and tail of the list are always
uncompressed for fast push/pop operations",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "list-max-ziplist-size",
    "ParameterValue": "-2",
    "Description": "The number of entries allowed per internal list node can
be specified as a fixed maximum size or a maximum number of elements",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "-5,-4,-3,-2,-1,1-",
    "IsModifiable": false,
```

```
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "lua-replicate-commands",
    "ParameterValue": "yes",
    "Description": "Always enable Lua effect replication or not",
    "Source": "user",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "lua-time-limit",
    "ParameterValue": "5000",
    "Description": "Max execution time of a Lua script in milliseconds. 0
for unlimited execution without warnings.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "5000",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "maxclients",
    "ParameterValue": "65000",
    "Description": "The maximum number of Redis clients.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "1-65000",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "requires-reboot"
  },
  {
    "ParameterName": "maxmemory-policy",
    "ParameterValue": "volatile-lru",
    "Description": "Max memory policy.",
    "Source": "user",
    "DataType": "string",
```

```
    "AllowedValues": "volatile-lru,allkeys-lru,volatile-lfu,allkeys-
lfu,volatile-random,allkeys-random,volatile-ttl,noeviction",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "maxmemory-samples",
    "ParameterValue": "3",
    "Description": "Max memory samples.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "1-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "min-replicas-max-lag",
    "ParameterValue": "10",
    "Description": "The maximum amount of replica lag in seconds beyond
which the master would stop taking writes. A value of 0 means the master always
takes writes.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "min-replicas-to-write",
    "ParameterValue": "0",
    "Description": "The minimum number of replicas that must be present with
lag no greater than min-replicas-max-lag for master to take writes. Setting this to
0 means the master always takes writes.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
```

```

    "ParameterName": "notify-keyspace-events",
    "Description": "The keyspace events for Redis to notify Pub/Sub clients
about. By default all notifications are disabled",
    "Source": "user",
    "DataType": "string",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "proto-max-bulk-len",
    "ParameterValue": "536870912",
    "Description": "Max size of a single element request",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "1048576-536870912",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "rename-commands",
    "ParameterValue": "",
    "Description": "Redis commands that can be dynamically renamed by the
customer",
    "Source": "user",
    "DataType": "string",
    "AllowedValues":
"APPEND,BITCOUNT,BITFIELD,BITOP,BITPOS,BLPOP,BRPOP,BRPOPLPUSH,BZPOPMIN,BZPOPMAX,CLIENT,COMM
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.3",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "repl-backlog-size",
    "ParameterValue": "1048576",
    "Description": "The replication backlog size in bytes for PSYNC. This is
the size of the buffer which accumulates slave data when slave is disconnected for
some time, so that when slave reconnects again, only transfer the portion of data
which the slave missed. Minimum value is 16K.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "16384-",
    "IsModifiable": true,

```

```
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "repl-backlog-ttl",
    "ParameterValue": "3600",
    "Description": "The amount of time in seconds after the master no longer
have any slaves connected for the master to free the replication backlog. A value
of 0 means to never release the backlog.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "replica-allow-chaining",
    "ParameterValue": "no",
    "Description": "Configures if chaining of replicas is allowed",
    "Source": "system",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "replica-ignore-maxmemory",
    "ParameterValue": "yes",
    "Description": "Determines if replica ignores maxmemory setting by not
evicting items independent from the master",
    "Source": "system",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "replica-lazy-flush",
    "ParameterValue": "no",
    "Description": "Perform an asynchronous flushDB during replica sync",
    "Source": "system",
```

```
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "reserved-memory-percent",
    "ParameterValue": "25",
    "Description": "The percent of memory reserved for non-cache memory
usage. You may want to increase this parameter for nodes with read replicas, AOF
enabled, etc, to reduce swap usage.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-100",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "set-max-intset-entries",
    "ParameterValue": "512",
    "Description": "The limit in the size of the set in order for the
dataset to be compressed.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "slowlog-log-slower-than",
    "ParameterValue": "10000",
    "Description": "The execution time, in microseconds, to exceed in order
for the command to get logged. Note that a negative number disables the slow log,
while a value of zero forces the logging of every command.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
},
```



```
{
  "ParameterName": "slowlog-max-len",
  "ParameterValue": "128",
  "Description": "The length of the slow log. There is no limit to this
length. Just be aware that it will consume memory. You can reclaim memory used by
the slow log with SLOWLOG RESET.",
  "Source": "user",
  "DataType": "integer",
  "AllowedValues": "0-",
  "IsModifiable": true,
  "MinimumEngineVersion": "5.0.0",
  "ChangeType": "immediate"
},
{
  "ParameterName": "stream-node-max-bytes",
  "ParameterValue": "4096",
  "Description": "The maximum size of a single node in a stream in bytes",
  "Source": "user",
  "DataType": "integer",
  "AllowedValues": "0-",
  "IsModifiable": true,
  "MinimumEngineVersion": "5.0.0",
  "ChangeType": "immediate"
},
{
  "ParameterName": "stream-node-max-entries",
  "ParameterValue": "100",
  "Description": "The maximum number of items a single node in a stream
can contain",
  "Source": "user",
  "DataType": "integer",
  "AllowedValues": "0-",
  "IsModifiable": true,
  "MinimumEngineVersion": "5.0.0",
  "ChangeType": "immediate"
},
{
  "ParameterName": "tcp-keepalive",
  "ParameterValue": "300",
  "Description": "If non-zero, send ACKs every given number of seconds.",
  "Source": "user",
  "DataType": "integer",
  "AllowedValues": "0-",
  "IsModifiable": true,
```

```
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "timeout",
    "ParameterValue": "0",
    "Description": "Close connection if client is idle for a given number of
seconds, or never if 0.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0,20-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "zset-max-ziplist-entries",
    "ParameterValue": "128",
    "Description": "The maximum number of sorted set entries in order for
the dataset to be compressed.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "zset-max-ziplist-value",
    "ParameterValue": "64",
    "Description": "The threshold of biggest sorted set entries in order for
the dataset to be compressed.",
    "Source": "user",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  }
]
}
```

有关更多信息，请参阅《Elasticache 用户指南》中的[参数管理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCacheParameters](#)中的。

## describe-cache-subnet-groups

以下代码示例演示了如何使用 describe-cache-subnet-groups。

### AWS CLI

描述缓存子网组

以下describe-cache-subnet-groups示例返回子网组列表。

```
aws elasticache describe-cache-subnet-groups
```

输出：

```
{
  "CacheSubnetGroups": [
    {
      "CacheSubnetGroupName": "default",
      "CacheSubnetGroupDescription": "Default CacheSubnetGroup",
      "VpcId": "vpc-a3e97cdb",
      "Subnets": [
        {
          "SubnetIdentifier": "subnet-8d4bacf5",
          "SubnetAvailabilityZone": {
            "Name": "us-west-2b"
          }
        },
        {
          "SubnetIdentifier": "subnet-dde21380",
          "SubnetAvailabilityZone": {
            "Name": "us-west-2c"
          }
        },
        {
          "SubnetIdentifier": "subnet-6485ec4f",
          "SubnetAvailabilityZone": {
            "Name": "us-west-2d"
          }
        },
        {
          "SubnetIdentifier": "subnet-b4ebebff",
```

```

        "SubnetAvailabilityZone": {
            "Name": "us-west-2a"
        }
    ]
},
{
    "CacheSubnetGroupName": "kxkxk",
    "CacheSubnetGroupDescription": "mygroup",
    "VpcId": "vpc-a3e97cdb",
    "Subnets": [
        {
            "SubnetIdentifier": "subnet-b4ebebff",
            "SubnetAvailabilityZone": {
                "Name": "us-west-2a"
            }
        }
    ]
},
{
    "CacheSubnetGroupName": "test",
    "CacheSubnetGroupDescription": "test",
    "VpcId": "vpc-a3e97cdb",
    "Subnets": [
        {
            "SubnetIdentifier": "subnet-b4ebebff",
            "SubnetAvailabilityZone": {
                "Name": "us-west-2a"
            }
        }
    ]
}
]
}

```

有关更多信息，请参阅 Elasticache 用户指南中的[子网和子网组](#)或ElastiCache 适用于 Memcached 的用户指南中的[子网和子网组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCacheSubnetGroups](#)中的。

## describe-engine-default-parameters

以下代码示例演示了如何使用 describe-engine-default-parameters。

## AWS CLI

### 描述引擎默认参数

以下describe-engine-default-parameters示例返回指定缓存引擎的默认引擎和系统参数信息。

```
aws elasticache describe-engine-default-parameters \  
  --cache-parameter-group-family "redis5.0"
```

输出：

```
{  
  "EngineDefaults": {  
    "Parameters": [  
      {  
        "ParameterName": "activedefrag",  
        "ParameterValue": "no",  
        "Description": "Enabled active memory defragmentation",  
        "Source": "system",  
        "DataType": "string",  
        "AllowedValues": "yes,no",  
        "IsModifiable": true,  
        "MinimumEngineVersion": "5.0.0",  
        "ChangeType": "immediate"  
      },  
      {  
        "ParameterName": "active-defrag-cycle-max",  
        "ParameterValue": "75",  
        "Description": "Maximal effort for defrag in CPU percentage",  
        "Source": "system",  
        "DataType": "integer",  
        "AllowedValues": "1-75",  
        "IsModifiable": true,  
        "MinimumEngineVersion": "5.0.0",  
        "ChangeType": "immediate"  
      },  
      {  
        "ParameterName": "active-defrag-cycle-min",  
        "ParameterValue": "5",  
        "Description": "Minimal effort for defrag in CPU percentage",  
        "Source": "system",  
        "DataType": "integer",
```

```
        "AllowedValues": "1-75",
        "IsModifiable": true,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "immediate"
    },
    {
        "ParameterName": "active-defrag-ignore-bytes",
        "ParameterValue": "104857600",
        "Description": "Minimum amount of fragmentation waste to start
active defrag",
        "Source": "system",
        "DataType": "integer",
        "AllowedValues": "1048576-",
        "IsModifiable": true,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "immediate"
    },
    {
        "ParameterName": "active-defrag-max-scan-fields",
        "ParameterValue": "1000",
        "Description": "Maximum number of set/hash/zset/list fields that
will be processed from the main dictionary scan",
        "Source": "system",
        "DataType": "integer",
        "AllowedValues": "1-1000000",
        "IsModifiable": true,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "immediate"
    },
    {
        "ParameterName": "active-defrag-threshold-lower",
        "ParameterValue": "10",
        "Description": "Minimum percentage of fragmentation to start active
defrag",
        "Source": "system",
        "DataType": "integer",
        "AllowedValues": "1-100",
        "IsModifiable": true,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "immediate"
    },
    {
        "ParameterName": "active-defrag-threshold-upper",
        "ParameterValue": "100",
```

```
        "Description": "Maximum percentage of fragmentation at which we use
maximum effort",
        "Source": "system",
        "DataType": "integer",
        "AllowedValues": "1-100",
        "IsModifiable": true,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "immediate"
    },
    {
        "ParameterName": "activeresharding",
        "ParameterValue": "yes",
        "Description": "Apply rehashing or not.",
        "Source": "system",
        "DataType": "string",
        "AllowedValues": "yes,no",
        "IsModifiable": false,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "requires-reboot"
    },
    {
        "ParameterName": "appendfsync",
        "ParameterValue": "everysec",
        "Description": "fsync policy for AOF persistence",
        "Source": "system",
        "DataType": "string",
        "AllowedValues": "always,everysec,no",
        "IsModifiable": false,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "immediate"
    },
    {
        "ParameterName": "appendonly",
        "ParameterValue": "no",
        "Description": "Enable Redis persistence.",
        "Source": "system",
        "DataType": "string",
        "AllowedValues": "yes,no",
        "IsModifiable": false,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "immediate"
    },
    {
        "ParameterName": "client-output-buffer-limit-normal-hard-limit",
```

```
    "ParameterValue": "0",
    "Description": "Normal client output buffer hard limit in bytes.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-output-buffer-limit-normal-soft-limit",
    "ParameterValue": "0",
    "Description": "Normal client output buffer soft limit in bytes.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-output-buffer-limit-normal-soft-seconds",
    "ParameterValue": "0",
    "Description": "Normal client output buffer soft limit in seconds.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-output-buffer-limit-pubsub-hard-limit",
    "ParameterValue": "33554432",
    "Description": "Pubsub client output buffer hard limit in bytes.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-output-buffer-limit-pubsub-soft-limit",
```



```
    "ParameterValue": "8388608",
    "Description": "Pubsub client output buffer soft limit in bytes.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-output-buffer-limit-pubsub-soft-seconds",
    "ParameterValue": "60",
    "Description": "Pubsub client output buffer soft limit in seconds.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-output-buffer-limit-replica-soft-seconds",
    "ParameterValue": "60",
    "Description": "Replica client output buffer soft limit in
seconds.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "client-query-buffer-limit",
    "ParameterValue": "1073741824",
    "Description": "Max size of a single client query buffer",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "1048576-1073741824",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
```

```
    "ParameterName": "close-on-replica-write",
    "ParameterValue": "yes",
    "Description": "If enabled, clients who attempt to write to a read-
only replica will be disconnected. Applicable to 2.8.23 and higher.",
    "Source": "system",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "cluster-enabled",
    "ParameterValue": "no",
    "Description": "Enable cluster mode",
    "Source": "system",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "requires-reboot"
  },
  {
    "ParameterName": "cluster-require-full-coverage",
    "ParameterValue": "no",
    "Description": "Whether cluster becomes unavailable if one or more
slots are not covered",
    "Source": "system",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "databases",
    "ParameterValue": "16",
    "Description": "Set the number of databases.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "1-1200000",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "requires-reboot"
  }
}
```

```
    },
    {
      "ParameterName": "hash-max-ziplist-entries",
      "ParameterValue": "512",
      "Description": "The maximum number of hash entries in order for the
dataset to be compressed.",
      "Source": "system",
      "DataType": "integer",
      "AllowedValues": "0-",
      "IsModifiable": true,
      "MinimumEngineVersion": "5.0.0",
      "ChangeType": "immediate"
    },
    {
      "ParameterName": "hash-max-ziplist-value",
      "ParameterValue": "64",
      "Description": "The threshold of biggest hash entries in order for
the dataset to be compressed.",
      "Source": "system",
      "DataType": "integer",
      "AllowedValues": "0-",
      "IsModifiable": true,
      "MinimumEngineVersion": "5.0.0",
      "ChangeType": "immediate"
    },
    {
      "ParameterName": "hll-sparse-max-bytes",
      "ParameterValue": "3000",
      "Description": "HyperLogLog sparse representation bytes limit",
      "Source": "system",
      "DataType": "integer",
      "AllowedValues": "1-16000",
      "IsModifiable": true,
      "MinimumEngineVersion": "5.0.0",
      "ChangeType": "immediate"
    },
    {
      "ParameterName": "lazyfree-lazy-eviction",
      "ParameterValue": "no",
      "Description": "Perform an asynchronous delete on evictions",
      "Source": "system",
      "DataType": "string",
      "AllowedValues": "yes,no",
      "IsModifiable": true,
```

```
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "lazyfree-lazy-expire",
    "ParameterValue": "no",
    "Description": "Perform an asynchronous delete on expired keys",
    "Source": "system",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "lazyfree-lazy-server-del",
    "ParameterValue": "no",
    "Description": "Perform an asynchronous delete on key updates",
    "Source": "system",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "lfu-decay-time",
    "ParameterValue": "1",
    "Description": "The amount of time in minutes to decrement the key
counter for LFU eviction policy",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "lfu-log-factor",
    "ParameterValue": "10",
    "Description": "The log factor for incrementing key counter for LFU
eviction policy",
    "Source": "system",
    "DataType": "integer",
```

```

    "AllowedValues": "1-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "list-compress-depth",
    "ParameterValue": "0",
    "Description": "Number of quicklist ziplist nodes from each side
of the list to exclude from compression. The head and tail of the list are always
uncompressed for fast push/pop operations",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "list-max-ziplist-size",
    "ParameterValue": "-2",
    "Description": "The number of entries allowed per internal list node
can be specified as a fixed maximum size or a maximum number of elements",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "-5,-4,-3,-2,-1,1-",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "lua-replicate-commands",
    "ParameterValue": "yes",
    "Description": "Always enable Lua effect replication or not",
    "Source": "system",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "lua-time-limit",
    "ParameterValue": "5000",

```

```

        "Description": "Max execution time of a Lua script in milliseconds.
0 for unlimited execution without warnings.",
        "Source": "system",
        "DataType": "integer",
        "AllowedValues": "5000",
        "IsModifiable": false,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "immediate"
    },
    {
        "ParameterName": "maxclients",
        "ParameterValue": "65000",
        "Description": "The maximum number of Redis clients.",
        "Source": "system",
        "DataType": "integer",
        "AllowedValues": "1-65000",
        "IsModifiable": false,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "requires-reboot"
    },
    {
        "ParameterName": "maxmemory-policy",
        "ParameterValue": "volatile-lru",
        "Description": "Max memory policy.",
        "Source": "system",
        "DataType": "string",
        "AllowedValues": "volatile-lru,allkeys-lru,volatile-lfu,allkeys-
lfu,volatile-random,allkeys-random,volatile-ttl,noeviction",
        "IsModifiable": true,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "immediate"
    },
    {
        "ParameterName": "maxmemory-samples",
        "ParameterValue": "3",
        "Description": "Max memory samples.",
        "Source": "system",
        "DataType": "integer",
        "AllowedValues": "1-",
        "IsModifiable": true,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "immediate"
    },
    {

```

```
    "ParameterName": "min-replicas-max-lag",
    "ParameterValue": "10",
    "Description": "The maximum amount of replica lag in seconds beyond
which the master would stop taking writes. A value of 0 means the master always
takes writes.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "min-replicas-to-write",
    "ParameterValue": "0",
    "Description": "The minimum number of replicas that must be present
with lag no greater than min-replicas-max-lag for master to take writes. Setting
this to 0 means the master always takes writes.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "notify-keyspace-events",
    "Description": "The keyspace events for Redis to notify Pub/Sub
clients about. By default all notifications are disabled",
    "Source": "system",
    "DataType": "string",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "proto-max-bulk-len",
    "ParameterValue": "536870912",
    "Description": "Max size of a single element request",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "1048576-536870912",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
```

```

    "ChangeType": "immediate"
  },
  {
    "ParameterName": "rename-commands",
    "ParameterValue": "",
    "Description": "Redis commands that can be dynamically renamed by
the customer",
    "Source": "system",
    "DataType": "string",
    "AllowedValues":
"APPEND,BITCOUNT,BITFIELD,BITOP,BITPOS,BLPOP,BRPOP,BRPOPLUSH,BZPOPMIN,BZPOPMAX,CLIENT,COMM
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.3",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "repl-backlog-size",
    "ParameterValue": "1048576",
    "Description": "The replication backlog size in bytes for PSYNC.
This is the size of the buffer which accumulates slave data when slave is
disconnected for some time, so that when slave reconnects again, only transfer the
portion of data which the slave missed. Minimum value is 16K.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "16384-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "repl-backlog-ttl",
    "ParameterValue": "3600",
    "Description": "The amount of time in seconds after the master no
longer have any slaves connected for the master to free the replication backlog. A
value of 0 means to never release the backlog.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "replica-allow-chaining",

```



```

    "ParameterValue": "no",
    "Description": "Configures if chaining of replicas is allowed",
    "Source": "system",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "replica-ignore-maxmemory",
    "ParameterValue": "yes",
    "Description": "Determines if replica ignores maxmemory setting by
not evicting items independent from the master",
    "Source": "system",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "replica-lazy-flush",
    "ParameterValue": "no",
    "Description": "Perform an asynchronous flushDB during replica
sync",
    "Source": "system",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "IsModifiable": false,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "reserved-memory-percent",
    "ParameterValue": "25",
    "Description": "The percent of memory reserved for non-cache memory
usage. You may want to increase this parameter for nodes with read replicas, AOF
enabled, etc, to reduce swap usage.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-100",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",

```

```
        "ChangeType": "immediate"
    },
    {
        "ParameterName": "set-max-intset-entries",
        "ParameterValue": "512",
        "Description": "The limit in the size of the set in order for the
dataset to be compressed.",
        "Source": "system",
        "DataType": "integer",
        "AllowedValues": "0-",
        "IsModifiable": true,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "immediate"
    },
    {
        "ParameterName": "slowlog-log-slower-than",
        "ParameterValue": "10000",
        "Description": "The execution time, in microseconds, to exceed in
order for the command to get logged. Note that a negative number disables the slow
log, while a value of zero forces the logging of every command.",
        "Source": "system",
        "DataType": "integer",
        "AllowedValues": "-",
        "IsModifiable": true,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "immediate"
    },
    {
        "ParameterName": "slowlog-max-len",
        "ParameterValue": "128",
        "Description": "The length of the slow log. There is no limit to
this length. Just be aware that it will consume memory. You can reclaim memory used
by the slow log with SLOWLOG RESET.",
        "Source": "system",
        "DataType": "integer",
        "AllowedValues": "0-",
        "IsModifiable": true,
        "MinimumEngineVersion": "5.0.0",
        "ChangeType": "immediate"
    },
    {
        "ParameterName": "stream-node-max-bytes",
        "ParameterValue": "4096",
```

```

    "Description": "The maximum size of a single node in a stream in
bytes",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "stream-node-max-entries",
    "ParameterValue": "100",
    "Description": "The maximum number of items a single node in a
stream can contain",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "tcp-keepalive",
    "ParameterValue": "300",
    "Description": "If non-zero, send ACKs every given number of
seconds.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  },
  {
    "ParameterName": "timeout",
    "ParameterValue": "0",
    "Description": "Close connection if client is idle for a given
number of seconds, or never if 0.",
    "Source": "system",
    "DataType": "integer",
    "AllowedValues": "0,20-",
    "IsModifiable": true,
    "MinimumEngineVersion": "5.0.0",
    "ChangeType": "immediate"
  }
}

```

```

    },
    {
      "ParameterName": "zset-max-ziplist-entries",
      "ParameterValue": "128",
      "Description": "The maximum number of sorted set entries in order
for the dataset to be compressed.",
      "Source": "system",
      "DataType": "integer",
      "AllowedValues": "0-",
      "IsModifiable": true,
      "MinimumEngineVersion": "5.0.0",
      "ChangeType": "immediate"
    },
    {
      "ParameterName": "zset-max-ziplist-value",
      "ParameterValue": "64",
      "Description": "The threshold of biggest sorted set entries in order
for the dataset to be compressed.",
      "Source": "system",
      "DataType": "integer",
      "AllowedValues": "0-",
      "IsModifiable": true,
      "MinimumEngineVersion": "5.0.0",
      "ChangeType": "immediate"
    }
  ]
}
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEngineDefaultParameters](#)中的。

## describe-events

以下代码示例演示了如何使用 describe-events。

### AWS CLI

描述复制组的事件

以下describe-events示例返回复制组的事件列表。

```

aws elasticache describe-events \
  --source-identifier test-cluster \

```

```
--source-type replication-group
```

输出：

```
{
  "Events": [
    {
      "SourceIdentifier": "test-cluster",
      "SourceType": "replication-group",
      "Message": "Automatic failover has been turned on for replication group
test-cluster",
      "Date": "2020-03-18T23:51:34.457Z"
    },
    {
      "SourceIdentifier": "test-cluster",
      "SourceType": "replication-group",
      "Message": "Replication group test-cluster created",
      "Date": "2020-03-18T23:50:31.378Z"
    }
  ]
}
```

有关更多信息，请参阅《Elasticache 用户指南》中的[监控事件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEvents](#)中的。

## describe-global-replication-groups

以下代码示例演示了如何使用 describe-global-replication-groups。

AWS CLI

描述全局复制组

以下describe-global-replication-groups示例返回全局数据存储的详细信息。

```
aws elasticache describe-global-replication-groups \
  --global-replication-group-id my-grg
```

输出：

```
{
  "GlobalReplicationGroups": [
```

```
{
  "GlobalReplicationGroupId": "my-grg",
  "GlobalReplicationGroupDescription": "my-grg",
  "Status": "creating",
  "CacheNodeType": "cache.r5.large",
  "Engine": "redis",
  "EngineVersion": "5.0.6",
  "ClusterEnabled": false,
  "AuthTokenEnabled": false,
  "TransitEncryptionEnabled": false,
  "AtRestEncryptionEnabled": false
}
```

有关更多信息，请参阅《Elasticache 用户指南》中的[使用全球数据存储跨 AWS 区域复制](#)。

• 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeGlobalReplicationGroups](#) 中的。

## describe-replication-groups

以下代码示例演示了如何使用 describe-replication-groups。

### AWS CLI

返回复制组详细信息列表

以下 describe-replication-groups 示例返回复制组。

```
aws elasticache describe-replication-groups
```

输出：

```
{
  "ReplicationGroups": [
    {
      "ReplicationGroupId": "my-cluster",
      "Description": "mycluster",
      "Status": "available",
      "PendingModifiedValues": {},
      "MemberClusters": [
        "pat-cluster-001",
        "pat-cluster-002",
```

```
        "pat-cluster-003",
        "pat-cluster-004"
    ],
    "NodeGroups": [
        {
            "NodeGroupId": "0001",
            "Status": "available",
            "PrimaryEndpoint": {
                "Address": "my-
cluster.xxxxih.ng.0001.usw2.cache.amazonaws.com",
                "Port": 6379
            },
            "ReaderEndpoint": {
                "Address": "my-cluster-
ro.xxxxih.ng.0001.usw2.cache.amazonaws.com",
                "Port": 6379
            },
            "NodeGroupMembers": [
                {
                    "CacheClusterId": "my-cluster-001",
                    "CacheNodeId": "0001",
                    "ReadEndpoint": {
                        "Address": "pat-
cluster-001.xxxxih.0001.usw2.cache.amazonaws.com",
                        "Port": 6379
                    },
                    "PreferredAvailabilityZone": "us-west-2a",
                    "CurrentRole": "primary"
                },
                {
                    "CacheClusterId": "my-cluster-002",
                    "CacheNodeId": "0001",
                    "ReadEndpoint": {
                        "Address": "pat-
cluster-002.xxxxih.0001.usw2.cache.amazonaws.com",
                        "Port": 6379
                    },
                    "PreferredAvailabilityZone": "us-west-2a",
                    "CurrentRole": "replica"
                },
                {
                    "CacheClusterId": "my-cluster-003",
                    "CacheNodeId": "0001",
                    "ReadEndpoint": {
```

```

        "Address": "pat-
cluster-003.xxxxih.0001.usw2.cache.amazonaws.com",
        "Port": 6379
    },
    "PreferredAvailabilityZone": "us-west-2a",
    "CurrentRole": "replica"
},
{
    "CacheClusterId": "my-cluster-004",
    "CacheNodeId": "0001",
    "ReadEndpoint": {
        "Address": "pat-
cluster-004.xxxxih.0001.usw2.cache.amazonaws.com",
        "Port": 6379
    },
    "PreferredAvailabilityZone": "us-west-2a",
    "CurrentRole": "replica"
}
]
}
],
"AutomaticFailover": "disabled",
"SnapshotRetentionLimit": 0,
"SnapshotWindow": "07:30-08:30",
"ClusterEnabled": false,
"CacheNodeType": "cache.r5.xlarge",
"AuthTokenEnabled": false,
"TransitEncryptionEnabled": false,
"AtRestEncryptionEnabled": false,
"ARN": "arn:aws:elasticache:us-
west-2:xxxxxxxxxxxx152:replicationgroup:my-cluster",
"LogDeliveryConfigurations": [
    {
        "LogType": "slow-log",
        "DestinationType": "cloudwatch-logs",
        "DestinationDetails": {
            "CloudWatchLogsDetails": {
                "LogGroup": "test-log"
            }
        },
        "LogFormat": "json",
        "Status": "active"
    }
]

```



```
    }  
  ]  
}
```

有关更多信息，请参阅《Elasticache 用户指南》中的[管理集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeReplicationGroups](#)中的。

## describe-reserved-cache-nodes-offerings

以下代码示例演示了如何使用 `describe-reserved-cache-nodes-offerings`。

### AWS CLI

为了描述 `reserved-cache-nodes-offerings`

以下`describe-reserved-cache-nodes-offerings`示例返回 `reserved-cache-node` 选项的详细信息。

```
aws elasticache describe-reserved-cache-nodes-offerings
```

输出：

```
{  
  "ReservedCacheNodesOfferings": [  
    {  
      "ReservedCacheNodesOfferingId": "01ce0a19-a476-41cb-8aee-48eacbc8e5",  
      "CacheNodeType": "cache.t3.small",  
      "Duration": 31536000,  
      "FixedPrice": 97.0,  
      "UsagePrice": 0.0,  
      "ProductDescription": "memcached",  
      "OfferingType": "Partial Upfront",  
      "RecurringCharges": [  
        {  
          "RecurringChargeAmount": 0.011,  
          "RecurringChargeFrequency": "Hourly"  
        }  
      ]  
    },  
    {  
      "ReservedCacheNodesOfferingId": "0443a27b-4da5-4b90-b92d-929fbd7abed2",
```

```

    "CacheNodeType": "cache.m3.2xlarge",
    "Duration": 31536000,
    "FixedPrice": 1772.0,
    "UsagePrice": 0.0,
    "ProductDescription": "redis",
    "OfferingType": "Heavy Utilization",
    "RecurringCharges": [
      {
        "RecurringChargeAmount": 0.25,
        "RecurringChargeFrequency": "Hourly"
      }
    ],
    ...
  ]
}

```

有关更多信息，请参阅《Elasticache Redis 用户指南》中的“[获取有关预留节点产品的信息](#)”或《Elasticache Mem c ached 用户指南》中的“[获取有关预留节点产品的信息](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeReservedCacheNodesOfferings](#) 中的。

## describe-reserved-cache-nodes

以下代码示例演示了如何使用 describe-reserved-cache-nodes。

### AWS CLI

#### 描述预留缓存节点

以下 describe-reserved-cache-nodes 示例返回有关此账户的预留缓存节点或有关指定预留缓存节点的信息。

```
aws 弹性疼痛 describe-reserved-cache-nodes
```

输出：

```

{
  "ReservedCacheNodes": [
    {

```

```

    "ReservedCacheNodeId": "mynode",
    "ReservedCacheNodesOfferingId": "xxxxxxxx-xxxxx-xxxxx-xxxx-xxxxxxxx71",
    "CacheNodeType": "cache.t3.small",
    "StartTime": "2019-12-06T02:50:44.003Z",
    "Duration": 31536000,
    "FixedPrice": 0.0,
    "UsagePrice": 0.0,
    "CacheNodeCount": 1,
    "ProductDescription": "redis",
    "OfferingType": "No Upfront",
    "State": "payment-pending",
    "RecurringCharges": [
      {
        "RecurringChargeAmount": 0.023,
        "RecurringChargeFrequency": "Hourly"
      }
    ],
    "ReservationARN": "arn:aws:elasticache:us-
west-2:xxxxxxxxxxxx52:reserved-instance:mynode"
  }
]
}

```

有关更多信息，请参阅 Elasticache 用户[指南中的使用预留节点管理成本](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeReservedCacheNodes](#)中的。

## describe-service-updates

以下代码示例演示了如何使用 describe-service-updates。

### AWS CLI

#### 描述服务更新

以下 describe-service-updates 示例返回有关服务更新的详细信息。

```
aws elasticache describe-service-updates
```

输出：

```
{
  "ServiceUpdates": [
```

```

    {
      "ServiceUpdateName": "elc-xxxxxxx7-001",
      "ServiceUpdateReleaseDate": "2019-10-09T16:00:00Z",
      "ServiceUpdateEndDate": "2020-02-09T15:59:59Z",
      "ServiceUpdateSeverity": "important",
      "ServiceUpdateRecommendedApplyByDate": "2019-11-08T15:59:59Z",
      "ServiceUpdateStatus": "available",
      "ServiceUpdateDescription": "Upgrades to improve the security,
reliability, and operational performance of your ElastiCache nodes",
      "ServiceUpdateType": "security-update",
      "Engine": "redis, memcached",
      "EngineVersion": "redis 2.6.13 and onwards, memcached 1.4.5 and
onwards",
      "AutoUpdateAfterRecommendedApplyByDate": false,
      "EstimatedUpdateTime": "30 minutes per node"
    },
    {
      "ServiceUpdateName": "elc-xxxxxxx4-001",
      "ServiceUpdateReleaseDate": "2019-06-11T15:00:00Z",
      "ServiceUpdateEndDate": "2019-10-01T09:24:00Z",
      "ServiceUpdateSeverity": "important",
      "ServiceUpdateRecommendedApplyByDate": "2019-07-11T14:59:59Z",
      "ServiceUpdateStatus": "expired",
      "ServiceUpdateDescription": "Upgrades to improve the security,
reliability, and operational performance of your ElastiCache nodes",
      "ServiceUpdateType": "security-update",
      "Engine": "redis",
      "EngineVersion": "redis 3.2.6, redis 4.0 and onwards",
      "AutoUpdateAfterRecommendedApplyByDate": false,
      "EstimatedUpdateTime": "30 minutes per node"
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeServiceUpdates](#)中的。

## describe-snapshots

以下代码示例演示了如何使用 describe-snapshots。

### AWS CLI

#### 描述快照

以下 “describe-snapshots” 示例返回有关您的集群或复制组快照的信息。

```
aws elasticache describe-snapshots
```

输出：

```
{
  "Snapshots": [
    {
      "SnapshotName": "automatic.my-cluster2-002-2019-12-05-06-38",
      "CacheClusterId": "my-cluster2-002",
      "SnapshotStatus": "available",
      "SnapshotSource": "automated",
      "CacheNodeType": "cache.r5.large",
      "Engine": "redis",
      "EngineVersion": "5.0.5",
      "NumCacheNodes": 1,
      "PreferredAvailabilityZone": "us-west-2a",
      "CacheClusterCreateTime": "2019-11-26T01:22:52.396Z",
      "PreferredMaintenanceWindow": "mon:17:30-mon:18:30",
      "TopicArn": "arn:aws:sns:us-west-2:xxxxxxx52:My_Topic",
      "Port": 6379,
      "CacheParameterGroupName": "default.redis5.0",
      "CacheSubnetGroupName": "kxkxk",
      "VpcId": "vpc-a3e97cdb",
      "AutoMinorVersionUpgrade": true,
      "SnapshotRetentionLimit": 1,
      "SnapshotWindow": "06:30-07:30",
      "NodeSnapshots": [
        {
          "CacheNodeId": "0001",
          "CacheSize": "5 MB",
          "CacheNodeCreateTime": "2019-11-26T01:22:52.396Z",
          "SnapshotCreateTime": "2019-12-05T06:38:23Z"
        }
      ]
    },
    {
      "SnapshotName": "myreplica-backup",
      "CacheClusterId": "myreplica",
      "SnapshotStatus": "available",
      "SnapshotSource": "manual",
      "CacheNodeType": "cache.r5.large",
    }
  ]
}
```

```
"Engine": "redis",
"EngineVersion": "5.0.5",
"NumCacheNodes": 1,
"PreferredAvailabilityZone": "us-west-2a",
"CacheClusterCreateTime": "2019-11-26T00:14:52.439Z",
"PreferredMaintenanceWindow": "sat:10:00-sat:11:00",
"TopicArn": "arn:aws:sns:us-west-2:xxxxxxxxxx152:My_Topic",
"Port": 6379,
"CacheParameterGroupName": "default.redis5.0",
"CacheSubnetGroupName": "kxkxk",
"VpcId": "vpc-a3e97cdb",
"AutoMinorVersionUpgrade": true,
"SnapshotRetentionLimit": 0,
"SnapshotWindow": "09:00-10:00",
"NodeSnapshots": [
  {
    "CacheNodeId": "0001",
    "CacheSize": "5 MB",
    "CacheNodeCreateTime": "2019-11-26T00:14:52.439Z",
    "SnapshotCreateTime": "2019-11-26T00:25:01Z"
  }
]
},
{
  "SnapshotName": "my-cluster",
  "CacheClusterId": "my-cluster-003",
  "SnapshotStatus": "available",
  "SnapshotSource": "manual",
  "CacheNodeType": "cache.r5.large",
  "Engine": "redis",
  "EngineVersion": "5.0.5",
  "NumCacheNodes": 1,
  "PreferredAvailabilityZone": "us-west-2a",
  "CacheClusterCreateTime": "2019-11-25T23:56:17.186Z",
  "PreferredMaintenanceWindow": "sat:10:00-sat:11:00",
  "TopicArn": "arn:aws:sns:us-west-2:xxxxxxxxxx152:My_Topic",
  "Port": 6379,
  "CacheParameterGroupName": "default.redis5.0",
  "CacheSubnetGroupName": "kxkxk",
  "VpcId": "vpc-a3e97cdb",
  "AutoMinorVersionUpgrade": true,
  "SnapshotRetentionLimit": 0,
  "SnapshotWindow": "09:00-10:00",
  "NodeSnapshots": [
```

```

    {
      "CacheNodeId": "0001",
      "CacheSize": "5 MB",
      "CacheNodeCreateTime": "2019-11-25T23:56:17.186Z",
      "SnapshotCreateTime": "2019-11-26T03:08:33Z"
    }
  ]
}

```

有关更多信息，请参阅《Elasticache [用户 ElastiCache 指南](#)》中的 [Redis 备份和恢复](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeSnapshots](#) 中的。

## describe-update-actions

以下代码示例演示了如何使用 describe-update-actions。

### AWS CLI

#### 描述更新操作

以下 describe-update-actions 示例返回更新操作的详细信息。

```
aws elasticache describe-update-actions
```

输出：

```

{
  "UpdateActions": [
    {
      "ReplicationGroupId": "mycluster",
      "ServiceUpdateName": "elc-20191007-001",
      "ServiceUpdateReleaseDate": "2019-10-09T16:00:00Z",
      "ServiceUpdateSeverity": "important",
      "ServiceUpdateStatus": "available",
      "ServiceUpdateRecommendedApplyByDate": "2019-11-08T15:59:59Z",
      "ServiceUpdateType": "security-update",
      "UpdateActionAvailableDate": "2019-12-05T19:15:19.995Z",
      "UpdateActionStatus": "complete",
      "NodesUpdated": "9/9",
      "UpdateActionStatusModifiedDate": "2019-12-05T19:15:20.461Z",
    }
  ]
}

```

```
    "SlaMet": "n/a",
    "Engine": "redis"
  },
  {
    "CacheClusterId": "my-memcached-cluster",
    "ServiceUpdateName": "elc-20191007-001",
    "ServiceUpdateReleaseDate": "2019-10-09T16:00:00Z",
    "ServiceUpdateSeverity": "important",
    "ServiceUpdateStatus": "available",
    "ServiceUpdateRecommendedApplyByDate": "2019-11-08T15:59:59Z",
    "ServiceUpdateType": "security-update",
    "UpdateActionAvailableDate": "2019-12-04T18:26:05.349Z",
    "UpdateActionStatus": "complete",
    "NodesUpdated": "1/1",
    "UpdateActionStatusModifiedDate": "2019-12-04T18:26:05.352Z",
    "SlaMet": "n/a",
    "Engine": "redis"
  },
  {
    "ReplicationGroupId": "my-cluster",
    "ServiceUpdateName": "elc-20191007-001",
    "ServiceUpdateReleaseDate": "2019-10-09T16:00:00Z",
    "ServiceUpdateSeverity": "important",
    "ServiceUpdateStatus": "available",
    "ServiceUpdateRecommendedApplyByDate": "2019-11-08T15:59:59Z",
    "ServiceUpdateType": "security-update",
    "UpdateActionAvailableDate": "2019-11-26T03:36:26.320Z",
    "UpdateActionStatus": "complete",
    "NodesUpdated": "4/4",
    "UpdateActionStatusModifiedDate": "2019-12-04T22:11:12.664Z",
    "SlaMet": "n/a",
    "Engine": "redis"
  },
  {
    "ReplicationGroupId": "my-cluster2",
    "ServiceUpdateName": "elc-20191007-001",
    "ServiceUpdateReleaseDate": "2019-10-09T16:00:00Z",
    "ServiceUpdateSeverity": "important",
    "ServiceUpdateStatus": "available",
    "ServiceUpdateRecommendedApplyByDate": "2019-11-08T15:59:59Z",
    "ServiceUpdateType": "security-update",
    "UpdateActionAvailableDate": "2019-11-26T01:26:01.617Z",
    "UpdateActionStatus": "complete",
    "NodesUpdated": "3/3",
```



```
        "UpdateActionStatusModifiedDate": "2019-11-26T01:26:01.753Z",
        "SlaMet": "n/a",
        "Engine": "redis"
    }
]
}
```

有关更多信息，请参阅 Elasticache 用户指南 [ElastiCache 中的亚马逊自助服务更新](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeUpdateActions](#) 中的。

## describe-user-groups

以下代码示例演示了如何使用 describe-user-groups。

### AWS CLI

描述用户组

以下 describe-user-groups 示例返回用户组列表。

```
aws elasticache describe-user-groups
```

输出：

```
{
  "UserGroups": [
    {
      "UserGroupId": "myusergroup",
      "Status": "active",
      "Engine": "redis",
      "UserIds": [
        "default"
      ],
      "ReplicationGroups": [],
      "ARN": "arn:aws:elasticache:us-west-2:xxxxxxxxxx52:usergroup:myusergroup"
    }
  ]
}
```

有关更多信息，请参阅《Elasticache [用户指南](#)》中的使用基于角色的访问控制 (RBAC) 对用户进行[身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeUserGroups](#)中的。

## describe-users

以下代码示例演示了如何使用 describe-users。

### AWS CLI

#### 描述用户

以下describe-users示例返回用户列表。

```
aws elasticache describe-users
```

输出：

```
{
  "Users": [
    {
      "UserId": "default",
      "UserName": "default",
      "Status": "active",
      "Engine": "redis",
      "AccessString": "on ~* +@all",
      "UserGroupIds": [
        "myusergroup"
      ],
      "Authentication": {
        "Type": "no-password"
      },
      "ARN": "arn:aws:elasticache:us-west-2:xxxxxxxxxx52:user:default"
    },
    {
      "UserId": "user1",
      "UserName": "myUser",
      "Status": "active",
      "Engine": "redis",
      "AccessString": "on ~* +@all",
      "UserGroupIds": [],
      "Authentication": {
        "Type": "password",
        "PasswordCount": 1
      }
    }
  ]
}
```

```

    },
    "ARN": "arn:aws:elasticache:us-west-2:xxxxxxxxxx52:user:user1"
  },
  {
    "UserId": "user2",
    "UserName": "myUser",
    "Status": "active",
    "Engine": "redis",
    "AccessString": "on ~app:* -@all +@read +@hash +@bitmap +@geo -setbit -
bitfield -hset -hsetnx -hmset -hincrby -hincrbyfloat -hdel -bitop -geoadd -georadius
-georadiusbymember",
    "UserGroupIds": [],
    "Authentication": {
      "Type": "password",
      "PasswordCount": 1
    },
    "ARN": "arn:aws:elasticache:us-west-2:xxxxxxxxxx52:user:user2"
  }
]
}

```

有关更多信息，请参阅《Elasticache [用户指南](#)》中的[使用基于角色的访问控制 \(RBAC\) 对用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeUsers](#)中的。

## disassociate-global-replication-group

以下代码示例演示了如何使用 disassociate-global-replication-group。

### AWS CLI

取消辅助群集与全局复制组的关联

以下disassociate-global-replication-group示例从全局数据存储中移除辅助群集

```

aws elasticache disassociate-global-replication-group \
  --global-replication-group-id my-grg \
  --replication-group-id my-cluster-grg-secondary \
  --replication-group-region us-east-1

```

输出：

```
{
  "GlobalReplicationGroup": {
    "GlobalReplicationGroupId": "my-grg",
    "GlobalReplicationGroupDescription": "my-grg",
    "Status": "modifying",
    "CacheNodeType": "cache.r5.large",
    "Engine": "redis",
    "EngineVersion": "5.0.6",
    "Members": [
      {
        "ReplicationGroupId": "my-cluster-grg-secondary",
        "ReplicationGroupRegion": "us-east-1",
        "Role": "SECONDARY",
        "AutomaticFailover": "enabled",
        "Status": "associated"
      },
      {
        "ReplicationGroupId": "my-cluster-grg",
        "ReplicationGroupRegion": "us-west-2",
        "Role": "PRIMARY",
        "AutomaticFailover": "enabled",
        "Status": "associated"
      }
    ],
    "ClusterEnabled": false,
    "AuthTokenEnabled": false,
    "TransitEncryptionEnabled": false,
    "AtRestEncryptionEnabled": false
  }
}
```

有关更多信息，请参阅《Elasticache 用户指南》中的[使用全球数据存储跨 AWS 区域复制](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateGlobalReplicationGroup](#) 中的。

## increase-node-groups-in-global-replication-group

以下代码示例演示了如何使用 `increase-node-groups-in-global-replication-group`。

### AWS CLI

#### 增加全局复制组中的节点组数量

以下内容使用 Redis 引擎 `increase-node-groups-in-global-replication-group` 增加了节点组数量。

```
aws elasticache increase-node-groups-in-global-replication-group \  
  --global-replication-group-id sgau-pat-test-4 \  
  --node-group-count 6 \  
  --apply-immediately
```

输出：

```
{  
  "GlobalReplicationGroup": {  
    "GlobalReplicationGroupId": "sgau-test-4",  
    "GlobalReplicationGroupDescription": "test-4",  
    "Status": "modifying",  
    "CacheNodeType": "cache.r5.large",  
    "Engine": "redis",  
    "EngineVersion": "5.0.6",  
    "Members": [  
      {  
        "ReplicationGroupId": "my-cluster-b",  
        "ReplicationGroupRegion": "us-east-1",  
        "Role": "SECONDARY",  
        "AutomaticFailover": "enabled",  
        "Status": "associated"  
      },  
      {  
        "ReplicationGroupId": "my-cluster-a",  
        "ReplicationGroupRegion": "us-west-2",  
        "Role": "PRIMARY",  
        "AutomaticFailover": "enabled",  
        "Status": "associated"  
      }  
    ],  
    "ClusterEnabled": true,  
    "GlobalNodeGroups": [  
      {  
        "GlobalNodeGroupId": "sgau-test-4-0001",  
        "Slots": "0-234,2420-5461"  
      },  
      {  
        "GlobalNodeGroupId": "sgau-test-4-0002",  
        "Slots": "5462-5904,6997-9830"  
      }  
    ]  
  }  
}
```

```
    },
    {
      "GlobalNodeGroupId": "sgau-test-4-0003",
      "Slots": "10923-11190,13375-16383"
    },
    {
      "GlobalNodeGroupId": "sgau-test-4-0004",
      "Slots": "235-2419,5905-6996"
    },
    {
      "GlobalNodeGroupId": "sgau-test-4-0005",
      "Slots": "9831-10922,11191-13374"
    }
  ],
  "AuthTokenEnabled": false,
  "TransitEncryptionEnabled": false,
  "AtRestEncryptionEnabled": false
}
```

有关更多信息，请参阅《Elasticache 用户指南》中的[使用全球数据存储跨 AWS 区域复制](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [IncreaseNodeGroupsInGlobalReplicationGroup](#) 中的。

## increase-replica-count

以下代码示例演示了如何使用 `increase-replica-count`。

### AWS CLI

#### 增加副本数量

以下 `increase-replica-count` 示例做了两件事之一。它可以动态增加 Redis ( 已禁用集群模式 ) 复制组中的副本数量。或者，它可以动态增加 Redis ( 已启用集群模式 ) 复制组的一个或多个节点组 ( 分片 ) 中的副本节点数量。此操作是在没有集群停机时间的情况下执行的。

```
aws elasticache increase-replica-count \
  --replication-group-id "my-cluster" \
  --apply-immediately \
  --new-replica-count 3
```

输出：

```
{
  "ReplicationGroup": {
    "ReplicationGroupId": "my-cluster",
    "Description": " ",
    "Status": "modifying",
    "PendingModifiedValues": {},
    "MemberClusters": [
      "my-cluster-001",
      "my-cluster-002",
      "my-cluster-003",
      "my-cluster-004"
    ],
    "NodeGroups": [
      {
        "NodeGroupId": "0001",
        "Status": "modifying",
        "PrimaryEndpoint": {
          "Address": "my-
cluster.xxxxxih.ng.0001.usw2.cache.amazonaws.com",
          "Port": 6379
        },
        "ReaderEndpoint": {
          "Address": "my-cluster-
ro.xxxxxih.ng.0001.usw2.cache.amazonaws.com",
          "Port": 6379
        },
        "NodeGroupMembers": [
          {
            "CacheClusterId": "my-cluster-001",
            "CacheNodeId": "0001",
            "ReadEndpoint": {
              "Address": "my-
cluster-001.xxxxxih.0001.usw2.cache.amazonaws.com",
              "Port": 6379
            },
            "PreferredAvailabilityZone": "us-west-2a",
            "CurrentRole": "primary"
          },
          {
            "CacheClusterId": "my-cluster-003",
            "CacheNodeId": "0001",
            "ReadEndpoint": {
```

```

        "Address": "my-
cluster-003.xxxxxih.0001.usw2.cache.amazonaws.com",
        "Port": 6379
    },
    "PreferredAvailabilityZone": "us-west-2a",
    "CurrentRole": "replica"
}
]
}
],
"AutomaticFailover": "disabled",
"SnapshotRetentionLimit": 0,
"SnapshotWindow": "07:30-08:30",
"ClusterEnabled": false,
"CacheNodeType": "cache.r5.xlarge",
"TransitEncryptionEnabled": false,
"AtRestEncryptionEnabled": false
}
}

```

有关更多信息，请参阅 [Elasticache 用户指南中的增加分片中的副本数量](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [IncreaseReplicaCount](#) 中的。

## list-allowed-node-type-modifications

以下代码示例演示了如何使用 `list-allowed-node-type-modifications`。

### AWS CLI

列出允许的节点修改

以下 `list-allowed-node-type-modifications` 示例列出了您可以将 Redis 集群或复制组的当前节点类型扩展到的所有可用节点类型。

```
aws elasticache list-allowed-node-type-modifications \
  --replication-group-id "my-replication-group"
```

输出：

```
{
  "ScaleUpModifications": [
    "cache.m5.12xlarge",
```



```
    "cache.m5.24xlarge",
    "cache.m5.4xlarge",
    "cache.r5.12xlarge",
    "cache.r5.24xlarge",
    "cache.r5.2xlarge",
    "cache.r5.4xlarge"
  ],
  "ScaleDownModifications": [
    "cache.m3.large",
    "cache.m3.medium",
    "cache.m3.xlarge",
    "cache.m4.large",
    "cache.m4.xlarge",
    "cache.m5.2xlarge",
    "cache.m5.large",
    "cache.m5.xlarge",
    "cache.r3.large",
    "cache.r4.large",
    "cache.r4.xlarge",
    "cache.r5.large",
    "cache.t2.medium",
    "cache.t2.micro",
    "cache.t2.small",
    "cache.t3.medium",
    "cache.t3.micro",
    "cache.t3.small"
  ]
}
```

有关更多信息，请参阅 Elasticache 用户 [指南中的扩展 ElastiCache Redis 集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAllowedNodeTypeModifications](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出资源的标签

以下 `list-tags-for-resource` 示例列出了资源的标签。

```
aws elasticache list-tags-for-resource \
```

```
--resource-name "arn:aws:elasticache:us-east-1:123456789012:cluster:my-cluster"
```

输出：

```
{
  "TagList": [
    {
      "Key": "Project",
      "Value": "querySpeedUp"
    },
    {
      "Key": "Environment",
      "Value": "PROD"
    }
  ]
}
```

有关更多信息，请参阅 Elasticache 用户 [AWS 指南中的使用 CLI 列出标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## modify-cache-cluster

以下代码示例演示了如何使用 modify-cache-cluster。

### AWS CLI

#### 修改缓存集群

以下 modify-cache-cluster 示例修改了指定集群的设置。

```
aws elasticache modify-cache-cluster \
  --cache-cluster-id "my-cluster" \
  --num-cache-nodes 1
```

输出：

```
{
  "CacheCluster": {
    "CacheClusterId": "my-cluster",
    "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
  }
}
```

```
"CacheNodeType": "cache.m5.large",
"Engine": "redis",
"EngineVersion": "5.0.5",
"CacheClusterStatus": "available",
"NumCacheNodes": 1,
"PreferredAvailabilityZone": "us-west-2c",
"CacheClusterCreateTime": "2019-12-04T18:24:56.652Z",
"PreferredMaintenanceWindow": "sat:10:00-sat:11:00",
"PendingModifiedValues": {},
"CacheSecurityGroups": [],
"CacheParameterGroup": {
    "CacheParameterGroupName": "default.redis5.0",
    "ParameterApplyStatus": "in-sync",
    "CacheNodeIdsToReboot": []
},
"CacheSubnetGroupName": "default",
"AutoMinorVersionUpgrade": true,
"SnapshotRetentionLimit": 0,
"SnapshotWindow": "07:00-08:00",
"TransitEncryptionEnabled": false,
"AtRestEncryptionEnabled": false
}
}
```

有关更多信息，请参阅 Elasticache 用户指南中的[修改 ElastiCache 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyCacheCluster](#)中的。

## modify-cache-parameter-group

以下代码示例演示了如何使用 modify-cache-parameter-group。

### AWS CLI

#### 修改缓存参数组

以下modify-cache-parameter-group示例修改了指定缓存参数组的参数。

```
aws elasticache modify-cache-parameter-group \
  --cache-parameter-group-name "mygroup" \
  --parameter-name-values "ParameterName=activedefrag, ParameterValue=no"
```

输出：

```
{
  "CacheParameterGroupName": "mygroup"
}
```

有关更多信息，请参阅 Elasticache 用户指南中的[修改参数组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyCacheParameterGroup](#)中的。

## modify-cache-subnet-group

以下代码示例演示了如何使用 modify-cache-subnet-group。

### AWS CLI

#### 修改缓存子网组

以下modify-cache-subnet-group示例修改了指定的缓存子网组。

```
aws elasticache modify-cache-subnet-group \
  --cache-subnet-group-name kxkxk \
  --cache-subnet-group-description "mygroup"
```

#### 输出：

```
{
  "CacheSubnetGroup": {
    "CacheSubnetGroupName": "kxkxk",
    "CacheSubnetGroupDescription": "mygroup",
    "VpcId": "vpc-xxxxcdb",
    "Subnets": [
      {
        "SubnetIdentifier": "subnet-xxxxbff",
        "SubnetAvailabilityZone": {
          "Name": "us-west-2a"
        }
      }
    ]
  }
}
```

有关更多信息，请参阅 Elasticache 用户指南中的[修改子网组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyCacheSubnetGroup](#)中的。

## modify-global-replication-group

以下代码示例演示了如何使用 modify-global-replication-group。

### AWS CLI

#### 修改全局复制组

以下内容使用 Redis 引擎modify-global-replication-group修改全局复制组的属性，在本例中为禁用自动故障转移。

```
aws elasticache modify-global-replication-group \  
  --global-replication-group-id sgau1-pat-group \  
  --apply-immediately \  
  --no-automatic-failover-enabled
```

#### 输出

```
{  
  "GlobalReplicationGroup": {  
    "GlobalReplicationGroupId": "sgau1-test-group",  
    "GlobalReplicationGroupDescription": " ",  
    "Status": "modifying",  
    "CacheNodeType": "cache.r5.large",  
    "Engine": "redis",  
    "EngineVersion": "5.0.6",  
    "ClusterEnabled": false,  
    "AuthTokenEnabled": false,  
    "TransitEncryptionEnabled": false,  
    "AtRestEncryptionEnabled": false  
  }  
}
```

有关更多信息，请参阅《Elasticache 用户指南》中的[使用全球数据存储跨 AWS 区域复制](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyGlobalReplicationGroup](#)中的。

## modify-replication-group-shard-configuration

以下代码示例演示了如何使用 modify-replication-group-shard-configuration。

## AWS CLI

### 修改复制组分片配置

以下方法使用 Redis 引擎 `modify-replication-group-shard-configuration` 减少了节点组数量。

```
aws elasticache modify-replication-group-shard-configuration \  
  --replication-group-id mycluster \  
  --node-group-count 3 \  
  --apply-immediately \  
  --node-groups-to-remove 0002
```

### 输出

```
{  
  "ReplicationGroup": {  
    "ReplicationGroupId": "mycluster",  
    "Description": "mycluster",  
    "GlobalReplicationGroupInfo": {},  
    "Status": "modifying",  
    "PendingModifiedValues": {},  
    "MemberClusters": [  
      "mycluster-0002-001",  
      "mycluster-0002-002",  
      "mycluster-0002-003",  
      "mycluster-0003-001",  
      "mycluster-0003-002",  
      "mycluster-0003-003",  
      "mycluster-0003-004",  
      "mycluster-0004-001",  
      "mycluster-0004-002",  
      "mycluster-0004-003",  
      "mycluster-0005-001",  
      "mycluster-0005-002",  
      "mycluster-0005-003"  
    ],  
    "NodeGroups": [  
      {  
        "NodeGroupId": "0002",  
        "Status": "modifying",  
        "Slots": "894-1767,3134-4443,5149-5461,6827-7332,12570-13662",  
        "NodeGroupMembers": [  

```

```
        {
            "CacheClusterId": "mycluster-0002-001",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2c"
        },
        {
            "CacheClusterId": "mycluster-0002-002",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2a"
        },
        {
            "CacheClusterId": "mycluster-0002-003",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2b"
        }
    ]
},
{
    "NodeGroupId": "0003",
    "Status": "modifying",
    "Slots":
"0-324,5462-5692,6784-6826,7698-8191,10923-11075,12441-12569,13663-16383",
    "NodeGroupMembers": [
        {
            "CacheClusterId": "mycluster-0003-001",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2c"
        },
        {
            "CacheClusterId": "mycluster-0003-002",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2b"
        },
        {
            "CacheClusterId": "mycluster-0003-003",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2a"
        },
        {
            "CacheClusterId": "mycluster-0003-004",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2c"
        }
    ]
}
```

```
    },
    {
      "NodeGroupId": "0004",
      "Status": "modifying",
      "Slots": "325-336,4706-5148,7333-7697,9012-10922,11076-12440",
      "NodeGroupMembers": [
        {
          "CacheClusterId": "mycluster-0004-001",
          "CacheNodeId": "0001",
          "PreferredAvailabilityZone": "us-west-2b"
        },
        {
          "CacheClusterId": "mycluster-0004-002",
          "CacheNodeId": "0001",
          "PreferredAvailabilityZone": "us-west-2a"
        },
        {
          "CacheClusterId": "mycluster-0004-003",
          "CacheNodeId": "0001",
          "PreferredAvailabilityZone": "us-west-2c"
        }
      ]
    },
    {
      "NodeGroupId": "0005",
      "Status": "modifying",
      "Slots": "337-893,1768-3133,4444-4705,5693-6783,8192-9011",
      "NodeGroupMembers": [
        {
          "CacheClusterId": "mycluster-0005-001",
          "CacheNodeId": "0001",
          "PreferredAvailabilityZone": "us-west-2a"
        },
        {
          "CacheClusterId": "mycluster-0005-002",
          "CacheNodeId": "0001",
          "PreferredAvailabilityZone": "us-west-2c"
        },
        {
          "CacheClusterId": "mycluster-0005-003",
          "CacheNodeId": "0001",
          "PreferredAvailabilityZone": "us-west-2b"
        }
      ]
    }
  ]
}
```



```
    }
  ],
  "AutomaticFailover": "enabled",
  "MultiAZ": "enabled",
  "ConfigurationEndpoint": {
    "Address": "mycluster.g2xbih.clustercfg.usw2.cache.amazonaws.com",
    "Port": 6379
  },
  "SnapshotRetentionLimit": 1,
  "SnapshotWindow": "13:00-14:00",
  "ClusterEnabled": true,
  "CacheNodeType": "cache.r5.xlarge",
  "TransitEncryptionEnabled": false,
  "AtRestEncryptionEnabled": false
}
}
```

有关更多信息，请参阅 Elasticache 用户 [指南中的扩展 ElastiCache Redis 集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyReplicationGroupShardConfiguration](#) 中的。

## modify-replication-group

以下代码示例演示了如何使用 modify-replication-group。

### AWS CLI

#### 修改复制组

以下内容 modify-replication-group 禁用使用 Redis 引擎的多可用区。

```
aws elasticache modify-replication-group \
  --replication-group-id test-cluster \
  --no-multi-az-enabled \
  --apply-immediately
```

#### 输出

```
{
  "ReplicationGroup": {
    "ReplicationGroupId": "test-cluster",
    "Description": "test-cluster",
```

```
"GlobalReplicationGroupInfo": {
  "GlobalReplicationGroupId": "sgaui-pat-group",
  "GlobalReplicationGroupMemberRole": "PRIMARY"
},
"Status": "available",
"PendingModifiedValues": {},
"MemberClusters": [
  "test-cluster-001",
  "test-cluster-002",
  "test-cluster-003"
],
"NodeGroups": [
  {
    "NodeGroupId": "0001",
    "Status": "available",
    "PrimaryEndpoint": {
      "Address": "test-
cluster.g2xbih.ng.0001.usw2.cache.amazonaws.com",
      "Port": 6379
    },
    "ReaderEndpoint": {
      "Address": "test-cluster-
ro.g2xbih.ng.0001.usw2.cache.amazonaws.com",
      "Port": 6379
    },
    "NodeGroupMembers": [
      {
        "CacheClusterId": "test-cluster-001",
        "CacheNodeId": "0001",
        "ReadEndpoint": {
          "Address": "test-
cluster-001.g2xbih.0001.usw2.cache.amazonaws.com",
          "Port": 6379
        },
        "PreferredAvailabilityZone": "us-west-2c",
        "CurrentRole": "primary"
      },
      {
        "CacheClusterId": "test-cluster-002",
        "CacheNodeId": "0001",
        "ReadEndpoint": {
          "Address": "test-
cluster-002.g2xbih.0001.usw2.cache.amazonaws.com",
          "Port": 6379
        }
      }
    ]
  }
]
```

```

        },
        "PreferredAvailabilityZone": "us-west-2b",
        "CurrentRole": "replica"
    },
    {
        "CacheClusterId": "test-cluster-003",
        "CacheNodeId": "0001",
        "ReadEndpoint": {
            "Address": "test-
cluster-003.g2xbih.0001.usw2.cache.amazonaws.com",
            "Port": 6379
        },
        "PreferredAvailabilityZone": "us-west-2a",
        "CurrentRole": "replica"
    }
]
}
],
"SnapshottingClusterId": "test-cluster-002",
"AutomaticFailover": "enabled",
"MultiAZ": "disabled",
"SnapshotRetentionLimit": 1,
"SnapshotWindow": "08:00-09:00",
"ClusterEnabled": false,
"CacheNodeType": "cache.r5.large",
"TransitEncryptionEnabled": false,
"AtRestEncryptionEnabled": false
}
}

```

有关更多信息，请参阅《Elasticache 用户指南》中的[修改复制组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ModifyReplicationGroup](#)中的。

## modify-user-group

以下代码示例演示了如何使用 `modify-user-group`。

### AWS CLI

#### 修改用户组

以下 `modify-user-group` 示例将用户添加到用户组。

```
aws elasticache modify-user-group \  
  --user-group-id myusergroup \  
  --user-ids-to-add user1
```

输出：

```
{  
  "UserGroupId": "myusergroup",  
  "Status": "modifying",  
  "Engine": "redis",  
  "UserIds": [  
    "default"  
  ],  
  "PendingChanges": {  
    "UserIdsToAdd": [  
      "user1"  
    ]  
  },  
  "ReplicationGroups": [],  
  "ARN": "arn:aws:elasticache:us-west-2:xxxxxxxxxx52:usergroup:myusergroup"  
}
```

有关更多信息，请参阅《Elasticache [用户指南](#)》中的[使用基于角色的访问控制 \(RBAC\) 对用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyUserGroup](#)中的。

## modify-user

以下代码示例演示了如何使用 modify-user。

### AWS CLI

#### 修改用户

以下modify-user示例修改了用户的访问字符串。

```
aws elasticache modify-user \  
  --user-id user2 \  
  --append-access-string "on ~* +@all"
```

输出：

```
{
  "UserId": "user2",
  "UserName": "myUser",
  "Status": "modifying",
  "Engine": "redis",
  "AccessString": "on ~* +@all",
  "UserGroupIds": [],
  "Authentication": {
    "Type": "password",
    "PasswordCount": 1
  },
  "ARN": "arn:aws:elasticache:us-west-2:xxxxxxxxxx52:user:user2"
}
```

有关更多信息，请参阅《Elasticache [用户指南](#)》中的[使用基于角色的访问控制 \(RBAC\) 对用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyUser](#)中的。

## **purchase-reserved-cache-nodes-offering**

以下代码示例演示了如何使用 `purchase-reserved-cache-nodes-offering`。

### AWS CLI

要购买 `reserved-cache-node-offering`

以下`purchase-reserved-cache-nodes-offering`示例允许您购买预留缓存节点产品。

```
aws elasticache purchase-reserved-cache-nodes-offering \
  --reserved-cache-nodes-offering-id xxxxxxxx-4da5-4b90-b92d-929fbd7abed2
```

### 输出

```
{
  "ReservedCacheNode": {
    "ReservedCacheNodeId": "ri-2020-06-30-17-59-40-474",
    "ReservedCacheNodesOfferingId": "xxxxxxx-4da5-4b90-b92d-929fbd7abed2",
    "CacheNodeType": "cache.m3.2xlarge",
    "StartTime": "2020-06-30T17:59:40.474000+00:00",
    "Duration": 31536000,
    "FixedPrice": 1772.0,
  }
}
```

```
    "UsagePrice": 0.0,
    "CacheNodeCount": 1,
    "ProductDescription": "redis",
    "OfferingType": "Heavy Utilization",
    "State": "payment-pending",
    "RecurringCharges": [
      {
        "RecurringChargeAmount": 0.25,
        "RecurringChargeFrequency": "Hourly"
      }
    ]
  }
}
```

有关更多信息，请参阅《Elasticache Redis 用户指南》中的“[获取有关预留节点产品的信息](#)”或《Elasticache Memcached 用户指南》中的“[获取有关预留节点产品的信息](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PurchaseReservedCacheNodesOffering](#)中的。

## reboot-cache-cluster

以下代码示例演示了如何使用 `reboot-cache-cluster`。

### AWS CLI

#### 重启缓存集群

以下 `reboot-cache-cluster` 示例重新启动已配置集群中的部分或全部缓存节点。此操作将所有修改过的缓存参数组应用于集群。重启操作会尽快进行，并导致集群暂时中断。在重启期间，群集状态设置为 REBOOTING。

```
aws elasticache reboot-cache-cluster \
  --cache-cluster-id "my-cluster-001" \
  --cache-node-ids-to-reboot "0001"
```

输出：

```
{
  "CacheCluster": {
    "CacheClusterId": "my-cluster-001",
    "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
  }
}
```

```
"CacheNodeType": "cache.r5.xlarge",
"Engine": "redis",
"EngineVersion": "5.0.5",
"CacheClusterStatus": "rebooting cache cluster nodes",
"NumCacheNodes": 1,
"PreferredAvailabilityZone": "us-west-2a",
"CacheClusterCreateTime": "2019-11-26T03:35:04.546Z",
"PreferredMaintenanceWindow": "mon:04:05-mon:05:05",
"PendingModifiedValues": {},
"NotificationConfiguration": {
  "TopicArn": "arn:aws:sns:us-west-2:xxxxxxxxxx152:My_Topic",
  "TopicStatus": "active"
},
"CacheSecurityGroups": [],
"CacheParameterGroup": {
  "CacheParameterGroupName": "mygroup",
  "ParameterApplyStatus": "in-sync",
  "CacheNodeIdsToReboot": []
},
"CacheSubnetGroupName": "kxkxk",
"AutoMinorVersionUpgrade": true,
"SecurityGroups": [
  {
    "SecurityGroupId": "sg-xxxxxxxxxxxx836",
    "Status": "active"
  },
  {
    "SecurityGroupId": "sg-xxxxxxx7b",
    "Status": "active"
  }
],
"ReplicationGroupId": "my-cluster",
"SnapshotRetentionLimit": 0,
"SnapshotWindow": "07:30-08:30",
"TransitEncryptionEnabled": false,
"AtRestEncryptionEnabled": false
}
}
```

有关更多信息，请参阅 Elasticache 用户指南中的重启集群 < <https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/clusters.rebooting.html>。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RebootCacheCluster](#) 中的。

## reset-cache-parameter-group

以下代码示例演示了如何使用 `reset-cache-parameter-group`。

### AWS CLI

#### 重置缓存参数组

以下 `reset-cache-parameter-group` 示例将缓存参数组的参数修改为引擎或系统的默认值。您可以通过提交参数名称列表来重置特定参数。要重置整个缓存参数组，请指定 `--reset-all-parameters` 和 `--cache-parameter-group-name` 参数。

```
aws elasticache reset-cache-parameter-group \  
  --cache-parameter-group-name "mygroup" \  
  --reset-all-parameters
```

输出：

```
{  
  "CacheParameterGroupName": "mygroup"  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ResetCacheParameterGroup](#) 中的。

## start-migration

以下代码示例演示了如何使用 `start-migration`。

### AWS CLI

#### 开始迁移

以下内容使用 Redis `start-migration` 引擎将您的数据从亚马逊 EC2 上的自托管 Redis 迁移到亚马逊 ElastiCache。

```
aws elasticache start-migration \  
  --replication-group-id test \  
  --customer-node-endpoint-list  
  "Address='test.g2xbih.ng.0001.usw2.cache.amazonaws.com',Port=6379"
```



## 输出

```
{
  "ReplicationGroup": {
    "ReplicationGroupId": "test",
    "Description": "test",
    "GlobalReplicationGroupInfo": {},
    "Status": "modifying",
    "PendingModifiedValues": {},
    "MemberClusters": [
      "test-001",
      "test-002",
      "test-003"
    ],
    "NodeGroups": [
      {
        "NodeGroupId": "0001",
        "Status": "available",
        "PrimaryEndpoint": {
          "Address": "test.g2xbih.ng.0001.usw2.cache.amazonaws.com",
          "Port": 6379
        },
        "ReaderEndpoint": {
          "Address": "test-ro.g2xbih.ng.0001.usw2.cache.amazonaws.com",
          "Port": 6379
        },
        "NodeGroupMembers": [
          {
            "CacheClusterId": "test-001",
            "CacheNodeId": "0001",
            "ReadEndpoint": {
              "Address":
"test-001.g2xbih.0001.usw2.cache.amazonaws.com",
              "Port": 6379
            },
            "PreferredAvailabilityZone": "us-west-2a",
            "CurrentRole": "primary"
          },
          {
            "CacheClusterId": "test-002",
            "CacheNodeId": "0001",
            "ReadEndpoint": {
              "Address":
"test-002.g2xbih.0001.usw2.cache.amazonaws.com",
```

```

        "Port": 6379
      },
      "PreferredAvailabilityZone": "us-west-2c",
      "CurrentRole": "replica"
    },
    {
      "CacheClusterId": "test-003",
      "CacheNodeId": "0001",
      "ReadEndpoint": {
        "Address":
"test-003.g2xbih.0001.usw2.cache.amazonaws.com",
        "Port": 6379
      },
      "PreferredAvailabilityZone": "us-west-2b",
      "CurrentRole": "replica"
    }
  ]
}
],
"SnapshottingClusterId": "test-002",
"AutomaticFailover": "enabled",
"MultiAZ": "enabled",
"SnapshotRetentionLimit": 1,
"SnapshotWindow": "07:30-08:30",
"ClusterEnabled": false,
"CacheNodeType": "cache.r5.large",
"TransitEncryptionEnabled": false,
"AtRestEncryptionEnabled": false
}
}

```

有关更多信息，请参阅《Elasticache 用户指南》ElastiCache 中的[在线迁移到](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[StartMigration](#) 中的。

## test-failover

以下代码示例演示了如何使用 test-failover。

### AWS CLI

#### 测试节点组的故障转移

以下test-failover示例测试复制组（控制台中称为群集）中指定节点组（在控制台中称为分片）上的自动故障转移。

```
aws elasticache test-failover /
  --replication-group-id "mycluster" /
  --node-group-id "0001"
```

输出：

```
{
  "ReplicationGroup": {
    "ReplicationGroupId": "mycluster",
    "Description": "My Cluster",
    "Status": "available",
    "PendingModifiedValues": {},
    "MemberClusters": [
      "mycluster-0001-001",
      "mycluster-0001-002",
      "mycluster-0001-003",
      "mycluster-0002-001",
      "mycluster-0002-002",
      "mycluster-0002-003",
      "mycluster-0003-001",
      "mycluster-0003-002",
      "mycluster-0003-003"
    ],
    "NodeGroups": [
      {
        "NodeGroupId": "0001",
        "Status": "available",
        "Slots": "0-5461",
        "NodeGroupMembers": [
          {
            "CacheClusterId": "mycluster-0001-001",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2b"
          },
          {
            "CacheClusterId": "mycluster-0001-002",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2a"
          },
          {
```

```
        "CacheClusterId": "mycluster-0001-003",
        "CacheNodeId": "0001",
        "PreferredAvailabilityZone": "us-west-2c"
    }
]
},
{
    "NodeGroupId": "0002",
    "Status": "available",
    "Slots": "5462-10922",
    "NodeGroupMembers": [
        {
            "CacheClusterId": "mycluster-0002-001",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2a"
        },
        {
            "CacheClusterId": "mycluster-0002-002",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2b"
        },
        {
            "CacheClusterId": "mycluster-0002-003",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2c"
        }
    ]
},
{
    "NodeGroupId": "0003",
    "Status": "available",
    "Slots": "10923-16383",
    "NodeGroupMembers": [
        {
            "CacheClusterId": "mycluster-0003-001",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2c"
        },
        {
            "CacheClusterId": "mycluster-0003-002",
            "CacheNodeId": "0001",
            "PreferredAvailabilityZone": "us-west-2b"
        }
    ]
}
```

```
        "CacheClusterId": "mycluster-0003-003",
        "CacheNodeId": "0001",
        "PreferredAvailabilityZone": "us-west-2a"
    }
]
},
"AutomaticFailover": "enabled",
"ConfigurationEndpoint": {
    "Address": "mycluster.xxxxih.clustercfg.usw2.cache.amazonaws.com",
    "Port": 6379
},
"SnapshotRetentionLimit": 1,
"SnapshotWindow": "13:00-14:00",
"ClusterEnabled": true,
"CacheNodeType": "cache.r5.large",
"TransitEncryptionEnabled": false,
"AtRestEncryptionEnabled": false
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TestFailover](#)中的。

## MediaStore 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 MediaStore。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### **create-container**

以下代码示例演示了如何使用 `create-container`。

#### AWS CLI

##### 创建容器

以下 `create-container` 示例创建一个新的空容器。

```
aws mediastore create-container --container-name ExampleContainer
```

输出：

```
{
  "Container": {
    "AccessLoggingEnabled": false,
    "CreationTime": 1563557265,
    "Name": "ExampleContainer",
    "Status": "CREATING",
    "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer"
  }
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的 [创建容器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateContainer](#) 中的。

### **delete-container-policy**

以下代码示例演示了如何使用 `delete-container-policy`。

#### AWS CLI

##### 删除容器策略

以下 `delete-container-policy` 示例删除分配给指定容器的策略。删除策略后，AWS Elemental MediaStore 会自动将默认策略分配给容器。

```
aws mediastore delete-container-policy \  
  --container-name LiveEvents
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS Elemental MediaStore API 参考[DeleteContainerPolicy](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteContainerPolicy](#)中的。

## delete-container

以下代码示例演示了如何使用 delete-container。

### AWS CLI

#### 删除容器

以下delete-container示例删除了指定的容器。您只能在容器没有对象时将其删除。

```
aws mediastore delete-container \  
  --container-name=ExampleLiveDemo
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[删除容器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteContainer](#)中的。

## delete-cors-policy

以下代码示例演示了如何使用 delete-cors-policy。

### AWS CLI

#### 删除 CORS 策略

以下delete-cors-policy示例删除分配给指定容器的跨源资源共享 (CORS) 策略。

```
aws mediastore delete-cors-policy \  
  --container-name ExampleContainer
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 基本 MediaStore 用户指南》中的[删除 CORS 策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCorsPolicy](#)中的。

## delete-lifecycle-policy

以下代码示例演示了如何使用 delete-lifecycle-policy。

### AWS CLI

#### 删除对象生命周期策略

以下delete-lifecycle-policy示例删除附加到指定容器的对象生命周期策略。此更改最多可能需要 20 分钟才能生效。

```
aws mediastore delete-lifecycle-policy \  
  --container-name LiveEvents
```

此命令不生成任何输出。

有关更多信息，请参阅 [Element AWS al MediaStore 用户指南中的删除对象生命周期策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLifecyclePolicy](#)中的。

## describe-container

以下代码示例演示了如何使用 describe-container。

### AWS CLI

#### 查看容器的详细信息

以下describe-container示例显示了指定容器的详细信息。

```
aws mediastore describe-container \  
  --container-name ExampleContainer
```

输出：



```
{
  "Container": {
    "CreationTime": 1563558086,
    "AccessLoggingEnabled": false,
    "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
    "Status": "ACTIVE",
    "Name": "ExampleContainer",
    "Endpoint": "https://aaabbbcccddee.data.mediastore.us-west-2.amazonaws.com"
  }
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[查看容器的详细信息](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeContainer](#)中的。

## describe-object

以下代码示例演示了如何使用 describe-object。

### AWS CLI

查看特定容器中的对象和文件夹列表

以下 describe-object 示例显示存储在特定容器中的项目（对象和文件夹）。

```
aws mediastore-data describe-object \
  --endpoint https://aaabbbcccddee.data.mediastore.us-west-2.amazonaws.com \
  --path /folder_name/file1234.jpg
```

输出：

```
{
  "ContentType": "image/jpeg",
  "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
  "ContentLength": "2307346",
  "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f5555555555555555da6d3"
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[查看对象的详细信息](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeObject](#)中的。

## get-container-policy

以下代码示例演示了如何使用 `get-container-policy`。

### AWS CLI

#### 查看容器策略

以下 `get-container-policy` 示例显示了指定容器的基于资源的策略。

```
aws mediastore get-container-policy \  
  --container-name ExampleLiveDemo
```

输出：

```
{  
  "Policy": {  
    "Version": "2012-10-17",  
    "Statement": [  
      {  
        "Sid": "PublicReadOverHttps",  
        "Effect": "Allow",  
        "Principal": {  
          "AWS": "arn:aws:iam::111122223333:root"  
        },  
        "Action": [  
          "mediastore:GetObject",  
          "mediastore:DescribeObject"  
        ],  
        "Resource": "arn:aws:mediastore:us-west-2:111122223333:container/  
ExampleLiveDemo/",  
        "Condition": {  
          "Bool": {  
            "aws:SecureTransport": "true"  
          }  
        }  
      }  
    ]  
  }  
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[查看容器策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetContainerPolicy](#) 中的。

## get-cors-policy

以下代码示例演示了如何使用 `get-cors-policy`。

### AWS CLI

#### 查看 CORS 策略

以下 `get-cors-policy` 示例显示了分配给指定容器的跨源资源共享 (CORS) 策略。

```
aws mediastore get-cors-policy \  
  --container-name ExampleContainer \  
  --region us-west-2
```

输出：

```
{  
  "CorsPolicy": [  
    {  
      "AllowedMethods": [  
        "GET",  
        "HEAD"  
      ],  
      "MaxAgeSeconds": 3000,  
      "AllowedOrigins": [  
        ""  
      ],  
      "AllowedHeaders": [  
        ""  
      ]  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS 基本 MediaStore 用户指南》中的[查看 CORS 策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetCorsPolicy](#) 中的。

## get-lifecycle-policy

以下代码示例演示了如何使用 `get-lifecycle-policy`。

## AWS CLI

### 查看对象生命周期策略

以下`get-lifecycle-policy`示例显示了附加到指定容器的对象生命周期策略。

```
aws mediastore get-lifecycle-policy \  
  --container-name LiveEvents
```

输出：

```
{  
  "LifecyclePolicy": {  
    "rules": [  
      {  
        "definition": {  
          "path": [  
            {  
              "prefix": "Football/"  
            },  
            {  
              "prefix": "Baseball/"  
            }  
          ],  
          "days_since_create": [  
            {  
              "numeric": [  
                ">",  
                28  
              ]  
            }  
          ]  
        },  
        "action": "EXPIRE"  
      }  
    ]  
  }  
}
```

有关更多信息，请参阅 [AWS Elemental MediaStore 用户指南中的查看对象生命周期策略](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 GetLifecyclePolicy](#) 中的。

## get-object

以下代码示例演示了如何使用 `get-object`。

### AWS CLI

#### 下载对象

以下 `get-object` 示例将对象下载到指定的终端节点。

```
aws mediastore-data get-object \  
  --endpoint https://aaabbbccdddee.data.mediastore.us-west-2.amazonaws.com \  
  --path=/folder_name/README.md README.md
```

输出：

```
{  
  "ContentLength": "2307346",  
  "ContentType": "image/jpeg",  
  "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",  
  "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9e9999e4dd89ff7f5555555555555da6d3",  
  "StatusCode": 200  
}
```

#### 下载对象的一部分

以下 `get-object` 示例将对象的一部分下载到指定的终端节点。

```
aws mediastore-data get-object \  
  --endpoint https://aaabbbccdddee.data.mediastore.us-west-2.amazonaws.com \  
  --path /folder_name/README.md \  
  --range="bytes=0-100" README2.md
```

输出：

```
{  
  "StatusCode": 206,  
  "ContentRange": "bytes 0-100/2307346",  
  "ContentLength": "101",  
  "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",  
  "ContentType": "image/jpeg",  
  "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9e9999e4dd89ff7f5555555555555da6d3"
```

```
}
```

有关更多信息，请参阅 [AWS Elemental MediaStore 用户指南中的下载对象](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `GetObject`](#) 中的。

## list-containers

以下代码示例演示了如何使用 `list-containers`。

### AWS CLI

#### 查看容器列表

以下 `list-containers` 示例显示了与您的账户关联的所有容器的列表。

```
aws mediastore list-containers
```

输出：

```
{
  "Containers": [
    {
      "CreationTime": 1505317931,
      "Endpoint": "https://aaabbbcccddee.data.mediastore.us-
west-2.amazonaws.com",
      "Status": "ACTIVE",
      "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleLiveDemo",
      "AccessLoggingEnabled": false,
      "Name": "ExampleLiveDemo"
    },
    {
      "CreationTime": 1506528818,
      "Endpoint": "https://fffggghhhiiijj.data.mediastore.us-
west-2.amazonaws.com",
      "Status": "ACTIVE",
      "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
      "AccessLoggingEnabled": false,
      "Name": "ExampleContainer"
    }
  ]
}
```

```
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[查看容器列表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListContainers](#)中的。

## list-items

以下代码示例演示了如何使用 list-items。

### AWS CLI

示例 1：查看特定容器中的对象和文件夹列表

以下 list-items 示例显示存储在指定容器中的项目（对象和文件夹）。

```
aws mediastore-data list-items \  
  --endpoint https://aaabbbccdddee.data.mediastore.us-west-2.amazonaws.com
```

输出：

```
{  
  "Items": [  
    {  
      "ContentType": "image/jpeg",  
      "LastModified": 1563571859.379,  
      "Name": "filename.jpg",  
      "Type": "OBJECT",  
      "ETag":  
      "543ab21abcd1a234ab123456a1a2b12345ab12abc12a1234abc1a2bc12345a12",  
      "ContentLength": 3784  
    },  
    {  
      "Type": "FOLDER",  
      "Name": "ExampleLiveDemo"  
    }  
  ]  
}
```

示例 2：查看特定文件夹中的对象和文件夹列表

以下 list-items 示例显示存储在特定文件夹中的项目（对象和文件夹）。

```
aws mediastore-data list-items \  
  --endpoint https://aaabbbccdddee.data.mediastore.us-west-2.amazonaws.com
```

输出：

```
{  
  "Items": [  
    {  
      "ContentType": "image/jpeg",  
      "LastModified": 1563571859.379,  
      "Name": "filename.jpg",  
      "Type": "OBJECT",  
      "ETag":  
      "543ab21abcd1a234ab123456a1a2b12345ab12abc12a1234abc1a2bc12345a12",  
      "ContentLength": 3784  
    },  
    {  
      "Type": "FOLDER",  
      "Name": "ExampleLiveDemo"  
    }  
  ]  
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[查看对象列表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListItems](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

AWS CLI

列出容器的标签

以下 `list-tags-for-resource` 示例显示了分配给指定容器的标签键和值。

```
aws mediastore list-tags-for-resource \  
  --resource arn:aws:mediastore:us-west-2:1213456789012:container/ExampleContainer
```

输出：



```
{
  "Tags": [
    {
      "Value": "Test",
      "Key": "Environment"
    },
    {
      "Value": "West",
      "Key": "Region"
    }
  ]
}
```

有关更多信息，请参阅 AWS Elemental MediaStore API 参考[ListTagsForResource](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## put-container-policy

以下代码示例演示了如何使用 put-container-policy。

### AWS CLI

#### 编辑容器策略

以下put-container-policy示例为指定的容器分配了不同的策略。在此示例中，更新的策略是在名为的文件中定义的LiveEventsContainerPolicy.json。

```
aws mediastore put-container-policy \
  --container-name LiveEvents \
  --policy file://LiveEventsContainerPolicy.json
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[编辑容器策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutContainerPolicy](#)中的。

## put-cors-policy

以下代码示例演示了如何使用 put-cors-policy。

## AWS CLI

### 示例 1：添加 CORS 策略

以下 `put-cors-policy` 示例向指定容器添加跨源资源共享 (CORS) 策略。CORS 策略的内容位于名为 `corsPolicy.json` 的文件中。

```
aws mediastore put-cors-policy \  
  --container-name ExampleContainer \  
  --cors-policy file://corsPolicy.json
```

此命令不生成任何输出。

有关更多信息，请参阅 [AWS Elemental MediaStore 用户指南中的向容器添加 CORS 策略](#)。

### 示例 2：编辑 CORS 策略

以下 `put-cors-policy` 示例更新了分配给指定容器的跨源资源共享 (CORS) 策略。更新后的 CORS 策略的内容位于名为 `corsPolicy2.json` 的文件中。

有关更多信息，请参阅《AWS 基本 MediaStore 用户指南》中的 [编辑 CORS 策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutCorsPolicy](#) 中的。

## **put-lifecycle-policy**

以下代码示例演示了如何使用 `put-lifecycle-policy`。

## AWS CLI

### 创建对象生命周期策略

以下 `put-lifecycle-policy` 示例将对象生命周期策略附加到指定的容器。这使您可以指定服务应在容器中存储对象多长时间。MediaStore 在容器中的对象达到过期日期后将其删除，如策略所示，该策略位于名为 `LiveEventsLifecyclePolicy.json` 的文件中。

```
aws mediastore put-lifecycle-policy \  
  --container-name ExampleContainer \  
  --lifecycle-policy file://ExampleLifecyclePolicy.json
```

此命令不生成任何输出。

有关更多信息，请参阅 [AWS Elemental MediaStore 用户指南中的向容器添加对象生命周期策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutLifecyclePolicy](#)中的。

## put-object

以下代码示例演示了如何使用 put-object。

### AWS CLI

#### 上传对象

以下put-object示例将对象上传到指定的容器。您可以指定在容器中保存对象的文件夹路径。如果该文件夹已经存在，AWS Elemental 会将该对象 MediaStore 存储在文件夹中。如果该文件夹不存在，则服务会创建该文件夹，然后将该对象存储在文件夹中。

```
aws mediastore-data put-object \  
  --endpoint https://aaabbbcccddee.data.mediastore.us-west-2.amazonaws.com \  
  --body README.md \  
  --path /folder_name/README.md \  
  --cache-control "max-age=6, public" \  
  --content-type binary/octet-stream
```

#### 输出：

```
{  
  "ContentSHA256":  
    "74b5fdb517f423ed750ef214c44adfe2be36e37d861eafe9c842cbe1bf387a9d",  
  "StorageClass": "TEMPORAL",  
  "ETag": "af3e4731af032167a106015d1f2fe934e68b32ed1aa297a9e325f5c64979277b"  
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[上传对象](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutObject](#)中的。

## start-access-logging

以下代码示例演示了如何使用 start-access-logging。

### AWS CLI

#### 在容器上启用访问日志记录

以下 `start-access-logging` 示例在指定容器上启用访问日志记录。

```
aws mediastore start-access-logging \  
  --container-name LiveEvents
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[为容器启用访问日志记录](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[StartAccessLogging](#)中的。

## stop-access-logging

以下代码示例演示了如何使用 `stop-access-logging`。

### AWS CLI

禁用容器上的访问日志记录

以下 `stop-access-logging` 示例在指定容器上禁用访问日志记录。

```
aws mediastore stop-access-logging \  
  --container-name LiveEvents
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[禁用容器的访问日志记录](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[StopAccessLogging](#)中的。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

### AWS CLI

向容器添加标签

以下 `tag-resource` 示例将标签键和值添加到指定的容器。

```
aws mediastore tag-resource \  
  --container-name LiveEvents
```

```
--resource arn:aws:mediastore:us-west-2:123456789012:container/ExampleContainer \
--tags '[{"Key": "Region", "Value": "West"}, {"Key": "Environment", "Value": "Test"}]'
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS Elemental MediaStore API 参考 [TagResource](#) 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从容器中移除标签

以下 untag-resource 示例从容器中移除指定的标签键及其关联值。

```
aws mediastore untag-resource \
--resource arn:aws:mediastore:us-west-2:123456789012:container/ExampleContainer \
--tag-keys Region
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS Elemental MediaStore API 参考 [UntagResource](#) 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## 使用 Amazon EMR 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon EMR 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## add-instance-fleet

以下代码示例演示了如何使用 add-instance-fleet。

AWS CLI

向集群添加任务实例队列

此示例将新的任务实例队列添加到指定的集群。

命令：

```
aws emr add-instance-fleet --cluster-id 'j-12ABCDEFGH134JK' --instance-fleet
InstanceFleetType=TASK,TargetSpotCapacity=1,LaunchSpecifications={SpotSpecification='{Timeo
```

输出：

```
{
  "ClusterId": "j-12ABCDEFGH134JK",
  "InstanceFleetId": "if-23ABCDEFGH145JJ"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddInstanceFleet](#)中的。

## add-steps

以下代码示例演示了如何使用 add-steps。

AWS CLI

1. 向集群添加自定义 JAR 步骤

**命令:**

```
aws emr add-steps --cluster-id j-XXXXXXXX --steps
  Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,Jar=s3://mybucket/
mytest.jar,Args=arg1,arg2,arg3
  Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,Jar=s3://mybucket/
mytest.jar,MainClass=mymainclass,Args=arg1,arg2,arg3
```

**必填参数 :**

Jar

**可选参数 :**

Type, Name, ActionOnFailure, Args

**输出 :**

```
{
  "StepIds":[
    "s-XXXXXXXX",
    "s-YYYYYYYY"
  ]
}
```

**2. 向集群添加流式处理步骤****命令:**

```
aws emr add-steps --cluster-id j-XXXXXXXX --steps Type=STREAMING,Name='Streaming
Program',ActionOnFailure=CONTINUE,Args=[-files,s3://elasticmapreduce/samples/
wordcount/wordSplitter.py,-mapper,wordSplitter.py,-reducer,aggregate,-input,s3://
elasticmapreduce/samples/wordcount/input,-output,s3://mybucket/wordcount/output]
```

**必填参数 :**

Type, Args

**可选参数 :**

```
Name, ActionOnFailure
```

JSON 等效项 ( step.json 的内容 ) :

```
[
  {
    "Name": "JSON Streaming Step",
    "Args": ["-files", "s3://elasticmapreduce/samples/wordcount/wordSplitter.py", "-mapper", "wordSplitter.py", "-reducer", "aggregate", "-input", "s3://elasticmapreduce/samples/wordcount/input", "-output", "s3://mybucket/wordcount/output"],
    "ActionOnFailure": "CONTINUE",
    "Type": "STREAMING"
  }
]
```

注意 : JSON 参数必须将选项和值作为它们自己的项目包含在列表中。

命令 ( 使用 step.json ) :

```
aws emr add-steps --cluster-id j-XXXXXXXX --steps file://./step.json
```

输出 :

```
{
  "StepIds": [
    "s-XXXXXXXX",
    "s-YYYYYYYY"
  ]
}
```

3. 向集群添加包含多个文件的流式处理步骤 ( 仅限 JSON )

JSON ( 多文件.json ) :

```
[
  {
    "Name": "JSON Streaming Step",
    "Type": "STREAMING",
    "ActionOnFailure": "CONTINUE",
    "Args": [
      "-files",
```



```

        "s3://mybucket/mapper.py,s3://mybucket/reducer.py",
        "-mapper",
        "mapper.py",
        "-reducer",
        "reducer.py",
        "-input",
        "s3://mybucket/input",
        "-output",
        "s3://mybucket/output"]
    }
]

```

**命令:**

```
aws emr add-steps --cluster-id j-XXXXXXXX --steps file://./multiplefiles.json
```

**必填参数:**

Type, Args

**可选参数:**

Name, ActionOnFailure

**输出:**

```

{
  "StepIds":[
    "s-XXXXXXXX",
  ]
}

```

**4. 向集群添加 Hive 步骤****命令:**

```
aws emr add-steps --cluster-id j-XXXXXXXX --steps Type=HIVE,Name='Hive
program',ActionOnFailure=CONTINUE,Args=[-f,s3://mybucket/myhivescript.q,-
d,INPUT=s3://mybucket/myhiveinput,-d,OUTPUT=s3://mybucket/myhiveoutput,arg1,arg2]
Type=HIVE,Name='Hive steps',ActionOnFailure=TERMINATE_CLUSTER,Args=[-
```

```
f,s3://elasticmapreduce/samples/hive-ads/libs/model-build.q,-d,INPUT=s3://
elasticmapreduce/samples/hive-ads/tables,-d,OUTPUT=s3://mybucket/hive-ads/
output/2014-04-18/11-07-32,-d,LIBS=s3://elasticmapreduce/samples/hive-ads/libs]
```

必填参数：

Type, Args

可选参数：

Name, ActionOnFailure

输出：

```
{
  "StepIds":[
    "s-XXXXXXXX",
    "s-YYYYYYYY"
  ]
}
```

## 5. 向集群添加 Pig 步骤

命令：

```
aws emr add-steps --cluster-id j-XXXXXXXX --steps Type=PIG,Name='Pig
program',ActionOnFailure=CONTINUE,Args=[-f,s3://mybucket/mypigscript.pig,-
p,INPUT=s3://mybucket/mypiginput,-p,OUTPUT=s3://mybucket/mypigoutput,arg1,arg2]
Type=PIG,Name='Pig program',Args=[-f,s3://elasticmapreduce/samples/pig-apache/do-
reports2.pig,-p,INPUT=s3://elasticmapreduce/samples/pig-apache/input,-p,OUTPUT=s3://
mybucket/pig-apache/output,arg1,arg2]
```

必填参数：

Type, Args

可选参数：

Name, ActionOnFailure

输出：

```
{
  "StepIds": [
    "s-XXXXXXXX",
    "s-YYYYYYYY"
  ]
}
```

## 6. 向集群添加 Impala 步骤

命令：

```
aws emr add-steps --cluster-id j-XXXXXXXX --steps Type=IMPALA,Name='Impala
program',ActionOnFailure=CONTINUE,Args=--impala-script,s3://myimpala/input,--
console-output-path,s3://myimpala/output
```

必填参数：

Type, Args

可选参数：

Name, ActionOnFailure

输出：

```
{
  "StepIds": [
    "s-XXXXXXXX",
    "s-YYYYYYYY"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddSteps](#)中的。

## add-tags

以下代码示例演示了如何使用 add-tags。

## AWS CLI

### 1. 向集群添加标签

命令:

```
aws emr add-tags --resource-id j-xxxxxxx --tags name="John Doe" age=29 sex=male
address="123 East NW Seattle"
```

输出:

```
None
```

### 2. 列出集群的标签

--命令:

```
aws emr describe-cluster --cluster-id j-XXXXXXYY --query Cluster.Tags
```

输出:

```
[
  {
    "Value": "male",
    "Key": "sex"
  },
  {
    "Value": "123 East NW Seattle",
    "Key": "address"
  },
  {
    "Value": "John Doe",
    "Key": "name"
  },
  {
    "Value": "29",
    "Key": "age"
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddTags](#)中的。

## create-cluster-examples

以下代码示例演示了如何使用 `create-cluster-examples`。

### AWS CLI

以下大多数示例都假设您指定了自己的 Amazon EMR 服务角色和 Amazon EC2 实例配置文件。如果您尚未执行此操作，则必须指定每个必需的 IAM 角色或在创建集群时使用 `--use-default-roles` 参数。有关指定 IAM 角色的更多信息，请参阅 [Amazon EMR 管理指南中的为亚马逊 EMR AWS 服务权限配置 IAM 角色](#)。

#### 示例 1：创建集群

以下 `create-cluster` 示例创建了一个简单的 EMR 集群。

```
aws emr create-cluster \  
  --release-label emr-5.14.0 \  
  --instance-type m4.large \  
  --instance-count 2
```

此命令不生成任何输出。

#### 示例 2：创建具有默认 ServiceRole 角色和角色的 Amazon EMR 集群 InstanceProfile

以下 `create-cluster` 示例创建了一个使用该配置的 Amazon EMR 集群。 `--instance-groups`

```
aws emr create-cluster \  
  --release-label emr-5.14.0 \  
  --service-role EMR_DefaultRole \  
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \  
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large  
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large
```

#### 示例 3：创建使用实例队列的 Amazon EMR 集群

以下 `create-cluster` 示例创建了一个使用该 `--instance-fleets` 配置的 Amazon EMR 集群，为每个队列指定两种实例类型和两个 EC2 子网。

```
aws emr create-cluster \  
  --release-label emr-5.14.0 \  
  --instance-fleets
```

```

--service-role EMR_DefaultRole \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,SubnetIds=['subnet-
ab12345c','subnet-de67890f'] \
--instance-fleets
InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m4.la
InstanceFleetType=CORE,TargetSpotCapacity=11,InstanceTypeConfigs=['{InstanceType=m4.large,B

```

#### 示例 4：创建具有默认角色的集群

以下 `create-cluster` 示例使用 `--use-default-roles` 参数指定默认服务角色和实例配置文件。

```

aws emr create-cluster \
--release-label emr-5.9.0 \
--use-default-roles \
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \
--auto-terminate

```

#### 示例 5：创建集群并指定要安装的应用程序

以下 `create-cluster` 示例使用 `--applications` 参数指定 Amazon EMR 安装的应用程序。此示例安装了 Hadoop、Hive 和 Pig。

```

aws emr create-cluster \
--applications Name=Hadoop Name=Hive Name=Pig \
--release-label emr-5.9.0 \
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \
--auto-terminate

```

#### 示例 6：创建包含 Spark 的集群

以下示例安装了 Spark。

```

aws emr create-cluster \
--release-label emr-5.9.0 \
--applications Name=Spark \
--ec2-attributes KeyName=myKey \
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \

```

```
--auto-terminate
```

### 示例 7：指定用于集群实例的自定义 AMI

以下 `create-cluster` 示例基于标识为 Amazon Linux AMI 的集群实例 `ami-a518e6df`。

```
aws emr create-cluster \  
  --name "Cluster with My Custom AMI" \  
  --custom-ami-id ami-a518e6df \  
  --ebs-root-volume-size 20 \  
  --release-label emr-5.9.0 \  
  --use-default-roles \  
  --instance-count 2 \  
  --instance-type m4.large
```

### 示例 8：自定义应用程序配置

以下示例使用 `--configurations` 参数指定包含 Hadoop 应用程序自定义项的 JSON 配置文件。有关更多信息，请参阅 Amazon EMR 发行版指南中的 [配置应用程序](#)。

`configurations.json` 的内容：

```
[  
  {  
    "Classification": "mapred-site",  
    "Properties": {  
      "mapred.tasktracker.map.tasks.maximum": 2  
    }  
  },  
  {  
    "Classification": "hadoop-env",  
    "Properties": {},  
    "Configurations": [  
      {  
        "Classification": "export",  
        "Properties": {  
          "HADOOP_DATANODE_HEAPSIZE": 2048,  
          "HADOOP_NAMENODE_OPTS": "-XX:GCTimeRatio=19"  
        }  
      }  
    ]  
  }  
]
```

]

以下示例以本地文件`configurations.json`形式引用。

```
aws emr create-cluster \  
  --configurations file://configurations.json \  
  --release-label emr-5.9.0 \  
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large \  
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \  
  --auto-terminate
```

以下示例在 Amazon S3 中以文件`configurations.json`形式引用。

```
aws emr create-cluster \  
  --configurations https://s3.amazonaws.com/myBucket/configurations.json \  
  --release-label emr-5.9.0 \  
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large \  
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \  
  --auto-terminate
```

示例 9：创建包含主实例组、核心实例组和任务实例组的集群

以下`create-cluster`示例用于指定`--instance-groups`用于主实例组、核心实例组和任务实例组的 EC2 实例的类型和数量。

```
aws emr create-cluster \  
  --release-label emr-5.9.0 \  
  --instance-groups  
  Name=Master,InstanceGroupType=MASTER,InstanceType=m4.large,InstanceCount=1  
  Name=Core,InstanceGroupType=CORE,InstanceType=m4.large,InstanceCount=2  
  Name=Task,InstanceGroupType=TASK,InstanceType=m4.large,InstanceCount=2
```

示例 10：指定集群应在完成所有步骤后终止

以下`create-cluster`示例`--auto-terminate`用于指定集群应在完成所有步骤后自动关闭。

```
aws emr create-cluster \  
  --release-label emr-5.9.0 \  
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large \  
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \  
  --auto-terminate
```



示例 11：指定集群配置详细信息，例如 Amazon EC2 密钥对、网络配置和安全组

以下 `create-cluster` 示例使用名为 Amazon EC2 密钥对 `myKey` 和名为的自定义实例配置文件创建一个集群 `myProfile`。密钥对用于授权与群集节点（通常是主节点）的 SSH 连接。有关更多信息，请参阅 [《亚马逊 EMR 管理指南》](#) 中的“使用 Amazon EC2 密钥对获取 SSH 凭证”。

```
aws emr create-cluster \  
  --ec2-attributes KeyName=myKey,InstanceProfile=myProfile \  
  --release-label emr-5.9.0 \  
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large \  
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \  
  --auto-terminate
```

以下示例在 Amazon VPC 子网中创建了一个集群。

```
aws emr create-cluster \  
  --ec2-attributes SubnetId=subnet-xxxxx \  
  --release-label emr-5.9.0 \  
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large \  
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \  
  --auto-terminate
```

以下示例在 `us-east-1b` 可用区中创建集群。

```
aws emr create-cluster \  
  --ec2-attributes AvailabilityZone=us-east-1b \  
  --release-label emr-5.9.0 \  
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large \  
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large
```

以下示例创建了一个集群并仅指定了 Amazon EMR 托管的安全组。

```
aws emr create-cluster \  
  --release-label emr-5.9.0 \  
  --service-role myServiceRole \  
  --ec2-attributes InstanceProfile=myRole,EmrManagedMasterSecurityGroup=sg-  
  master1,EmrManagedSlaveSecurityGroup=sg-slave1 \  
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large \  
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large
```

以下示例创建了一个集群并仅指定了其他 Amazon EC2 安全组。

```
aws emr create-cluster \
  --release-label emr-5.9.0 \
  --service-role myServiceRole \
  --ec2-attributes InstanceProfile=myRole,AdditionalMasterSecurityGroups=[sg-
addMaster1,sg-addMaster2,sg-addMaster3,sg-
addMaster4],AdditionalSlaveSecurityGroups=[sg-addSlave1,sg-addSlave2,sg-
addSlave3,sg-addSlave4] \
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large
```

以下示例创建集群并指定 EMR 管理的安全组以及其他安全组。

```
aws emr create-cluster \
  --release-label emr-5.9.0 \
  --service-role myServiceRole \
  --ec2-attributes InstanceProfile=myRole,EmrManagedMasterSecurityGroup=sg-
master1,EmrManagedSlaveSecurityGroup=sg-slave1,AdditionalMasterSecurityGroups=[sg-
addMaster1,sg-addMaster2,sg-addMaster3,sg-
addMaster4],AdditionalSlaveSecurityGroups=[sg-addSlave1,sg-addSlave2,sg-
addSlave3,sg-addSlave4] \
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large
```

以下示例在 VPC 私有子网中创建集群，并使用特定的 Amazon EC2 安全组启用 Amazon EMR 服务访问权限，这是私有子网中的集群所必需的。

```
aws emr create-cluster \
  --release-label emr-5.9.0 \
  --service-role myServiceRole \
  --ec2-attributes InstanceProfile=myRole,ServiceAccessSecurityGroup=sg-service-
access,EmrManagedMasterSecurityGroup=sg-master,EmrManagedSlaveSecurityGroup=sg-slave
\
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large
```

以下示例使用存储在本地的名为 JSON 文件 `ec2_attributes.json` 来指定安全组配置参数。注意：JSON 参数必须将选项和值作为它们自己的项目包含在列表中。

```
aws emr create-cluster \
  --release-label emr-5.9.0 \
  --service-role myServiceRole \
```

```
--ec2-attributes file://ec2_attributes.json \
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large
```

ec2\_attributes.json 的内容：

```
[
  {
    "SubnetId": "subnet-xxxxxx",
    "KeyName": "myKey",
    "InstanceProfile": "myRole",
    "EmrManagedMasterSecurityGroup": "sg-master1",
    "EmrManagedSlaveSecurityGroup": "sg-slave1",
    "ServiceAccessSecurityGroup": "sg-service-access",
    "AdditionalMasterSecurityGroups": ["sg-addMaster1", "sg-addMaster2", "sg-
addMaster3", "sg-addMaster4"],
    "AdditionalSlaveSecurityGroups": ["sg-addSlave1", "sg-addSlave2", "sg-
addSlave3", "sg-addSlave4"]
  }
]
```

示例 12：启用调试并指定日志 URI

以下 create-cluster 示例使用 --enable-debugging 参数，该参数允许您使用 Amazon EMR 控制台中的调试工具更轻松地查看日志文件。--log-uri 参数是必需的 --enable-debugging。

```
aws emr create-cluster \
--enable-debugging \
--log-uri s3://myBucket/myLog \
--release-label emr-5.9.0 \
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \
--auto-terminate
```

示例 13：在创建集群时添加标签

标签是键值对，可帮助您识别和管理集群。以下 create-cluster 示例使用 --tags 参数为集群创建三个标签，一个带有密钥名称 name 和值 Shirley Rodriguez，第二个标签包含密钥名称 age 和值 29，第三个标签包含密钥名称 department 和值 Analytics。

```
aws emr create-cluster \
```

```
--tags name="Shirley Rodriguez" age=29 department="Analytics" \
--release-label emr-5.32.0 \
--instance-type m5.xlarge \
--instance-count 3 \
--use-default-roles
```

以下示例列出了应用于集群的标签。

```
aws emr describe-cluster \
  --cluster-id j-XXXXXXXXYY \
  --query Cluster.Tags
```

#### 示例 14：使用启用加密和其他安全功能的安全配置

以下create-cluster示例使用--security-configuration参数为 EMR 集群指定安全配置。您可以在 Amazon EMR 4.8.0 或更高版本中使用安全配置。

```
aws emr create-cluster \
  --instance-type m4.large \
  --release-label emr-5.9.0 \
  --security-configuration mySecurityConfiguration
```

#### 示例 15：创建具有为实例组配置的额外 EBS 存储卷的集群

指定其他 EBS 卷时，需要以下参数：VolumeType，SizeInGB如果EbsBlockDeviceConfigs已指定。

以下create-cluster示例创建了一个集群，其中有多 EBS 卷连接到核心实例组中的 EC2 实例。

```
aws emr create-cluster \
  --release-label emr-5.9.0 \
  --use-default-roles \
  --instance-groups
  InstanceGroupType=MASTER,InstanceCount=1,InstanceType=d2.xlarge
  'InstanceGroupType=CORE,InstanceCount=2,InstanceType=d2.xlarge,EbsConfiguration={EbsOptimiz
  {VolumeSpecification={VolumeType=io1,SizeInGB=100,Iops=100},VolumesPerInstance=4}}'
  \
  --auto-terminate
```

以下示例创建了一个集群，其中有多 EBS 卷连接到主实例组中的 EC2 实例。

```
aws emr create-cluster \
  --release-label emr-5.9.0 \
  --use-default-roles \
  --instance-groups 'InstanceGroupType=MASTER, InstanceCount=1,
InstanceType=d2.xlarge, EbsConfiguration={EbsOptimized=true,
EbsBlockDeviceConfigs=[{VolumeSpecification={VolumeType=io1, SizeInGB=100,
Iops=100}}],
{VolumeSpecification={VolumeType=standard, SizeInGB=50}, VolumesPerInstance=3}]}'
InstanceGroupType=CORE, InstanceCount=2, InstanceType=d2.xlarge \
  --auto-terminate
```

### 示例 16：使用自动扩展策略创建集群

您可以使用 Amazon EMR 版本 4.0 及更高版本将自动扩展策略附加到核心实例组和任务实例组。自动扩展策略会根据 Amazon CloudWatch 指标动态添加和删除 EC2 实例。有关更多信息，请参阅亚马逊 EMR 管理指南中的在亚马逊 EMR 中使用自动缩放 < <https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-automatic-scaling.html> >`\_。

附加自动扩展策略时，还必须使用 `--auto-scaling-role` `EMR_AutoScaling_DefaultRole` 指定自动扩展的默认角色。

以下 `create-cluster` 示例使用带有嵌入式 JSON 结构的 `AutoScalingPolicy` 参数指定 CORE 实例组的自动扩展策略，该结构指定了扩展策略配置。具有嵌入式 JSON 结构的实例组必须将整个参数集合用单引号括起来。对于没有嵌入式 JSON 结构的实例组，可选择使用单引号。

```
aws emr create-cluster
  --release-label emr-5.9.0 \
  --use-default-roles --auto-scaling-role EMR_AutoScaling_DefaultRole \
  --instance-groups
InstanceGroupType=MASTER, InstanceType=d2.xlarge, InstanceCount=1
'InstanceGroupType=CORE, InstanceType=d2.xlarge, InstanceCount=2, AutoScalingPolicy={Constrai
```

以下示例使用 JSON 文件来指定集群中所有实例组的配置。 `instancegroupconfig.json` JSON 文件指定了核心实例组的自动扩展策略配置。

```
aws emr create-cluster \
  --release-label emr-5.9.0 \
  --service-role EMR_DefaultRole \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
  --instance-groups file://myfolder/instancegroupconfig.json \
  --auto-scaling-role EMR_AutoScaling_DefaultRole
```

## instancegroupconfig.json 的内容：

```
[
  {
    "InstanceCount": 1,
    "Name": "MyMasterIG",
    "InstanceGroupType": "MASTER",
    "InstanceType": "m4.large"
  },
  {
    "InstanceCount": 2,
    "Name": "MyCoreIG",
    "InstanceGroupType": "CORE",
    "InstanceType": "m4.large",
    "AutoScalingPolicy": {
      "Constraints": {
        "MinCapacity": 2,
        "MaxCapacity": 10
      },
      "Rules": [
        {
          "Name": "Default-scale-out",
          "Description": "Replicates the default scale-out rule in the
console for YARN memory.",
          "Action": {
            "SimpleScalingPolicyConfiguration": {
              "AdjustmentType": "CHANGE_IN_CAPACITY",
              "ScalingAdjustment": 1,
              "CoolDown": 300
            }
          },
          "Trigger": {
            "CloudWatchAlarmDefinition": {
              "ComparisonOperator": "LESS_THAN",
              "EvaluationPeriods": 1,
              "MetricName": "YARNMemoryAvailablePercentage",
              "Namespace": "AWS/ElasticMapReduce",
              "Period": 300,
              "Threshold": 15,
              "Statistic": "AVERAGE",
              "Unit": "PERCENT",
              "Dimensions": [
                {
                  "Key": "JobFlowId",
```

```

    "Value": "${emr.clusterId}"
    ]
  }
]

```

### 示例 17：在创建集群时添加自定义 JAR 步骤

以下 `create-cluster` 示例通过指定存储在 Amazon S3 中的 JAR 文件来添加步骤。步骤将工作提交到集群。JAR 文件中定义的主函数将在配置 EC2 实例、执行所有引导操作并安装应用程序之后执行。这些步骤是使用指定的 `Type=CUSTOM_JAR`。

自定义 JAR 步骤需要 `Jar` 参数，该参数指定 JAR 的路径和文件名。可选参数有 `TypeName`、`ActionOnFailure`、`Args`、和 `MainClass`。如果未指定主类，则 JAR 文件应在其清单文件 `Main-Class` 中指定。

```

aws emr create-cluster \
    --steps Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,Jar=s3://
myBucket/mytest.jar,Args=arg1,arg2,arg3
    Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,Jar=s3://myBucket/
mytest.jar,MainClass=mymainclass,Args=arg1,arg2,arg3 \
    --release-label emr-5.3.1 \
    --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
    InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \
    --auto-terminate

```

### 示例 18：在创建集群时添加直播步骤

以下 `create-cluster` 示例向集群添加了一个流式处理步骤，该步骤将在所有步骤运行后终止。直播步骤需要参数 `Type` 和 `Args`。直播步骤可选参数为 `Name` 和 `ActionOnFailure`。

以下示例指定了行内步骤。

```

aws emr create-cluster \
    --steps Type=STREAMING,Name='Streaming Program',ActionOnFailure=CONTINUE,Args=[-
files,s3://elasticmapreduce/samples/wordcount/wordSplitter.py,-

```

```
mapper,wordSplitter.py,-reducer,aggregate,-input,s3://elasticmapreduce/samples/
wordcount/input,-output,s3://mybucket/wordcount/output] \
  --release-label emr-5.3.1 \
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \
  --auto-terminate
```

以下示例使用本地存储的名为的 JSON 配置文件multiplefiles.json。JSON 配置指定了多个文件。要在一个步骤中指定多个文件，必须使用 JSON 配置文件来指定该步骤。JSON 参数必须将选项和值作为它们自己的项目包含在列表中。

```
aws emr create-cluster \
  --steps file://./multiplefiles.json \
  --release-label emr-5.9.0 \
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \
  --auto-terminate
```

multiplefiles.json 的内容：

```
[
  {
    "Name": "JSON Streaming Step",
    "Args": [
      "-files",
      "s3://elasticmapreduce/samples/wordcount/wordSplitter.py",
      "-mapper",
      "wordSplitter.py",
      "-reducer",
      "aggregate",
      "-input",
      "s3://elasticmapreduce/samples/wordcount/input",
      "-output",
      "s3://mybucket/wordcount/output"
    ],
    "ActionOnFailure": "CONTINUE",
    "Type": "STREAMING"
  }
]
```

示例 19：在创建集群时添加 Hive 步骤



以下示例在创建集群时添加 Hive 步骤。Hive 步骤需要参数Type和。ArgsHive 步骤可选参数为Name和。ActionOnFailure

```
aws emr create-cluster \
  --steps Type=HIVE,Name='Hive
  program',ActionOnFailure=CONTINUE,ActionOnFailure=TERMINATE_CLUSTER,Args=[-
  f,s3://elasticmapreduce/samples/hive-ads/libs/model-build.q,-d,INPUT=s3://
  elasticmapreduce/samples/hive-ads/tables,-d,OUTPUT=s3://mybucket/hive-ads/
  output/2014-04-18/11-07-32,-d,LIBS=s3://elasticmapreduce/samples/hive-ads/libs] \
  --applications Name=Hive \
  --release-label emr-5.3.1 \
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large
```

示例 20：在创建集群时添加 Pig 步骤

以下示例在创建集群时添加了 Pig 步骤。Pig 步骤所需的参数是Type和Args。Pig steps 可选参数是Name和ActionOnFailure。

```
aws emr create-cluster \
  --steps Type=PIG,Name='Pig program',ActionOnFailure=CONTINUE,Args=[-f,s3://
  elasticmapreduce/samples/pig-apache/do-reports2.pig,-p,INPUT=s3://elasticmapreduce/
  samples/pig-apache/input,-p,OUTPUT=s3://mybucket/pig-apache/output] \
  --applications Name=Pig \
  --release-label emr-5.3.1 \
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large
```

示例 21：添加引导操作

以下create-cluster示例运行两个定义为存储在 Amazon S3 中的脚本的引导操作。

```
aws emr create-cluster \
  --bootstrap-actions Path=s3://mybucket/
  myscript1,Name=BootstrapAction1,Args=[arg1,arg2] Path=s3://mybucket/
  myscript2,Name=BootstrapAction2,Args=[arg1,arg2] \
  --release-label emr-5.3.1 \
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \
  --auto-terminate
```

示例 22：启用 EMRFS 一致视图并自定义和设置 RetryCount RetryPeriod

以下 `create-cluster` 示例指定 EMRFS 一致视图的重试次数和重试周期。 `Consistent=true` 是必需参数。

```
aws emr create-cluster \  
  --instance-type m4.large \  
  --release-label emr-5.9.0 \  
  --emrfs Consistent=true,RetryCount=6,RetryPeriod=30
```

以下示例使用本地存储的名为 `emrfsconfig.json` 的 JSON 配置文件指定了与前一个示例相同的 EMRFS 配置。

```
aws emr create-cluster \  
  --instance-type m4.large \  
  --release-label emr-5.9.0 \  
  --emrfs file://emrfsconfig.json
```

`emrfsconfig.json` 的内容：

```
{  
  "Consistent": true,  
  "RetryCount": 6,  
  "RetryPeriod": 30  
}
```

示例 23：创建配置了 Kerberos 的集群

以下 `create-cluster` 示例使用启用了 Kerberos 的安全配置创建集群，并使用为集群建立 Kerberos 参数。 `--kerberos-attributes`

以下命令以内联方式指定集群的 Kerberos 属性。

```
aws emr create-cluster \  
  --instance-type m3.xlarge \  
  --release-label emr-5.10.0 \  
  --service-role EMR_DefaultRole \  
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \  
  --security-configuration mySecurityConfiguration \  
  --kerberos-attributes  
  Realm=EC2.INTERNAL,KdcAdminPassword=123,CrossRealmTrustPrincipalPassword=123
```

以下命令指定了相同的属性，但引用了名为的本地存储的 JSON 文件 `kerberos_attributes.json`。在此示例中，文件保存在您运行命令的同一目录中。您也可以参考保存在 Amazon S3 中的配置文件。

```
aws emr create-cluster \  
  --instance-type m3.xlarge \  
  --release-label emr-5.10.0 \  
  --service-role EMR_DefaultRole \  
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \  
  --security-configuration mySecurityConfiguration \  
  --kerberos-attributes file://kerberos_attributes.json
```

`kerberos_attributes.json` 的内容：

```
{  
  "Realm": "EC2.INTERNAL",  
  "KdcAdminPassword": "123",  
  "CrossRealmTrustPrincipalPassword": "123",  
}
```

以下 `create-cluster` 示例创建了一个使用该 `--instance-groups` 配置并具有托管扩展策略的 Amazon EMR 集群。

```
aws emr create-cluster \  
  --release-label emr-5.30.0 \  
  --service-role EMR_DefaultRole \  
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \  
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large  
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large  
  --managed-scaling-policy  
  ComputeLimits='{MinimumCapacityUnits=2,MaximumCapacityUnits=4,UnitType=Instances}'
```

以下 `create-cluster` 示例创建了一个 Amazon EMR 集群，该集群使用 `--log-encryption-kms-key-id` 来定义用于日志加密的 KMS 密钥 ID。

```
aws emr create-cluster \  
  --release-label emr-5.30.0 \  
  --log-uri s3://myBucket/myLog \  
  --log-encryption-kms-key-id arn:aws:kms:us-east-1:110302272565:key/  
  dd559181-283e-45d7-99d1-66da348c4d33 \  
  \
```

```
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large
```

以下 `create-cluster` 示例创建了一个 Amazon EMR 集群，该集群使用 “`--placement-group-configs`” 配置使用 SPREAD 放置策略将主节点放置在 EC2 置放群组内的高可用性 (HA) 集群中。

```
aws emr create-cluster \
  --release-label emr-5.30.0 \
  --service-role EMR_DefaultRole \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
  --instance-groups
InstanceGroupType=MASTER,InstanceCount=3,InstanceType=m4.largeInstanceGroupType=CORE,Instan
\
  --placement-group-configs InstanceRole=MASTER
```

以下 `create-cluster` 示例创建了一个 Amazon EMR 集群，该集群使用 “`--auto-termination-policy`” 配置为集群设置自动空闲终止阈值。

```
aws emr create-cluster \
  --release-label emr-5.34.0 \
  --service-role EMR_DefaultRole \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
InstanceGroupType=CORE,InstanceCount=1,InstanceType=m4.large \
  --auto-termination-policy IdleTimeout=100
```

以下 `create-cluster` 示例创建了一个 Amazon EMR 集群，该集群使用 “`--os-release-label`” 来定义用于集群启动的 Amazon Linux 版本

```
aws emr create-cluster \
  --release-label emr-6.6.0 \
  --os-release-label 2.0.20220406.1 \
  --service-role EMR_DefaultRole \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large
InstanceGroupType=CORE,InstanceCount=1,InstanceType=m4.large
```

**示例 24：指定 EBS 根卷属性：使用 EMR 6.15.0 及更高版本创建的集群实例的大小、IOPS 和吞吐量**

以下 `create-cluster` 示例创建了一个 Amazon EMR 集群，该集群使用根卷属性为 EC2 实例配置根卷规范。

```
aws emr create-cluster \  
  --name "Cluster with My Custom AMI" \  
  --custom-ami-id ami-a518e6df \  
  --ebs-root-volume-size 20 \  
  --ebs-root-volume-iops 3000 \  
  --ebs-root-volume-throughput 125 \  
  --release-label emr-6.15.0 \  
  --use-default-roles \  
  --instance-count 2 \  
  --instance-type m4.large
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateClusterExamples](#) 中的。

## create-default-roles

以下代码示例演示了如何使用 `create-default-roles`。

### AWS CLI

1. 为 EC2 创建默认 IAM 角色

命令:

```
aws emr create-default-roles
```

输出:

If the role already exists then the command returns nothing.

If the role does not exist then the output will be:

```
[  
  {  
    "RolePolicy": {  
      "Version": "2012-10-17",  
      "Statement": [  
        {  
          "Action": [  
            "cloudwatch:*",
```

```

        "dynamodb:*",
        "ec2:Describe*",
        "elasticmapreduce:Describe*",
        "elasticmapreduce:ListBootstrapActions",
        "elasticmapreduce:ListClusters",
        "elasticmapreduce:ListInstanceGroups",
        "elasticmapreduce:ListInstances",
        "elasticmapreduce:ListSteps",
        "kinesis:CreateStream",
        "kinesis>DeleteStream",
        "kinesis:DescribeStream",
        "kinesis:GetRecords",
        "kinesis:GetShardIterator",
        "kinesis:MergeShards",
        "kinesis:PutRecord",
        "kinesis:SplitShard",
        "rds:Describe*",
        "s3:*",
        "sdb:*",
        "sns:*",
        "sqs:*"
    ],
    "Resource": "*",
    "Effect": "Allow"
}
]
},
"Role": {
    "AssumeRolePolicyDocument": {
        "Version": "2008-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Sid": "",
                "Effect": "Allow",
                "Principal": {
                    "Service": "ec2.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AR0AIQ5SIQUGL5KMYBJX6",
    "CreateDate": "2015-06-09T17:09:04.602Z",
    "RoleName": "EMR_EC2_DefaultRole",

```

```
    "Path": "/",
    "Arn": "arn:aws:iam::176430881729:role/EMR_EC2_DefaultRole"
  }
},
{
  "RolePolicy": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": [
          "ec2:AuthorizeSecurityGroupIngress",
          "ec2:CancelSpotInstanceRequests",
          "ec2:CreateSecurityGroup",
          "ec2:CreateTags",
          "ec2>DeleteTags",
          "ec2:DescribeAvailabilityZones",
          "ec2:DescribeAccountAttributes",
          "ec2:DescribeInstances",
          "ec2:DescribeInstanceStatus",
          "ec2:DescribeKeyPairs",
          "ec2:DescribePrefixLists",
          "ec2:DescribeRouteTables",
          "ec2:DescribeSecurityGroups",
          "ec2:DescribeSpotInstanceRequests",
          "ec2:DescribeSpotPriceHistory",
          "ec2:DescribeSubnets",
          "ec2:DescribeVpcAttribute",
          "ec2:DescribeVpcEndpoints",
          "ec2:DescribeVpcEndpointServices",
          "ec2:DescribeVpcs",
          "ec2:ModifyImageAttribute",
          "ec2:ModifyInstanceAttribute",
          "ec2:RequestSpotInstances",
          "ec2:RunInstances",
          "ec2:TerminateInstances",
          "iam:GetRole",
          "iam:GetRolePolicy",
          "iam:ListInstanceProfiles",
          "iam:ListRolePolicies",
          "iam:PassRole",
          "s3:CreateBucket",
          "s3:Get*",
          "s3:List*",
          "sdb:BatchPutAttributes",
```

```

        "sdb:Select",
        "sqs:CreateQueue",
        "sqs>Delete*",
        "sqs:GetQueue*",
        "sqs:ReceiveMessage"
    ],
    "Resource": "*",
    "Effect": "Allow"
  }
]
},
"Role": {
  "AssumeRolePolicyDocument": {
    "Version": "2008-10-17",
    "Statement": [
      {
        "Action": "sts:AssumeRole",
        "Sid": "",
        "Effect": "Allow",
        "Principal": {
          "Service": "elasticmapreduce.amazonaws.com"
        }
      }
    ]
  },
  "RoleId": "AR0AI3SRVPPVSRDLARBPY",
  "CreateDate": "2015-06-09T17:09:10.401Z",
  "RoleName": "EMR_DefaultRole",
  "Path": "/",
  "Arn": "arn:aws:iam::176430881729:role/EMR_DefaultRole"
}
]

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDefaultRoles](#)中的。

## create-security-configuration

以下代码示例演示了如何使用 create-security-configuration。



## AWS CLI

1. 创建安全配置，启用传输中加密，证书提供者使用 PEM 启用 PEM，对于 S3 加密，使用 SSE-S3 启用静态加密，本地磁盘密 AWS 密钥提供者使用 -KMS 启用静态加密

命令:

```
aws emr create-security-configuration --name MySecurityConfig --security-configuration '{
  "EncryptionConfiguration": {
    "EnableInTransitEncryption" : true,
    "EnableAtRestEncryption" : true,
    "InTransitEncryptionConfiguration" : {
      "TLSCertificateConfiguration" : {
        "CertificateProviderType" : "PEM",
        "S3Object" : "s3://mycertstore/artifacts/MyCerts.zip"
      }
    },
    "AtRestEncryptionConfiguration" : {
      "S3EncryptionConfiguration" : {
        "EncryptionMode" : "SSE-S3"
      },
      "LocalDiskEncryptionConfiguration" : {
        "EncryptionKeyProviderType" : "AwsKms",
        "AwsKmsKey" : "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
      }
    }
  }
}'
```

输出:

```
{
  "CreationDateTime": 1474070889.129,
  "Name": "MySecurityConfig"
}
```

JSON 等效项 (安全配置.json 的内容):

```
{
  "EncryptionConfiguration": {
```

```

    "EnableInTransitEncryption": true,
    "EnableAtRestEncryption": true,
    "InTransitEncryptionConfiguration": {
      "TLSCertificateConfiguration": {
        "CertificateProviderType": "PEM",
        "S3Object": "s3://mycertstore/artifacts/MyCerts.zip"
      }
    },
    "AtRestEncryptionConfiguration": {
      "S3EncryptionConfiguration": {
        "EncryptionMode": "SSE-S3"
      },
      "LocalDiskEncryptionConfiguration": {
        "EncryptionKeyProviderType": "AwsKms",
        "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
      }
    }
  }
}

```

命令 ( 使用 security\_configuration.json ) :

```
aws emr create-security-configuration --name "MySecurityConfig" --security-
configuration file://./security_configuration.json
```

输出 :

```
{
  "CreationDateTime": 1474070889.129,
  "Name": "MySecurityConfig"
}
```

## 2. 使用集群专用 KDC 和跨领域信任创建启用 Kerberos 的安全配置

命令:

```
aws emr create-security-configuration --name MySecurityConfig --security-
configuration '{
  "AuthenticationConfiguration": {
    "KerberosConfiguration": {
      "Provider": "ClusterDedicatedKdc",
      "ClusterDedicatedKdcConfiguration": {
```

```

        "TicketLifetimeInHours": 24,
        "CrossRealmTrustConfiguration": {
            "Realm": "AD.DOMAIN.COM",
            "Domain": "ad.domain.com",
            "AdminServer": "ad.domain.com",
            "KdcServer": "ad.domain.com"
        }
    }
}'

```

输出：

```

{
  "CreationDateTime": 1490225558.982,
  "Name": "MySecurityConfig"
}

```

JSON 等效项 ( 安全配置.json 的内容 )：

```

{
  "AuthenticationConfiguration": {
    "KerberosConfiguration": {
      "Provider": "ClusterDedicatedKdc",
      "ClusterDedicatedKdcConfiguration": {
        "TicketLifetimeInHours": 24,
        "CrossRealmTrustConfiguration": {
          "Realm": "AD.DOMAIN.COM",
          "Domain": "ad.domain.com",
          "AdminServer": "ad.domain.com",
          "KdcServer": "ad.domain.com"
        }
      }
    }
  }
}

```

命令 ( 使用 security\_configuration.json )：

```

aws emr create-security-configuration --name "MySecurityConfig" --security-
configuration file://./security_configuration.json

```

输出：

```
{
  "CreationDateTime": 1490225558.982,
  "Name": "MySecurityConfig"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSecurityConfiguration](#)中的。

## delete-security-configuration

以下代码示例演示了如何使用 delete-security-configuration。

AWS CLI

删除当前区域中的安全配置

命令：

```
aws emr delete-security-configuration --name MySecurityConfig
```

输出：

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSecurityConfiguration](#)中的。

## describe-cluster

以下代码示例演示了如何使用 describe-cluster。

AWS CLI

命令：

```
aws emr describe-cluster --cluster-id j-XXXXXXXX
```

输出：

```
For release-label based uniform instance groups cluster:
```

```
{
  "Cluster": {
    "Status": {
      "Timeline": {
        "ReadyDateTime": 1436475075.199,
        "CreationDateTime": 1436474656.563,
      },
      "State": "WAITING",
      "StateChangeReason": {
        "Message": "Waiting for steps to run"
      }
    },
    "Ec2InstanceAttributes": {
      "ServiceAccessSecurityGroup": "sg-xxxxxxx",
      "EmrManagedMasterSecurityGroup": "sg-xxxxxxx",
      "IamInstanceProfile": "EMR_EC2_DefaultRole",
      "Ec2KeyName": "myKey",
      "Ec2AvailabilityZone": "us-east-1c",
      "EmrManagedSlaveSecurityGroup": "sg-yyyyyyyyy"
    },
    "Name": "My Cluster",
    "ServiceRole": "EMR_DefaultRole",
    "Tags": [],
    "TerminationProtected": true,
    "UnhealthyNodeReplacement": true,
    "ReleaseLabel": "emr-4.0.0",
    "NormalizedInstanceHours": 96,
    "InstanceGroups": [
      {
        "RequestedInstanceCount": 2,
        "Status": {
          "Timeline": {
            "ReadyDateTime": 1436475074.245,
            "CreationDateTime": 1436474656.564,
            "EndDateTime": 1436638158.387
          },
          "State": "RUNNING",
          "StateChangeReason": {
            "Message": ""
          }
        },
        "Name": "CORE",
        "InstanceGroupType": "CORE",
        "Id": "ig-YYYYYYY",
      }
    ]
  }
}
```

```
    "Configurations": [],
    "InstanceType": "m3.large",
    "Market": "ON_DEMAND",
    "RunningInstanceCount": 2
  },
  {
    "RequestedInstanceCount": 1,
    "Status": {
      "Timeline": {
        "ReadyDateTime": 1436475074.245,
        "CreationDateTime": 1436474656.564,
        "EndDateTime": 1436638158.387
      },
      "State": "RUNNING",
      "StateChangeReason": {
        "Message": ""
      }
    },
    "Name": "MASTER",
    "InstanceGroupType": "MASTER",
    "Id": "ig-XXXXXXXXX",
    "Configurations": [],
    "InstanceType": "m3.large",
    "Market": "ON_DEMAND",
    "RunningInstanceCount": 1
  }
],
"Applications": [
  {
    "Name": "Hadoop"
  }
],
"VisibleToAllUsers": true,
"BootstrapActions": [],
"MasterPublicDnsName": "ec2-54-147-144-78.compute-1.amazonaws.com",
"AutoTerminate": false,
"Id": "j-XXXXXXXXX",
"Configurations": [
  {
    "Properties": {
      "fs.s3.consistent.retryPeriodSeconds": "20",
      "fs.s3.enableServerSideEncryption": "true",
      "fs.s3.consistent": "false",
      "fs.s3.consistent.retryCount": "2"
    }
  }
]
```

```

        },
        "Classification": "emrfs-site"
    }
]
}
}

```

For release-label based instance fleet cluster:

```

{
  "Cluster": {
    "Status": {
      "Timeline": {
        "ReadyDateTime": 1487897289.705,
        "CreationDateTime": 1487896933.942
      },
      "State": "WAITING",
      "StateChangeReason": {
        "Message": "Waiting for steps to run"
      }
    },
    "Ec2InstanceAttributes": {
      "EmrManagedMasterSecurityGroup": "sg-xxxxx",
      "RequestedEc2AvailabilityZones": [],
      "RequestedEc2SubnetIds": [],
      "IamInstanceProfile": "EMR_EC2_DefaultRole",
      "Ec2AvailabilityZone": "us-east-1a",
      "EmrManagedSlaveSecurityGroup": "sg-xxxxx"
    },
    "Name": "My Cluster",
    "ServiceRole": "EMR_DefaultRole",
    "Tags": [],
    "TerminationProtected": false,
    "UnhealthyNodeReplacement": false,
    "ReleaseLabel": "emr-5.2.0",
    "NormalizedInstanceHours": 472,
    "InstanceCollectionType": "INSTANCE_FLEET",
    "InstanceFleets": [
      {
        "Status": {
          "Timeline": {
            "ReadyDateTime": 1487897212.74,
            "CreationDateTime": 1487896933.948
          },

```

```

        "State": "RUNNING",
        "StateChangeReason": {
            "Message": ""
        }
    },
    "ProvisionedSpotCapacity": 1,
    "Name": "MASTER",
    "InstanceFleetType": "MASTER",
    "LaunchSpecifications": {
        "SpotSpecification": {
            "TimeoutDurationMinutes": 60,
            "TimeoutAction": "TERMINATE_CLUSTER"
        }
    },
    "TargetSpotCapacity": 1,
    "ProvisionedOnDemandCapacity": 0,
    "InstanceTypeSpecifications": [
        {
            "BidPrice": "0.5",
            "InstanceType": "m3.xlarge",
            "WeightedCapacity": 1
        }
    ],
    "Id": "if-xxxxxxx",
    "TargetOnDemandCapacity": 0
}
],
"Applications": [
    {
        "Version": "2.7.3",
        "Name": "Hadoop"
    }
],
"ScaleDownBehavior": "TERMINATE_AT_INSTANCE_HOUR",
"VisibleToAllUsers": true,
"BootstrapActions": [],
"MasterPublicDnsName": "ec2-xxx-xx-xxx-xx.compute-1.amazonaws.com",
"AutoTerminate": false,
"Id": "j-xxxxx",
"Configurations": []
}
}

```

For ami based uniform instance group cluster:



```
{
  "Cluster": {
    "Status": {
      "Timeline": {
        "ReadyDateTime": 1399400564.432,
        "CreationDateTime": 1399400268.62
      },
      "State": "WAITING",
      "StateChangeReason": {
        "Message": "Waiting for steps to run"
      }
    },
    "Ec2InstanceAttributes": {
      "IamInstanceProfile": "EMR_EC2_DefaultRole",
      "Ec2AvailabilityZone": "us-east-1c"
    },
    "Name": "My Cluster",
    "Tags": [],
    "TerminationProtected": true,
    "UnhealthyNodeReplacement": true,
    "RunningAmiVersion": "2.5.4",
    "InstanceGroups": [
      {
        "RequestedInstanceCount": 1,
        "Status": {
          "Timeline": {
            "ReadyDateTime": 1399400558.848,
            "CreationDateTime": 1399400268.621
          },
          "State": "RUNNING",
          "StateChangeReason": {
            "Message": ""
          }
        },
        "Name": "Master instance group",
        "InstanceGroupType": "MASTER",
        "InstanceType": "m1.small",
        "Id": "ig-ABCD",
        "Market": "ON_DEMAND",
        "RunningInstanceCount": 1
      },
      {
        "RequestedInstanceCount": 2,
```

```
    "Status": {
      "Timeline": {
        "ReadyDateTime": 1399400564.439,
        "CreationDateTime": 1399400268.621
      },
      "State": "RUNNING",
      "StateChangeReason": {
        "Message": ""
      }
    },
    "Name": "Core instance group",
    "InstanceGroupType": "CORE",
    "InstanceType": "m1.small",
    "Id": "ig-DEF",
    "Market": "ON_DEMAND",
    "RunningInstanceCount": 2
  }
],
"Applications": [
  {
    "Version": "1.0.3",
    "Name": "hadoop"
  }
],
"BootstrapActions": [],
"VisibleToAllUsers": false,
"RequestedAmiVersion": "2.4.2",
"LogUri": "s3://myLogUri/",
"AutoTerminate": false,
"Id": "j-XXXXXXXX"
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCluster](#)中的。

## describe-step

以下代码示例演示了如何使用 describe-step。

### AWS CLI

以下命令描述集群中步骤 ID 为 s-3LZC0QUT43AM 和集群 ID 为 j-3SD91U2E1L2QX 的步骤：

```
aws emr describe-step --cluster-id j-3SD91U2E1L2QX --step-id s-3LZC0QUT43AM
```

输出：

```
{
  "Step": {
    "Status": {
      "Timeline": {
        "EndDateTime": 1433200470.481,
        "CreationDateTime": 1433199926.597,
        "StartDateTime": 1433200404.959
      },
      "State": "COMPLETED",
      "StateChangeReason": {}
    },
    "Config": {
      "Args": [
        "s3://us-west-2.elasticmapreduce/libs/hive/hive-script",
        "--base-path",
        "s3://us-west-2.elasticmapreduce/libs/hive/",
        "--install-hive",
        "--hive-versions",
        "0.13.1"
      ],
      "Jar": "s3://us-west-2.elasticmapreduce/libs/script-runner/script-runner.jar",
      "Properties": {}
    },
    "Id": "s-3LZC0QUT43AM",
    "ActionOnFailure": "TERMINATE_CLUSTER",
    "Name": "Setup hive"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStep](#)中的。

## get

以下代码示例演示了如何使用 get。

## AWS CLI

以下内容从集群中的主实例下载具有集群 ID 的 `hadoop-examples.jar` 档案 `j-3SD91U2E1L2QX` :

```
aws emr get --cluster-id j-3SD91U2E1L2QX --key-pair-file ~/.ssh/mykey.pem --src /home/hadoop-examples.jar --dest ~
```

- 有关 API 的详细信息，请参阅 [Get](#) in AWS CLI 命令参考。

## list-clusters

以下代码示例演示了如何使用 `list-clusters`。

## AWS CLI

以下命令列出了当前区域中所有活动的 EMR 集群：

```
aws emr list-clusters --active
```

输出：

```
{
  "Clusters": [
    {
      "Status": {
        "Timeline": {
          "ReadyDateTime": 1433200405.353,
          "CreationDateTime": 1433199926.596
        },
        "State": "WAITING",
        "StateChangeReason": {
          "Message": "Waiting after step completed"
        }
      },
      "NormalizedInstanceHours": 6,
      "Id": "j-3SD91U2E1L2QX",
      "Name": "my-cluster"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListClusters](#)中的。

## list-instance-fleets

以下代码示例演示了如何使用 `list-instance-fleets`。

### AWS CLI

获取集群中实例队列的配置详细信息

此示例列出了指定集群中实例队列的详细信息。

命令:

```
list-instance-fleets --cluster-id 'j-12ABCDEFGHI34JK'
```

输出:

```
{
  "InstanceFleets": [
    {
      "Status": {
        "Timeline": {
          "ReadyDateTime": 1488759094.637,
          "CreationDateTime": 1488758719.817
        },
        "State": "RUNNING",
        "StateChangeReason": {
          "Message": ""
        }
      },
      "ProvisionedSpotCapacity": 6,
      "Name": "CORE",
      "InstanceFleetType": "CORE",
      "LaunchSpecifications": {
        "SpotSpecification": {
          "TimeoutDurationMinutes": 60,
          "TimeoutAction": "TERMINATE_CLUSTER"
        }
      },
      "ProvisionedOnDemandCapacity": 2,
      "InstanceTypeSpecifications": [
        {
```

```

        "BidPrice": "0.5",
        "InstanceType": "m3.xlarge",
        "WeightedCapacity": 2
    }
],
    "Id": "if-1ABC2DEFGHIJ3"
},
{
    "Status": {
        "Timeline": {
            "ReadyDateTime": 1488759058.598,
            "CreationDateTime": 1488758719.811
        },
        "State": "RUNNING",
        "StateChangeReason": {
            "Message": ""
        }
    },
    "ProvisionedSpotCapacity": 0,
    "Name": "MASTER",
    "InstanceFleetType": "MASTER",
    "ProvisionedOnDemandCapacity": 1,
    "InstanceTypeSpecifications": [
        {
            "BidPriceAsPercentageOfOnDemandPrice": 100.0,
            "InstanceType": "m3.xlarge",
            "WeightedCapacity": 1
        }
    ],
    "Id": "if-2ABC4DEFGHIJ4"
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListInstanceFleets](#)中的。

## list-instances

以下代码示例演示了如何使用 list-instances。

### AWS CLI

以下命令列出了集群中所有具有集群 ID 的实例j-3C6XNQ39VR9WL：

```
aws emr list-instances --cluster-id j-3C6XNQ39VR9WL
```

输出：

For a uniform instance group based cluster

```
{
  "Instances": [
    {
      "Status": {
        "Timeline": {
          "ReadyDateTime": 1433200400.03,
          "CreationDateTime": 1433199960.152
        },
        "State": "RUNNING",
        "StateChangeReason": {}
      },
      "Ec2InstanceId": "i-f19ecfee",
      "PublicDnsName": "ec2-52-52-41-150.us-west-2.compute.amazonaws.com",
      "PrivateDnsName": "ip-172-21-11-216.us-west-2.compute.internal",
      "PublicIpAddress": "52.52.41.150",
      "Id": "ci-3NNHQ2TWB6Y",
      "PrivateIpAddress": "172.21.11.216"
    },
    {
      "Status": {
        "Timeline": {
          "ReadyDateTime": 1433200400.031,
          "CreationDateTime": 1433199949.102
        },
        "State": "RUNNING",
        "StateChangeReason": {}
      },
      "Ec2InstanceId": "i-1feee4c2",
      "PublicDnsName": "ec2-52-63-246-32.us-west-2.compute.amazonaws.com",
      "PrivateDnsName": "ip-172-31-24-130.us-west-2.compute.internal",
      "PublicIpAddress": "52.63.246.32",
      "Id": "ci-GAOCMKNKDCV7",
      "PrivateIpAddress": "172.21.11.215"
    },
    {
      "Status": {
        "Timeline": {
          "ReadyDateTime": 1433200400.031,
```

```

        "CreationDateTime": 1433199949.102
      },
      "State": "RUNNING",
      "StateChangeReason": {}
    },
    "Ec2InstanceId": "i-15cfee3",
    "PublicDnsName": "ec2-52-25-246-63.us-west-2.compute.amazonaws.com",
    "PrivateDnsName": "ip-172-31-24-129.us-west-2.compute.internal",
    "PublicIpAddress": "52.25.246.63",
    "Id": "ci-2W3TDFFB47UAD",
    "PrivateIpAddress": "172.21.11.214"
  }
]
}

```

For a fleet based cluster:

```

{
  "Instances": [
    {
      "Status": {
        "Timeline": {
          "ReadyDateTime": 1487810810.878,
          "CreationDateTime": 1487810588.367,
          "EndDateTime": 1488022990.924
        },
        "State": "TERMINATED",
        "StateChangeReason": {
          "Message": "Instance was terminated."
        }
      },
      "Ec2InstanceId": "i-xxxxx",
      "InstanceFleetId": "if-xxxxx",
      "EbsVolumes": [],
      "PublicDnsName": "ec2-xx-xxx-xxx-xxx.compute-1.amazonaws.com",
      "InstanceType": "m3.xlarge",
      "PrivateDnsName": "ip-xx-xx-xxx-xx.ec2.internal",
      "Market": "SPOT",
      "PublicIpAddress": "xx.xx.xxx.xxx",
      "Id": "ci-xxxxx",
      "PrivateIpAddress": "10.47.191.80"
    }
  ]
}

```



```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListInstances](#)中的。

## list-security-configurations

以下代码示例演示了如何使用 list-security-configurations。

### AWS CLI

列出当前区域的安全配置

命令:

```
aws emr list-security-configurations
```

输出 :

```
{
  "SecurityConfigurations": [
    {
      "CreationDateTime": 1473889697.417,
      "Name": "MySecurityConfig-1"
    },
    {
      "CreationDateTime": 1473889697.417,
      "Name": "MySecurityConfig-2"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSecurityConfigurations](#)中的。

## list-steps

以下代码示例演示了如何使用 list-steps。

### AWS CLI

以下命令列出了集群 ID 为 j-3SD91U2E1L2QX 的集群的所有步骤：

```
aws emr list-steps --cluster-id j-3SD91U2E1L2QX
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSteps](#)中的。

## modify-cluster-attributes

以下代码示例演示了如何使用 `modify-cluster-attributes`。

### AWS CLI

以下命令将 ID 为 EMR 集群的可见性设置 `j-301CDNY0J5XM4` 为所有用户：

```
aws emr modify-cluster-attributes --cluster-id j-301CDNY0J5XM4 --visible-to-all-users
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyClusterAttributes](#)中的。

## modify-instance-fleet

以下代码示例演示了如何使用 `modify-instance-fleet`。

### AWS CLI

更改实例队列的目标容量

此示例将指定实例队列的按需和竞价目标容量更改为 1。

命令：

```
aws emr modify-instance-fleet --cluster-id 'j-12ABCDEFGH134JK' --instance-fleet InstanceFleetId='if-2ABC4DEFGHIJ4',TargetOnDemandCapacity=1,TargetSpotCapacity=1
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyInstanceFleet](#)中的。

## put

以下代码示例演示了如何使用 `put`。

## AWS CLI

以下命令将名为集群的主实例上传一个名为healthcheck.sh为集群的文件，该文件名为集群 ID：j-3SD91U2E1L2QX

```
aws emr put --cluster-id j-3SD91U2E1L2QX --key-pair-file ~/.ssh/mykey.pem --src ~/scripts/healthcheck.sh --dest /home/hadoop/bin/healthcheck.sh
```

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#)。

## remove-tags

以下代码示例演示了如何使用 remove-tags。

## AWS CLI

以下命令prod从集群 ID 为的集群中删除带有密钥的标签j-3SD91U2E1L2QX：

```
aws emr remove-tags --resource-id j-3SD91U2E1L2QX --tag-keys prod
```

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [RemoveTags](#) 中的。

## schedule-hbase-backup

以下代码示例演示了如何使用 schedule-hbase-backup。

## AWS CLI

注意：此命令只能与 AMI 版本 2.x 和 3.x 上的 HBase 一起使用

1. 要安排完整的 HBase 备份 >>>>> 0 6ab6d6e13564b5733d75abaf3b599f93cf39a23

命令：

```
aws emr schedule-hbase-backup --cluster-id j-XXXXXXYY --type full --dir s3://myBucket/backup --interval 10 --unit hours --start-time 2014-04-21T05:26:10Z --consistent
```

输出：

```
None
```

## 2. 计划增量 HBase 备份

命令:

```
aws emr schedule-hbase-backup --cluster-id j-XXXXXXYY --type incremental
--dir s3://myBucket/backup --interval 30 --unit minutes --start-time
2014-04-21T05:26:10Z --consistent
```

输出:

```
None
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ScheduleHbaseBackup](#) 中的。

## socks

以下代码示例演示了如何使用 socks。

### AWS CLI

以下命令使用集群 ID 打开与集群中主实例的 socks 连接 j-3SD91U2E1L2QX：

```
aws emr socks --cluster-id j-3SD91U2E1L2QX --key-pair-file ~/.ssh/mykey.pem
```

key pair file 选项采用私钥文件的本地路径。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [Socks](#)。

## ssh

以下代码示例演示了如何使用 ssh。

### AWS CLI

以下命令使用集群 ID 打开与集群中主实例的 ssh 连接 j-3SD91U2E1L2QX：

```
aws emr ssh --cluster-id j-3SD91U2E1L2QX --key-pair-file ~/.ssh/mykey.pem
```

key pair file 选项采用私钥文件的本地路径。

输出:

```
ssh -o StrictHostKeyChecking=no -o ServerAliveInterval=10 -i /home/local/user/.ssh/
mykey.pem hadoop@ec2-52-52-41-150.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-52-52-41-150.us-
west-2.compute.amazonaws.com,52.52.41.150' (ECDSA) to the list of known hosts.
Last login: Mon Jun  1 23:15:38 2015

  _|  _|_  )
  _| (    /  Amazon Linux AMI
  _|\__|__|

https://aws.amazon.com/amazon-linux-ami/2015.03-release-notes/
26 package(s) needed for security, out of 39 available
Run "sudo yum update" to apply all updates.

-----

Welcome to Amazon Elastic MapReduce running Hadoop and Amazon Linux.

Hadoop is installed in /home/hadoop. Log files are in /mnt/var/log/hadoop. Check
/mnt/var/log/hadoop/steps for diagnosing step failures.

The Hadoop UI can be accessed via the following commands:

ResourceManager    lynx http://ip-172-21-11-216:9026/
NameNode           lynx http://ip-172-21-11-216:9101/

-----

[hadoop@ip-172-31-16-216 ~]$
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [SSH](#)。

## EKS 上的 Amazon EMR 示例使用 AWS CLI

以下代码示例向您展示了如何在 EKS 上使用 Amazon EMR 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### update-role-trust-policy

以下代码示例演示了如何使用 `update-role-trust-policy`。

#### AWS CLI

更新要与 EKS 上的 Amazon EMR 一起使用的 IAM 角色的信任策略

此示例命令更新名为 `example_iam_role` 的角色的信任策略，以便该角色可以与名为 `example_cluster` 的 EKS 集群中的 `example_namespace` 命名空间的 EKS 上的 Amazon EMR 一起使用。

命令:

```
aws emr-containers update-role-trust-policy \
  --cluster example_cluster \
  --namespace example_namespace \
  --role-name example_iam_role
```

输出:

```
If the trust policy has already been updated, then the output will be:
Trust policy statement already exists for role example_iam_role. No
changes were made!
```

```
If the trust policy has not been updated yet, then the output will be:
Successfully updated trust policy of role example_iam_role.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRoleTrustPolicy](#)中的。

## EventBridge 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 EventBridge。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## **delete-rule**

以下代码示例演示了如何使用 delete-rule。

AWS CLI

删除 CloudWatch 事件规则

此示例删除名为 EC2 的规则 InstanceStateChanges：

```
aws events delete-rule --name "EC2InstanceStateChanges"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRule](#)中的。

## **describe-rule**

以下代码示例演示了如何使用 describe-rule。

AWS CLI

显示有关 CloudWatch 事件规则的信息

此示例显示有关名为 DailyLambdaFunction：的规则的信息

```
aws events describe-rule --name "DailyLambdaFunction"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeRule](#)中的。

## disable-rule

以下代码示例演示了如何使用 `disable-rule`。

### AWS CLI

禁用 CloudWatch 事件规则

此示例禁用名为 `DailyLambdaFunction` 的规则。该规则未删除：

```
aws events disable-rule --name "DailyLambdaFunction"
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisableRule](#) 中的。

## enable-rule

以下代码示例演示了如何使用 `enable-rule`。

### AWS CLI

启用 CloudWatch 事件规则

此示例启用名为 `DailyLambdaFunction` 的规则，该规则以前已被禁用：

```
aws events enable-rule --name "DailyLambdaFunction"
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [EnableRule](#) 中的。

## list-rule-names-by-target

以下代码示例演示了如何使用 `list-rule-names-by-target`。

### AWS CLI

显示具有指定目标的所有规则

此示例显示了所有以名为 `"MyFunctionName"` 的 Lambda 函数作为目标的规则：

```
aws events list-rule-names-by-target --target-arn "arn:aws:lambda:us-east-1:123456789012:function:MyFunctionName"
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListRuleNamesByTarget](#) 中的。



## list-rules

以下代码示例演示了如何使用 `list-rules`。

### AWS CLI

显示所有 CloudWatch 事件规则的列表

此示例显示该区域的所有 CloudWatch 事件规则：

```
aws events list-rules
```

显示以特定字符串开头 CloudWatch 的事件规则列表。

此示例显示该区域中名称以 “Daily” 开头的 CloudWatch 事件规则：

```
aws events list-rules --name-prefix "Daily"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRules](#)中的。

## list-targets-by-rule

以下代码示例演示了如何使用 `list-targets-by-rule`。

### AWS CLI

显示 CloudWatch 事件规则的所有目标

此示例显示名为 `DailyLambdaFunction` 的规则的所有目标：

```
aws events list-targets-by-rule --rule "DailyLambdaFunction"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTargetsByRule](#)中的。

## put-events

以下代码示例演示了如何使用 `put-events`。

### AWS CLI

向 Events 发送自定义 CloudWatch 事件

此示例向 Events 发送自定义 CloudWatch 事件。该事件包含在 putevents.json 文件中：

```
aws events put-events --entries file://putevents.json
```

以下是 putevents.json 文件的内容：

```
[
  {
    "Source": "com.mycompany.myapp",
    "Detail": "{ \"key1\": \"value1\", \"key2\": \"value2\" }",
    "Resources": [
      "resource1",
      "resource2"
    ],
    "DetailType": "myDetailType"
  },
  {
    "Source": "com.mycompany.myapp",
    "Detail": "{ \"key1\": \"value3\", \"key2\": \"value4\" }",
    "Resources": [
      "resource1",
      "resource2"
    ],
    "DetailType": "myDetailType"
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutEvents](#)中的。

## put-rule

以下代码示例演示了如何使用 put-rule。

### AWS CLI

#### 创建 CloudWatch 活动规则

该示例创建一条可在协调世界时每天上午 9:00 触发的规则。如果您使用 put-targets 将 Lambda 函数添加为该规则的目标，则可以在每天的指定时间运行该 Lambda 函数：

```
aws events put-rule --name "DailyLambdaFunction" --schedule-expression "cron(0 9 * * ? *)"
```

以下示例创建一条规则，将在区域中的任何 EC2 实例更改状态时触发该规则：

```
aws events put-rule --name "EC2InstanceStateChanges" --event-pattern "{\"source\": [\"aws.ec2\"], \"detail-type\": [\"EC2 Instance State-change Notification\"]}" --role-arn "arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

以下示例创建一条规则，将在区域中的任何 EC2 实例停止或终止时触发该规则：

```
aws events put-rule --name "EC2InstanceStateChangeStopOrTerminate" --event-pattern "{\"source\": [\"aws.ec2\"], \"detail-type\": [\"EC2 Instance State-change Notification\"], \"detail\": {\"state\": [\"stopped\", \"terminated\"]}}" --role-arn "arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutRule](#) 中的。

## put-targets

以下代码示例演示了如何使用 put-targets。

### AWS CLI

为 CloudWatch 事件规则添加目标

以下示例添加了一个 Lambda 函数作为规则的目标：

```
aws events put-targets --rule DailyLambdaFunction --targets "Id"="1", "Arn"="arn:aws:lambda:us-east-1:123456789012:function:MyFunctionName"
```

以下示例将 Amazon Kinesis 流设置为目标，以便将按此规则捕获的事件中继到该流：

```
aws events put-targets --rule EC2InstanceStateChanges --targets "Id"="1", "Arn"="arn:aws:kinesis:us-east-1:123456789012:stream/MyStream", "RoleArn"="arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

以下示例将两个 Amazon Kinesis 流设置为一条规则的目标：

```
aws events put-targets --rule DailyLambdaFunction --targets "Id"="Target1", "Arn"="arn:aws:kinesis:us-east-1:379642911888:stream/MyStream1", "RoleArn"="arn:aws:iam::379642911888:role/ MyRoleToAccessLambda" "Id"="Target2", " Arn"="arn:aws:kinesis:us-east-1:379642911888:stream/MyStream2", "RoleArn"="arn:aws:iam::379642911888:role/MyRoleToAccessLambda"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutTargets](#)中的。

## remove-targets

以下代码示例演示了如何使用 `remove-targets`。

### AWS CLI

#### 移除事件的目标

此示例将名为 `MyStream 1` 的 Amazon Kinesis 直播从规则的目标中移除。 `DailyLambdaFunction` 创建时 `DailyLambdaFunction`，此直播被设置为目标，ID 为 `Target1`：

```
aws events remove-targets --rule "DailyLambdaFunction" --ids "Target1"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveTargets](#)中的。

## test-event-pattern

以下代码示例演示了如何使用 `test-event-pattern`。

### AWS CLI

#### 检查事件模式是否与指定事件匹配

此示例测试模式 `source: com.mycompany.myapp` 是否与指定的事件匹配。在此示例中，输出将为 `true`：

```
aws events test-event-pattern --event-pattern "{\"source\": [\"com.mycompany.myapp \\\"]}" --event "{\"id\": \"1\", \"source\": \"com.mycompany.myapp\", \"detail-type\": \"myDetailType\", \"account\": \"123456789012\", \"region\": \"us-east-1\", \"time\": \"2017-04-11T20:11:04Z\"}"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TestEventPattern](#)中的。

## 使用 Firewall Manager 示例 AWS CLI

以下代码示例向您展示了如何使用与 Firewall Manager AWS Command Line Interface 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-admin-account**

以下代码示例演示了如何使用 `associate-admin-account`。

AWS CLI

设置 Firewall Manager 管理员帐户

以下 `associate-admin-account` 示例为 Firewall Manager 设置管理员帐户。

```
aws fms associate-admin-account \  
  --admin-account 123456789012
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS WAF、Firewal AWS I Manager 和 AWS Shield 高级开发者指南中的设置 AWS 防火墙管理器 [管理员帐户](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateAdminAccount](#) 中的。

### **delete-notification-channel**

以下代码示例演示了如何使用 `delete-notification-channel`。

AWS CLI

删除 Firewall Manager 日志的 SNS 主题信息

以下delete-notification-channel示例删除了 SNS 主题信息。

```
aws fms delete-notification-channel
```

此命令不生成任何输出。

有关更多信息，请参阅 [W A AWS F、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的配置 Amazon SNS 通知和亚马逊 CloudWatch 警报。](#)

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNotificationChannel](#)中的。

## delete-policy

以下代码示例演示了如何使用 delete-policy。

### AWS CLI

删除 Firewall Manager 策略

以下delete-policy示例删除了具有指定 ID 的策略及其所有资源。

```
aws fms delete-policy \  
  --policy-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --delete-all-policy-resources
```

此命令不生成任何输出。

有关更多信息，请参阅 [AWS WAF、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的使用 AWS 防火墙管理器策略。](#)

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePolicy](#)中的。

## disassociate-admin-account

以下代码示例演示了如何使用 disassociate-admin-account。

### AWS CLI

删除 Firewall Manager 管理员帐户

以下disassociate-admin-account示例从 Firewall Manager 中删除当前的管理员帐户关联。

```
aws fms disassociate-admin-account
```

此命令不生成任何输出。

有关更多信息，请参阅 [AWS WAF](#)、[Firewal AWS I Manager](#) 和 [AWS Shield 高级开发者指南](#) 中的 [设置 AWS 防火墙管理器管理员帐户](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 DisassociateAdminAccount](#) 中的。

## get-admin-account

以下代码示例演示了如何使用 `get-admin-account`。

### AWS CLI

检索 Firewall Manager 管理员帐户

以下 `get-admin-account` 示例检索管理员帐户。

```
aws fms get-admin-account
```

输出：

```
{
  "AdminAccount": "123456789012",
  "RoleStatus": "READY"
}
```

有关更多信息，请参阅 [AWS WAF](#)、[Firewal AWS I Manager](#) 和 [AWS Shield 高级开发者指南](#) 中的 [AWS 防火墙管理器先决条件](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 GetAdminAccount](#) 中的。

## get-compliance-detail

以下代码示例演示了如何使用 `get-compliance-detail`。

### AWS CLI

检索账户的合规信息

以下`get-compliance-detail`示例检索指定策略和成员账户的合规性信息。

```
aws fms get-compliance-detail \  
  --policy-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --member-account 123456789012
```

输出：

```
{  
  "PolicyComplianceDetail": {  
    "EvaluationLimitExceeded": false,  
    "IssueInfoMap": {},  
    "MemberAccount": "123456789012",  
    "PolicyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "PolicyOwner": "123456789012",  
    "Violators": []  
  }  
}
```

有关更多信息，请参阅 [AWS WAF、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的使用策略查看资源合规性](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `GetComplianceDetail`](#) 中的。

## `get-notification-channel`

以下代码示例演示了如何使用 `get-notification-channel`。

### AWS CLI

检索 Firewall Manager 日志的 SNS 主题信息

以下`get-notification-channel`示例检索 SNS 主题信息。

```
aws fms get-notification-channel
```

输出：

```
{  
  "SnsTopicArn": "arn:aws:sns:us-west-2:123456789012:us-west-2-fms",  
  "SnsRoleName": "arn:aws:iam::123456789012:role/aws-service-role/  
fms.amazonaws.com/AWSServiceRoleForFMS"
```



```
}
```

有关更多信息，请参阅 [W A AWS F、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的配置 Amazon SNS 通知和亚马逊 CloudWatch 警报。](#)

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetNotificationChannel](#) 中的。

## get-policy

以下代码示例演示了如何使用 get-policy。

### AWS CLI

检索 Firewall Manager 策略

以下 get-policy 示例检索具有指定 ID 的策略。

```
aws fms get-policy \
  --policy-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
  "Policy": {
    "PolicyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "PolicyName": "test",
    "PolicyUpdateToken": "1:p+2RpKR4wPFx7mcrL1U0QQ==",
    "SecurityServicePolicyData": {
      "Type": "SECURITY_GROUPS_COMMON",
      "ManagedServiceData": "{\"type\": \"SECURITY_GROUPS_COMMON\",
\\revertManualSecurityGroupChanges\": true, \\exclusiveResourceSecurityGroupManagement
\\: false, \\securityGroups\": [{\\id\": \"sg-045c43ccc9724e63e\"}]}"
    },
    "ResourceType": "AWS::EC2::Instance",
    "ResourceTags": [],
    "ExcludeResourceTags": false,
    "RemediationEnabled": false
  },
  "PolicyArn": "arn:aws:fms:us-west-2:123456789012:policy/d1ac59b8-938e-42b3-b2e0-7c620422ddc2"
}
```

有关更多信息，请参阅 AWS WAF、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的使用 AWS 防火墙管理器[策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetPolicy](#)中的。

## list-compliance-status

以下代码示例演示了如何使用 list-compliance-status。

### AWS CLI

检索成员账户的政策合规性信息

以下 list-compliance-status 示例检索指定策略的成员账户合规性信息。

```
aws fms list-compliance-status \  
  --policy-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "PolicyComplianceStatusList": [  
    {  
      "PolicyOwner": "123456789012",  
      "PolicyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "PolicyName": "test",  
      "MemberAccount": "123456789012",  
      "EvaluationResults": [  
        {  
          "ComplianceStatus": "COMPLIANT",  
          "ViolatorCount": 0,  
          "EvaluationLimitExceeded": false  
        },  
        {  
          "ComplianceStatus": "NON_COMPLIANT",  
          "ViolatorCount": 2,  
          "EvaluationLimitExceeded": false  
        }  
      ],  
      "LastUpdated": 1576283774.0,  
      "IssueInfoMap": {}  
    }  
  ]  
}
```

```
}
```

有关更多信息，请参阅 AWS WAF、AWS Firewall Manager 和 AWS Shield 高级开发者指南中的[使用策略查看资源合规性](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListComplianceStatus](#)中的。

## list-member-accounts

以下代码示例演示了如何使用 list-member-accounts。

### AWS CLI

检索组织中的成员账户

以下 list-member-accounts 示例列出了 Firewall Manager 管理员组织中的所有成员帐户。

```
aws fms list-member-accounts
```

输出：

```
{
  "MemberAccounts": [
    "222222222222",
    "333333333333",
    "444444444444"
  ]
}
```

有关更多信息，请参阅 AWS WAF、AWS Firewall Manager 和 AWS Shield 高级开发者指南中的 AWS 防火墙管理器。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListMemberAccounts](#)中的。

## list-policies

以下代码示例演示了如何使用 list-policies。

### AWS CLI

检索所有 Firewall Manager 策略

以下`list-policies`示例检索账户的策略列表。在此示例中，每个请求的输出限制为两个结果。每次调用都会返回一个`NextToken`，该值可用作下次`list-policies`调用中`--starting-token`参数的值，以获取列表的下一组结果。

```
aws fms list-policies \  
  --max-items 2
```

输出：

```
{  
  "PolicyList": [  
    {  
      "PolicyArn": "arn:aws:fms:us-west-2:123456789012:policy/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "PolicyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "PolicyName": "test",  
      "ResourceType": "AWS::EC2::Instance",  
      "SecurityServiceType": "SECURITY_GROUPS_COMMON",  
      "RemediationEnabled": false  
    },  
    {  
      "PolicyArn": "arn:aws:fms:us-west-2:123456789012:policy/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
      "PolicyId": "457c9b21-fc94-406c-ae63-21217395ba72",  
      "PolicyName": "test",  
      "ResourceType": "AWS::EC2::Instance",  
      "SecurityServiceType": "SECURITY_GROUPS_COMMON",  
      "RemediationEnabled": false  
    }  
  ],  
  "NextToken": "eyJ0ZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ=="  
}
```

有关更多信息，请参阅 [AWS WAF](#)、[Fi AWS rewall Manager](#) 和 [AWS Shield 高级开发者指南](#) 中的 [使用 AWS 防火墙管理器策略](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [ListPolicies](#) 中的。

## put-notification-channel

以下代码示例演示了如何使用 `put-notification-channel`。

## AWS CLI

设置 Firewall Manager 日志的 SNS 主题信息

以下put-notification-channel示例设置 SNS 主题信息。

```
aws fms put-notification-channel \  
  --sns-topic-arn arn:aws:sns:us-west-2:123456789012:us-west-2-fms \  
  --sns-role-name arn:aws:iam::123456789012:role/aws-service-role/  
fms.amazonaws.com/AWSServiceRoleForFMS
```

此命令不生成任何输出。

有关更多信息，请参阅 [W A AWS F、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的配置 Amazon SNS 通知和亚马逊 CloudWatch 警报。](#)

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutNotificationChannel](#)中的。

## put-policy

以下代码示例演示了如何使用 put-policy。

### AWS CLI

创建 Firewall Manager 策略

以下put-policy示例创建了 Firewall Manager 安全组策略。

```
aws fms put-policy \  
  --cli-input-json file://policy.json
```

policy.json 的内容：

```
{  
  "Policy": {  
    "PolicyName": "test",  
    "SecurityServicePolicyData": {  
      "Type": "SECURITY_GROUPS_USAGE_AUDIT",  
      "ManagedServiceData": "{\"type\":\"SECURITY_GROUPS_USAGE_AUDIT\",  
\\\"deleteUnusedSecurityGroups\\\":false,\\\"coalesceRedundantSecurityGroups\\\":true}"  
    },  
    "ResourceType": "AWS::EC2::SecurityGroup",  
    "ResourceTags": [],  
  },  
}
```

```

    "ExcludeResourceTags": false,
    "RemediationEnabled": false
  },
  "TagList": [
    {
      "Key": "foo",
      "Value": "foo"
    }
  ]
}

```

输出：

```

{
  "Policy": {
    "PolicyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "PolicyName": "test",
    "PolicyUpdateToken": "1:X9QGexP7HASDlsFp+G31Iw==",
    "SecurityServicePolicyData": {
      "Type": "SECURITY_GROUPS_USAGE_AUDIT",
      "ManagedServiceData": "{\"type\":\"SECURITY_GROUPS_USAGE_AUDIT\",
\\deleteUnusedSecurityGroups\":false,\\coalesceRedundantSecurityGroups\":true,
\\optionalDelayForUnusedInMinutes\":null}"
    },
    "ResourceType": "AWS::EC2::SecurityGroup",
    "ResourceTags": [],
    "ExcludeResourceTags": false,
    "RemediationEnabled": false
  },
  "PolicyArn": "arn:aws:fms:us-west-2:123456789012:policy/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}

```

有关更多信息，请参阅 AWS WAF、AWS Firewall Manager 和 AWS Shield 高级开发者指南中的使用 AWS 防火墙管理器 [策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutPolicy](#) 中的。

## AWS FIS 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS FIS。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## **create-experiment-template**

以下代码示例演示了如何使用 `create-experiment-template`。

AWS CLI

创建实验模板

以下 `create-experiment-template` 示例在您的 AWS FIS 账户中创建了一个实验模板。

```
aws fis create-experiment-template \  
  --cli-input-json file://myfile.json
```

myfile.json 的内容：

```
{  
  "description": "experimentTemplate",  
  "stopConditions": [  
    {  
      "source": "aws:cloudwatch:alarm",  
      "value": "arn:aws:cloudwatch:us-west-2:123456789012:alarm:alarmName"  
    }  
  ],  
  "targets": {  
    "Instances-Target-1": {  
      "resourceType": "aws:ec2:instance",  
      "resourceArns": [  
        "arn:aws:ec2:us-west-2:123456789012:instance/i-12a3b4c56d78e9012"  
      ]  
    }  
  }  
}
```

```

    ],
    "selectionMode": "ALL"
  }
},
"actions": {
  "reboot": {
    "actionId": "aws:ec2:reboot-instances",
    "description": "reboot",
    "parameters": {},
    "targets": {
      "Instances": "Instances-Target-1"
    }
  }
},
"roleArn": "arn:aws:iam::123456789012:role/myRole"
}

```

输出：

```

{
  "experimentTemplate": {
    "id": "ABCDE1fgHIJkLmNop",
    "description": "experimentTemplate",
    "targets": {
      "Instances-Target-1": {
        "resourceType": "aws:ec2:instance",
        "resourceArns": [
          "arn:aws:ec2:us-west-2:123456789012:instance/
i-12a3b4c56d78e9012"
        ],
        "selectionMode": "ALL"
      }
    },
    "actions": {
      "reboot": {
        "actionId": "aws:ec2:reboot-instances",
        "description": "reboot",
        "parameters": {},
        "targets": {
          "Instances": "Instances-Target-1"
        }
      }
    }
  },
}

```



```
    "stopConditions": [
      {
        "source": "aws:cloudwatch:alarm",
        "value": "arn:aws:cloudwatch:us-west-2:123456789012:alarm:alarmName"
      }
    ],
    "creationTime": 1616434850.659,
    "lastUpdateTime": 1616434850.659,
    "roleArn": "arn:aws:iam::123456789012:role/myRole",
    "tags": {}
  }
}
```

有关更多信息，请参阅《AWS 故障注入模拟器用户指南》中的[创建实验模板](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateExperimentTemplate](#) 中的。

## delete-experiment-template

以下代码示例演示了如何使用 delete-experiment-template。

### AWS CLI

#### 删除实验模板

以下 delete-experiment-template 示例删除了指定的实验模板。

```
aws fis delete-experiment-template \
  --id ABCDE1fgHIJkLmNop
```

输出：

```
{
  "experimentTemplate": {
    "id": "ABCDE1fgHIJkLmNop",
    "description": "myExperimentTemplate",
    "targets": {
      "Instances-Target-1": {
        "resourceType": "aws:ec2:instance",
        "resourceArns": [
          "arn:aws:ec2:us-west-2:123456789012:instance/
i-12a3b4c56d78e9012"
        ]
      }
    }
  }
}
```

```
        "selectionMode": "ALL"
      }
    },
    "actions": {
      "testaction": {
        "actionId": "aws:ec2:stop-instances",
        "parameters": {},
        "targets": {
          "Instances": "Instances-Target-1"
        }
      }
    },
    "stopConditions": [
      {
        "source": "none"
      }
    ],
    "creationTime": 1616017191.124,
    "lastUpdateTime": 1616017859.607,
    "roleArn": "arn:aws:iam::123456789012:role/FISRole"
  }
}
```

有关更多信息，请参阅《AWS 故障注入模拟器用户指南》中的[删除实验模板](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteExperimentTemplate](#) 中的。

## get-action

以下代码示例演示了如何使用 get-action。

### AWS CLI

#### 获取操作详情

以下 get-action 示例获取指定操作的详细信息。

```
aws fis get-action \  
  --id aws:ec2:stop-instances
```

输出：

```
{
```

```
"action": {
  "id": "aws:ec2:stop-instances",
  "description": "Stop the specified EC2 instances.",
  "parameters": {
    "startInstancesAfterDuration": {
      "description": "The time to wait before restarting the instances
(ISO 8601 duration).",
      "required": false
    }
  },
  "targets": {
    "Instances": {
      "resourceType": "aws:ec2:instance"
    }
  },
  "tags": {}
}
}
```

有关更多信息，请参阅《AWS 故障注入模拟器用户指南》中的[操作](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetAction](#)中的。

## get-experiment-template

以下代码示例演示了如何使用 get-experiment-template。

### AWS CLI

获取实验模板的详细信息

以下 get-experiment-template 示例获取指定实验模板的详细信息。

```
aws fis get-experiment-template \
  --id ABCDE1fgHIJkLmNop
```

输出：

```
{
  "experimentTemplate": {
    "id": "ABCDE1fgHIJkLmNop",
    "description": "myExperimentTemplate",
    "targets": {
```

```
    "Instances-Target-1": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": [
        "arn:aws:ec2:us-west-2:123456789012:instance/
i-12a3b4c56d78e9012"
      ],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "testaction": {
      "actionId": "aws:ec2:stop-instances",
      "parameters": {},
      "targets": {
        "Instances": "Instances-Target-1"
      }
    }
  },
  "stopConditions": [
    {
      "source": "none"
    }
  ],
  "creationTime": 1616017191.124,
  "lastUpdateTime": 1616017331.51,
  "roleArn": "arn:aws:iam::123456789012:role/FISRole",
  "tags": {
    "key": "value"
  }
}
}
```

有关更多信息，请参阅《AWS 故障注入模拟器用户指南》中的[实验模板](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetExperimentTemplate](#) 中的。

## get-experiment

以下代码示例演示了如何使用 get-experiment。

### AWS CLI

#### 获取实验详情

以下get-experiment示例获取指定实验的详细信息。

```
aws fis get-experiment \  
  --id ABC12DeFGhI3jKLMNOP
```

输出：

```
{  
  "experiment": {  
    "id": "ABC12DeFGhI3jKLMNOP",  
    "experimentTemplateId": "ABCDE1fgHIJkLmNop",  
    "roleArn": "arn:aws:iam::123456789012:role/myRole",  
    "state": {  
      "status": "completed",  
      "reason": "Experiment completed."  
    },  
    "targets": {  
      "Instances-Target-1": {  
        "resourceType": "aws:ec2:instance",  
        "resourceArns": [  
          "arn:aws:ec2:us-west-2:123456789012:instance/  
i-12a3b4c56d78e9012"  
        ],  
        "selectionMode": "ALL"  
      }  
    },  
    "actions": {  
      "reboot": {  
        "actionId": "aws:ec2:reboot-instances",  
        "parameters": {},  
        "targets": {  
          "Instances": "Instances-Target-1"  
        },  
        "state": {  
          "status": "completed",  
          "reason": "Action was completed."  
        }  
      }  
    },  
    "stopConditions": [  
      {  
        "source": "none"  
      }  
    ]  
  }  
}
```

```
    ],
    "creationTime": 1616432509.662,
    "startTime": 1616432509.962,
    "endTime": 1616432522.307,
    "tags": {}
  }
}
```

有关更多信息，请参阅《AWS 故障注入仿真器用户指南》中的 [AWS FIS 实验](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetExperiment](#) 中的。

## list-actions

以下代码示例演示了如何使用 `list-actions`。

### AWS CLI

#### 列出操作

以下 `list-actions` 示例列出了可用的操作。

```
aws fis list-actions
```

输出：

```
{
  "actions": [
    {
      "id": "aws:ec2:reboot-instances",
      "description": "Reboot the specified EC2 instances.",
      "targets": {
        "Instances": {
          "resourceType": "aws:ec2:instance"
        }
      },
      "tags": {}
    },
    {
      "id": "aws:ec2:stop-instances",
      "description": "Stop the specified EC2 instances.",
      "targets": {
        "Instances": {
```

```
        "resourceType": "aws:ec2:instance"
      }
    },
    "tags": {}
  },
  {
    "id": "aws:ec2:terminate-instances",
    "description": "Terminate the specified EC2 instances.",
    "targets": {
      "Instances": {
        "resourceType": "aws:ec2:instance"
      }
    },
    "tags": {}
  },
  {
    "id": "aws:ecs:drain-container-instances",
    "description": "Drain percentage of underlying EC2 instances on an ECS
cluster.",
    "targets": {
      "Clusters": {
        "resourceType": "aws:ecs:cluster"
      }
    },
    "tags": {}
  },
  {
    "id": "aws:eks:terminate-nodegroup-instances",
    "description": "Terminates a percentage of the underlying EC2 instances
in an EKS cluster.",
    "targets": {
      "Nodegroups": {
        "resourceType": "aws:eks:nodegroup"
      }
    },
    "tags": {}
  },
  {
    "id": "aws:fis:inject-api-internal-error",
    "description": "Cause an AWS service to return internal error responses
for specific callers and operations.",
    "targets": {
      "Roles": {
        "resourceType": "aws:iam:role"
      }
    }
  }
}
```

```
    }
  },
  "tags": {}
},
{
  "id": "aws:fis:inject-api-throttle-error",
  "description": "Cause an AWS service to return throttled responses for
specific callers and operations.",
  "targets": {
    "Roles": {
      "resourceType": "aws:iam:role"
    }
  },
  "tags": {}
},
{
  "id": "aws:fis:inject-api-unavailable-error",
  "description": "Cause an AWS service to return unavailable error
responses for specific callers and operations.",
  "targets": {
    "Roles": {
      "resourceType": "aws:iam:role"
    }
  },
  "tags": {}
},
{
  "id": "aws:fis:wait",
  "description": "Wait for the specified duration. Stop condition
monitoring will continue during this time.",
  "tags": {}
},
{
  "id": "aws:rds:failover-db-cluster",
  "description": "Failover a DB Cluster to one of the replicas.",
  "targets": {
    "Clusters": {
      "resourceType": "aws:rds:cluster"
    }
  },
  "tags": {}
},
{
  "id": "aws:rds:reboot-db-instances",
```



```

        "description": "Reboot the specified DB instances.",
        "targets": {
            "DBInstances": {
                "resourceType": "aws:rds:db"
            }
        },
        "tags": {}
    },
    {
        "id": "aws:ssm:send-command",
        "description": "Run the specified SSM document.",
        "targets": {
            "Instances": {
                "resourceType": "aws:ec2:instance"
            }
        },
        "tags": {}
    }
]
}

```

有关更多信息，请参阅《AWS 故障注入模拟器用户指南》中的[操作](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListActions](#) 中的。

## list-experiment-templates

以下代码示例演示了如何使用 list-experiment-templates。

### AWS CLI

#### 列出实验模板

以下 list-experiment-templates 示例列出了您 AWS 账户中的实验模板。

```
aws fis list-experiment-templates
```

输出：

```
{
  "experimentTemplates": [
    {
```

```
    "id": "ABCDE1fgHIJkLmNop",
    "description": "myExperimentTemplate",
    "creationTime": 1616017191.124,
    "lastUpdateTime": 1616017191.124,
    "tags": {
      "key": "value"
    }
  ]
}
```

有关更多信息，请参阅《AWS 故障注入模拟器用户指南》中的[实验模板](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListExperimentTemplates](#) 中的。

## list-experiments

以下代码示例演示了如何使用 list-experiments。

### AWS CLI

#### 列出实验

以下 list-experiments 示例列出了您 AWS 账户中的实验。

```
aws fis list-experiments
```

输出：

```
{
  "experiments": [
    {
      "id": "ABCdeF1GHijKLM23NO",
      "experimentTemplateId": "ABCDE1fgHIJkLmNop",
      "state": {
        "status": "running",
        "reason": "Experiment is running."
      },
      "creationTime": 1616017341.197,
      "tags": {
        "key": "value"
      }
    }
  ]
}
```

```
    }  
  ]  
}
```

有关更多信息，请参阅《AWS 故障注入模拟器用户指南》中的[实验](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListExperiments](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出资源的标签

以下`list-tags-for-resource`示例列出了指定资源的标签。

```
aws fis list-tags-for-resource \  
  --resource-arn arn:aws:fis:us-west-2:123456789012:experiment/ABC12DeFGhI3jKLMNOP
```

输出：

```
{  
  "tags": {  
    "key1": "value1",  
    "key2": "value2"  
  }  
}
```

有关更多信息，请参阅《AWS 故障注入模拟器用户指南》中的[标记您 AWS 的 FIS 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## start-experiment

以下代码示例演示了如何使用 `start-experiment`。

### AWS CLI

开始实验

以下 `start-experiment` 示例启动指定的实验。

```
aws fis start-experiment \  
  --experiment-template-id ABCDE1fgHIJkLmNop
```

输出：

```
{  
  "experiment": {  
    "id": "ABC12DeFGhI3jKLMNOP",  
    "experimentTemplateId": "ABCDE1fgHIJkLmNop",  
    "roleArn": "arn:aws:iam::123456789012:role/myRole",  
    "state": {  
      "status": "initiating",  
      "reason": "Experiment is initiating."  
    },  
    "targets": {  
      "Instances-Target-1": {  
        "resourceType": "aws:ec2:instance",  
        "resourceArns": [  
          "arn:aws:ec2:us-west-2:123456789012:instance/  
i-12a3b4c56d78e9012"  
        ],  
        "selectionMode": "ALL"  
      }  
    },  
    "actions": {  
      "reboot": {  
        "actionId": "aws:ec2:reboot-instances",  
        "parameters": {},  
        "targets": {  
          "Instances": "Instances-Target-1"  
        },  
        "state": {  
          "status": "pending",  
          "reason": "Initial state"  
        }  
      }  
    },  
    "stopConditions": [  
      {  
        "source": "none"  
      }  
    ]  
  }  
}
```

```
    ],
    "creationTime": 1616432464.025,
    "startTime": 1616432464.374,
    "tags": {}
  }
}
```

有关更多信息，请参阅《AWS 故障注入仿真器用户指南》中的 [AWS FIS 实验](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartExperiment](#) 中的。

## stop-experiment

以下代码示例演示了如何使用 stop-experiment。

### AWS CLI

#### 停止实验

以下 stop-experiment 示例停止运行指定的实验。

```
aws fis stop-experiment \
  --id ABC12DeFGhI3jKLMNOP
```

#### 输出：

```
{
  "experiment": {
    "id": "ABC12DeFGhI3jKLMNOP",
    "experimentTemplateId": "ABCDE1fgHIJkLmNop",
    "roleArn": "arn:aws:iam::123456789012:role/myRole",
    "state": {
      "status": "stopping",
      "reason": "Stopping Experiment."
    },
    "targets": {
      "Instances-Target-1": {
        "resourceType": "aws:ec2:instance",
        "resourceArns": [
          "arn:aws:ec2:us-west-2:123456789012:instance/i-12a3b4c56d78e9012"
        ]
      }
    }
  }
}
```

```
        "selectionMode": "ALL"
      }
    },
    "actions": {
      "reboot": {
        "actionId": "aws:ec2:reboot-instances",
        "parameters": {},
        "targets": {
          "Instances": "Instances-Target-1"
        },
        "startAfter": [
          "wait"
        ],
        "state": {
          "status": "pending",
          "reason": "Initial state."
        }
      },
      "wait": {
        "actionId": "aws:fis:wait",
        "parameters": {
          "duration": "PT5M"
        },
        "state": {
          "status": "running",
          "reason": ""
        }
      }
    },
    "stopConditions": [
      {
        "source": "none"
      }
    ],
    "creationTime": 1616432680.927,
    "startTime": 1616432681.177,
    "tags": {}
  }
}
```

有关更多信息，请参阅《AWS 故障注入仿真器用户指南》中的 [AWS FIS 实验](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StopExperiment](#) 中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源添加标签

以下tag-resource示例为指定的资源添加了标签。

```
aws fis tag-resource \  
  --resource-arn arn:aws:fis:us-west-2:123456789012:experiment/ABC12DeFGhI3jKLMNOP \  
  \  
  --tags key1=value1,key2=value2
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 故障注入模拟器用户指南》中的[标记您 AWS 的 FIS 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

取消资源标签

以下untag-resource示例从指定资源中删除标签。

```
aws fis untag-resource \  
  --resource-arn arn:aws:fis:us-west-2:123456789012:experiment/ABC12DeFGhI3jKLMNOP
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 故障注入模拟器用户指南》中的[标记您 AWS 的 FIS 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-experiment-template

以下代码示例演示了如何使用 update-experiment-template。

## AWS CLI

### 更新实验模板

以下update-experiment-template示例更新了指定实验模板的描述。

```
aws fis update-experiment-template \  
  --id ABCDE1fgHIJkLmNop \  
  ---description myExperimentTemplate
```

输出：

```
{  
  "experimentTemplate": {  
    "id": "ABCDE1fgHIJkLmNop",  
    "description": "myExperimentTemplate",  
    "targets": {  
      "Instances-Target-1": {  
        "resourceType": "aws:ec2:instance",  
        "resourceArns": [  
          "arn:aws:ec2:us-west-2:123456789012:instance/  
i-12a3b4c56d78e9012"  
        ],  
        "selectionMode": "ALL"  
      }  
    },  
    "actions": {  
      "testaction": {  
        "actionId": "aws:ec2:stop-instances",  
        "parameters": {},  
        "targets": {  
          "Instances": "Instances-Target-1"  
        }  
      }  
    },  
    "stopConditions": [  
      {  
        "source": "none"  
      }  
    ],  
    "creationTime": 1616017191.124,  
    "lastUpdateTime": 1616017859.607,  
    "roleArn": "arn:aws:iam::123456789012:role/FISRole",
```



```
    "tags": {
      "key": "value"
    }
  }
}
```

有关更多信息，请参阅《AWS 故障注入模拟器用户指南》中的[更新实验模板](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateExperimentTemplate](#) 中的。

## 使用亚马逊的 GameLift 示例 AWS CLI

以下代码示例向您展示如何在 Amazon 中使用来执行操作和实现常见场景 GameLift。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-build**

以下代码示例演示了如何使用 create-build。

AWS CLI

示例 1：使用 S3 存储桶中的文件创建游戏版本

以下 create-build 示例创建了自定义游戏编译资源。它使用存储在 S3 位置的由您控制的 AWS 账户中的压缩文件。此示例假设您已经创建了一个授予 Amazon 访问 S3 位置的 GameLift 权限的 IAM 角色。由于请求未指定操作系统，因此新的编译资源默认为 WINDOWS\_2012。

```
aws gamelift create-build \
```

```
--storage-location file://storage-loc.json \  
--name MegaFrogRaceServer.NA \  
--build-version 12345.678
```

storage-loc.json 的内容：

```
{  
  "Bucket": "MegaFrogRaceServer_NA_build_files"  
  "Key": "MegaFrogRaceServer_build_123.zip"  
  "RoleArn": "arn:aws:iam::123456789012:role/gamelift"  
}
```

输出：

```
{  
  "Build": {  
    "BuildArn": "arn:aws:gamelift:us-west-2::build/build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "BuildId": "build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "CreationTime": 1496708916.18,  
    "Name": "MegaFrogRaceServer.NA",  
    "OperatingSystem": "WINDOWS_2012",  
    "SizeOnDisk": 479303,  
    "Status": "INITIALIZED",  
    "Version": "12345.678"  
  },  
  "StorageLocation": {  
    "Bucket": "MegaFrogRaceServer_NA_build_files",  
    "Key": "MegaFrogRaceServer_build_123.zip"  
  }  
}
```

示例 2：创建游戏编译资源以便手动将文件上传到 GameLift

以下 create-build 示例创建了一个新的构建资源。它还会获得存储位置和临时证书，允许您手动将游戏版本上传到 Amazon S3 中的相应 GameLift 位置。成功上传版本后，该 GameLift 服务会验证版本并更新新版本的状态。

```
aws gamelift create-build \  
  --name MegaFrogRaceServer.NA \  
  --build-version 12345.678 \  
  --storage-location file://storage-loc.json \  
  --role-arn arn:aws:iam::123456789012:role/gamelift
```

```
--operating-system AMAZON_LINUX
```

输出：

```
{
  "Build": {
    "BuildArn": "arn:aws:gamelift:us-west-2::build/build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "BuildId": "build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "CreationTime": 1496708916.18,
    "Name": "MegaFrogRaceServer.NA",
    "OperatingSystem": "AMAZON_LINUX",
    "SizeOnDisk": 0,
    "Status": "INITIALIZED",
    "Version": "12345.678"
  },
  "StorageLocation": {
    "Bucket": "gamelift-builds-us-west-2",
    "Key": "123456789012/build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  },
  "UploadCredentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY",
    "SessionToken": "AgoGb3JpZ2luENz...EXAMPLETOKEN=="
  }
}
```

有关更多信息，请参阅《Amazon GameLift 开发者指南》GameLift中的[将自定义服务器版本上传到](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateBuild](#)中的。

## create-fleet

以下代码示例演示了如何使用 create-fleet。

### AWS CLI

#### 示例 1：创建基本的 Linux 舰队

以下create-fleet示例创建了一个配置最少的按需 Linux 实例队列来托管自定义服务器版本。您可以使用完成配置update-fleet。

```
aws gamelift create-fleet \
  --name MegaFrogRaceServer.NA.v2 \
  --description 'Hosts for v2 North America' \
  --build-id build-1111aaaa-22bb-33cc-44dd-5555eeee66ff \
  --certificate-configuration 'CertificateType=GENERATED' \
  --ec2-instance-type c4.large \
  --fleet-type ON_DEMAND \
  --runtime-configuration 'ServerProcesses=[{LaunchPath=/local/game/release-na/
MegaFrogRace_Server.exe,ConcurrentExecutions=1}]'
```

输出：

```
{
  "FleetAttributes": {
    "BuildId": "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",
    "CertificateConfiguration": {
      "CertificateType": "GENERATED"
    },
    "CreationTime": 1496365885.44,
    "Description": "Hosts for v2 North America",
    "FleetArn": "arn:aws:gamelift:us-west-2:444455556666:fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "FleetType": "ON_DEMAND",
    "InstanceType": "c4.large",
    "MetricGroups": ["default"],
    "Name": "MegaFrogRace.NA.v2",
    "NewGameSessionProtectionPolicy": "NoProtection",
    "OperatingSystem": "AMAZON_LINUX",
    "ServerLaunchPath": "/local/game/release-na/MegaFrogRace_Server.exe",
    "Status": "NEW"
  }
}
```

## 示例 2：创建基本的 Windows 队列

以下create-fleet示例创建了一个配置最少的 Spot Windows 实例队列来托管自定义服务器版本。您可以使用完成配置update-fleet。

```
aws gamelift create-fleet \
  --name MegaFrogRace.NA.v2 \
  --description 'Hosts for v2 North America' \
```

```
--build-id build-2222aaaa-33bb-44cc-55dd-6666eeee77ff \
--certificate-configuration 'CertificateType=GENERATED' \
--ec2-instance-type c4.large \
--fleet-type SPOT \
--runtime-configuration 'ServerProcesses=[{LaunchPath=C:\game
\Bin64.Release.Dedicated\MegaFrogRace_Server.exe,ConcurrentExecutions=1}]'
```

输出：

```
{
  "FleetAttributes": {
    "BuildId": "build-2222aaaa-33bb-44cc-55dd-6666eeee77ff",
    "CertificateConfiguration": {
      "CertificateType": "GENERATED"
    },
    "CreationTime": 1496365885.44,
    "Description": "Hosts for v2 North America",
    "FleetArn": "arn:aws:gamelift:us-west-2:444455556666:fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "FleetType": "SPOT",
    "InstanceType": "c4.large",
    "MetricGroups": ["default"],
    "Name": "MegaFrogRace.NA.v2",
    "NewGameSessionProtectionPolicy": "NoProtection",
    "OperatingSystem": "WINDOWS_2012",
    "ServerLaunchPath": "C:\game\Bin64.Release.Dedicated
\MegaFrogRace_Server.exe",
    "Status": "NEW"
  }
}
```

### 示例 3：创建完全配置的舰队

以下create-fleet示例为自定义服务器版本创建了一个 Spot Windows 实例队列，并提供了最常用的配置设置。

```
aws gamelift create-fleet \
  --name MegaFrogRace.NA.v2 \
  --description 'Hosts for v2 North America' \
  --build-id build-2222aaaa-33bb-44cc-55dd-6666eeee77ff \
  --certificate-configuration 'CertificateType=GENERATED' \
```

```

--ec2-instance-type c4.large \
--ec2-inbound-permissions
'FromPort=33435,ToPort=33435,IpRange=10.24.34.0/23,Protocol=UDP' \
--fleet-type SPOT \
--new-game-session-protection-policy FullProtection \
--runtime-configuration file://runtime-config.json \
--metric-groups default \
--instance-role-arn 'arn:aws:iam::444455556666:role/GameLiftS3Access'

```

runtime-config.json 的内容：

```

GameSessionActivationTimeoutSeconds=300,
MaxConcurrentGameSessionActivations=2,
ServerProcesses=[
  {LaunchPath=C:\game\Bin64.Release.Dedicated\MegaFrogRace_Server.exe,Parameters=-
debug,ConcurrentExecutions=1},
  {LaunchPath=C:\game\Bin64.Release.Dedicated
\MegaFrogRace_Server.exe,ConcurrentExecutions=1}]

```

输出：

```

{
  "FleetAttributes": {
    "InstanceRoleArn": "arn:aws:iam::444455556666:role/GameLiftS3Access",
    "Status": "NEW",
    "InstanceType": "c4.large",
    "FleetArn": "arn:aws:gamelift:us-west-2:444455556666:fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "Description": "Hosts for v2 North America",
    "FleetType": "SPOT",
    "OperatingSystem": "WINDOWS_2012",
    "Name": "MegaFrogRace.NA.v2",
    "CreationTime": 1569309011.11,
    "MetricGroups": [
      "default"
    ],
    "BuildId": "build-2222aaaa-33bb-44cc-55dd-6666eeee77ff",
    "ServerLaunchParameters": "abc",
    "ServerLaunchPath": "C:\\game\\Bin64.Release.Dedicated\\
MegaFrogRace_Server.exe",
    "NewGameSessionProtectionPolicy": "FullProtection",
    "CertificateConfiguration": {

```

```

        "CertificateType": "GENERATED"
    }
}
}

```

#### 示例 4：创建实时服务器队列

以下 `create-fleet` 示例使用已上传到 Amazon GameLift 的实时配置脚本创建竞价型实例队列。所有实时服务器都部署在 Linux 计算机上。在本示例中，假设上传的实时脚本包含多个脚本文件，该 `Init()` 函数位于该脚本文件中名为 `MainScript.js`。如图所示，此文件在运行时配置中被标识为启动脚本。

```

aws gamelift create-fleet \
  --name MegaFrogRace.NA.realtime \
  --description 'Mega Frog Race Realtime fleet' \
  --script-id script-1111aaaa-22bb-33cc-44dd-5555eeee66ff \
  --ec2-instance-type c4.large \
  --fleet-type SPOT \
  --certificate-configuration 'CertificateType=GENERATED' --runtime-configuration
  'ServerProcesses=[{LaunchPath=/local/game/MainScript.js,Parameters=+map
  Winter444,ConcurrentExecutions=5}]'

```

输出：

```

{
  "FleetAttributes": {
    "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "Status": "NEW",
    "CreationTime": 1569310745.212,
    "InstanceType": "c4.large",
    "NewGameSessionProtectionPolicy": "NoProtection",
    "CertificateConfiguration": {
      "CertificateType": "GENERATED"
    },
    "Name": "MegaFrogRace.NA.realtime",
    "ScriptId": "script-1111aaaa-22bb-33cc-44dd-5555eeee66ff",
    "FleetArn": "arn:aws:gamelift:us-west-2:444455556666:fleet/
    fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "FleetType": "SPOT",
    "MetricGroups": [
      "default"
    ],
  },
}

```

```
    "Description": "Mega Frog Race Realtime fleet",
    "OperatingSystem": "AMAZON_LINUX"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFleet](#)中的。

## create-game-session-queue

以下代码示例演示了如何使用 create-game-session-queue。

### AWS CLI

#### 示例 1：设置有序的游戏会话队列

以下create-game-session-queue示例创建了一个新的游戏会话队列，其目的地位于两个区域。它还会配置队列，使游戏会话请求在等待放置 10 分钟后超时。由于未定义延迟策略，因此会 GameLift 尝试将所有游戏会话放置在列出的第一个目标上。

```
aws gamelift create-game-session-queue \
  --name MegaFrogRaceServer-NA \
  --destinations file://destinations.json \
  --timeout-in-seconds 600
```

destinations.json 的内容：

```
{
  "Destinations": [
    {"DestinationArn": "arn:aws:gamelift:us-west-2::fleet/fleet-
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" },
    {"DestinationArn": "arn:aws:gamelift:us-west-1::fleet/fleet-
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222" }
  ]
}
```

输出：

```
{
  "GameSessionQueues": [
    {
```



```

        "Name": "MegaFrogRaceServer-NA",
        "GameSessionQueueArn": "arn:aws:gamelift:us-
west-2:123456789012:gamesessionqueue/MegaFrogRaceServer-NA",
        "TimeoutInSeconds": 600,
        "Destinations": [
            {"DestinationArn": "arn:aws:gamelift:us-west-2::fleet/fleet-
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"},
            {"DestinationArn": "arn:aws:gamelift:us-west-1::fleet/fleet-
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"}
        ]
    }
]
}

```

## 示例 2：使用玩家延迟策略设置游戏会话队列

以下 `create-game-session-queue` 示例使用两个玩家延迟策略创建了一个新的游戏会话队列。第一个策略设置了 100 毫秒的延迟上限，该上限在尝试放置游戏会话的第一分钟内强制执行。第二项策略将延迟上限提高到 200 毫秒，直到放置请求在 3 分钟后超时。

```

aws gamelift create-game-session-queue \
  --name MegaFrogRaceServer-NA \
  --destinations file://destinations.json \
  --player-latency-policies file://latency-policies.json \
  --timeout-in-seconds 180

```

`destinations.json` 的内容：

```

{
  "Destinations": [
    { "DestinationArn": "arn:aws:gamelift:us-west-2::fleet/fleet-
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" },
    { "DestinationArn": "arn:aws:gamelift:us-east-1::fleet/fleet-
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222" }
  ]
}

```

`latency-policies.json` 的内容：

```

{
  "PlayerLatencyPolicies": [

```

```

        {"MaximumIndividualPlayerLatencyMilliseconds": 200},
        {"MaximumIndividualPlayerLatencyMilliseconds": 100, "PolicyDurationSeconds":
60}
    ]
}

```

输出：

```

{
  "GameSessionQueue": {
    "Name": "MegaFrogRaceServer-NA",
    "GameSessionQueueArn": "arn:aws:gamelift:us-
west-2:111122223333:gamesessionqueue/MegaFrogRaceServer-NA",
    "TimeoutInSeconds": 600,
    "PlayerLatencyPolicies": [
      {
        "MaximumIndividualPlayerLatencyMilliseconds": 100,
        "PolicyDurationSeconds": 60
      },
      {
        "MaximumIndividualPlayerLatencyMilliseconds": 200
      }
    ]
    "Destinations": [
      {"DestinationArn": "arn:aws:gamelift:us-west-2::fleet/fleet-
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"},
      {"DestinationArn": "arn:aws:gamelift:us-east-1::fleet/fleet-
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"}
    ],
  }
}

```

有关更多信息，请参阅《Amazon GameLift 开发者指南》中的[创建队列](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateGameSessionQueue](#) 中的。

## delete-build

以下代码示例演示了如何使用 delete-build。

### AWS CLI

#### 删除自定义游戏版本

以下delete-build示例从您的 Amazon GameLift 账户中删除了一个版本。版本被删除后，您将无法使用它来创建新的舰队。此操作无法撤消。

```
aws gamelift delete-build \  
  --build-id build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBuild](#)中的。

## delete-fleet

以下代码示例演示了如何使用 delete-fleet。

### AWS CLI

删除不再使用的舰队

以下delete-fleet示例移除了已缩减为零实例的队列。如果队列容量大于零，则请求失败并出现 HTTP 400 错误。

```
aws gamelift delete-fleet \  
  --fleet-id fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon GameLift 开发者指南》中的[管理 GameLift 舰队](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFleet](#)中的。

## delete-game-session-queue

以下代码示例演示了如何使用 delete-game-session-queue。

### AWS CLI

删除游戏会话队列

以下delete-game-session-queue示例删除了指定的游戏会话队列。

```
aws gamelift delete-game-session-queue \  
  --name MegaFrogRace-NA
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteGameSessionQueue](#)中的。

## describe-build

以下代码示例演示了如何使用 describe-build。

### AWS CLI

获取有关自定义游戏版本的信息

以下describe-build示例检索游戏服务器版本资源的属性。

```
aws gamelift describe-build \  
  --build-id build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "Build": {  
    "BuildArn": "arn:aws:gamelift:us-west-2::build/build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "BuildId": "build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "CreationTime": 1496708916.18,  
    "Name": "My_Game_Server_Build_One",  
    "OperatingSystem": "AMAZON_LINUX",  
    "SizeOnDisk": 1304924,  
    "Status": "READY",  
    "Version": "12345.678"  
  }  
}
```

有关更多信息，请参阅《Amazon GameLift 开发者指南》 GameLift中的[将自定义服务器版本上传到](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeBuild](#)中的。

## describe-ec2-instance-limits

以下代码示例演示了如何使用 describe-ec2-instance-limits。

## AWS CLI

### 检索 EC2 实例类型的服务限制

以下describe-ec2-instance-limits示例显示了当前区域中指定 EC2 实例类型允许的最大实例数和当前使用的实例数。结果表明，在允许的二十个实例中，只使用了五个。

```
aws gamelift describe-ec2-instance-limits \  
  --ec2-instance-type m5.large
```

输出：

```
{  
  "EC2InstanceLimits": [  
    {  
      "EC2InstanceType": "m5.large",  
      "CurrentInstances": 5,  
      "InstanceLimit": 20  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon GameLift 开发者指南中的[选择计算资源](#)。

- 有关 API 的详细信息，请参阅InstanceLimits 《AWS CLI 命令参考》中的 [DescribeEc2](#)。

## describe-fleet-attributes

以下代码示例演示了如何使用 describe-fleet-attributes。

## AWS CLI

### 示例 1：查看舰队列表的属性

以下describe-fleet-attributes示例检索两个指定舰队的舰队属性。如图所示，所请求的队列采用相同的版本部署，一个用于按需实例，一个用于竞价型实例，但有一些细微的配置差异。

```
aws gamelift describe-fleet-attributes \  
  --fleet-ids arn:aws:gamelift:us-west-2::fleet/fleet-a1b2c3d4-5678-90ab-cdef-  
  EXAMPLE111111 fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222
```

输出：

```
{
  "FleetAttributes": [
    {
      "FleetId": "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "FleetArn": "arn:aws:gamelift:us-west-2::fleet/fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "FleetType": "ON_DEMAND",
      "InstanceType": "c4.large",
      "Description": "On-demand hosts for v2 North America",
      "Name": "MegaFrogRaceServer.NA.v2-od",
      "CreationTime": 1568836191.995,
      "Status": "ACTIVE",
      "BuildId": "build-a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
      "BuildArn": "arn:aws:gamelift:us-west-2::build/build-a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
      "ServerLaunchPath": "C:\\\\game\\\\MegaFrogRace_Server.exe",
      "ServerLaunchParameters": "+gamelift_start_server",
      "NewGameSessionProtectionPolicy": "NoProtection",
      "OperatingSystem": "WINDOWS_2012",
      "MetricGroups": [
        "default"
      ],
      "CertificateConfiguration": {
        "CertificateType": "DISABLED"
      }
    },
    {
      "FleetId": "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "FleetArn": "arn:aws:gamelift:us-west-2::fleet/fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "FleetType": "SPOT",
      "InstanceType": "c4.large",
      "Description": "On-demand hosts for v2 North America",
      "Name": "MegaFrogRaceServer.NA.v2-spot",
      "CreationTime": 1568838275.379,
      "Status": "ACTIVATING",
      "BuildId": "build-a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
      "BuildArn": "arn:aws:gamelift:us-west-2::build/build-a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
      "ServerLaunchPath": "C:\\\\game\\\\MegaFrogRace_Server.exe",
      "NewGameSessionProtectionPolicy": "NoProtection",
      "OperatingSystem": "WINDOWS_2012",
```

```

        "MetricGroups": [
            "default"
        ],
        "CertificateConfiguration": {
            "CertificateType": "GENERATED"
        }
    }
]
}

```

## 示例 2：请求所有舰队的属性

以下内容 `describe-fleet-attributes` 返回所有处于任何状态的舰队的舰队属性。此示例说明如何使用分页参数一次返回一个舰队。

```
aws gamelift describe-fleet-attributes \
  --limit 1
```

## 输出：

```

{
  "FleetAttributes": [
    {
      "FleetId": "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "FleetArn": "arn:aws:gamelift:us-west-2::fleet/fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "FleetType": "SPOT",
      "InstanceType": "c4.large",
      "Description": "On-demand hosts for v2 North America",
      "Name": "MegaFrogRaceServer.NA.v2-spot",
      "CreationTime": 1568838275.379,
      "Status": "ACTIVATING",
      "BuildId": "build-a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
      "BuildArn": "arn:aws:gamelift:us-west-2::build/build-a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
      "ServerLaunchPath": "C:\\game\\MegaFrogRace_Server.exe",
      "NewGameSessionProtectionPolicy": "NoProtection",
      "OperatingSystem": "WINDOWS_2012",
      "MetricGroups": [
        "default"
      ],
      "CertificateConfiguration": {

```

```

        "CertificateType": "GENERATED"
      }
    ]
  },
  "NextToken":
  "eyJhd3NBWY2NvdW50SWQiOnsicyI6IjMwMjc3NjAxNjM5OCJ9LCJidWlsZE1kIjp7InMiOiJidWlsZC01NWYxZTZmMS1"
}

```

输出包含一个NextToken值，您可以在第二次调用该命令时使用该值。将该值传递给--next-token参数以指定从何处获取输出。以下命令返回输出中的第二个结果。

```

aws gamelift describe-fleet-attributes \
  --limit 1 \
  --next-token
eyJhd3NBWY2NvdW50SWQiOnsicyI6IjMwMjc3NjAxNjM5OCJ9LCJidWlsZE1kIjp7InMiOiJidWlsZC01NWYxZTZmMS1

```

重复此操作，直到响应中不包含NextToken值。

有关更多信息，请参阅《Amazon GameLift 开发者指南》中的[设置 GameLift 队列](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFleetAttributes](#)中的。

## describe-fleet-capacity

以下代码示例演示了如何使用 describe-fleet-capacity。

### AWS CLI

查看舰队列表的容量状态

以下describe-fleet-capacity示例检索两个指定队列的当前容量。

```

aws gamelift describe-fleet-capacity \
  --fleet-ids arn:aws:gamelift:us-west-2::fleet/fleet-a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111 fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222

```

输出：

```

{
  "FleetCapacity": [
    {

```



```
    "FleetId": "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "InstanceType": "c5.large",
    "InstanceCounts": {
      "DESIRED": 10,
      "MINIMUM": 1,
      "MAXIMUM": 20,
      "PENDING": 0,
      "ACTIVE": 10,
      "IDLE": 3,
      "TERMINATING": 0
    }
  },
  {
    "FleetId": "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "InstanceType": "c5.large",
    "InstanceCounts": {
      "DESIRED": 13,
      "MINIMUM": 1,
      "MAXIMUM": 20,
      "PENDING": 0,
      "ACTIVE": 15,
      "IDLE": 2,
      "TERMINATING": 2
    }
  }
]
```

有关更多信息，请参阅《Amazon GameLift 开发者指南》中的[舰队 GameLift 指标](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeFleetCapacity](#) 中的。

## describe-fleet-events

以下代码示例演示了如何使用 describe-fleet-events。

### AWS CLI

请求指定时间段内的事件

以下 describe-fleet-events 示例显示了在指定时间段内发生的所有舰队相关事件的详细信息。

```
aws gamelift describe-fleet-events \  
  --fleet-id arn:aws:gamelift:us-west-2::fleet/fleet-a1b2c3d4-5678-90ab-cdef-  
EXAMPLE11111 \  
  --start-time 1579647600 \  
  --end-time 1579649400 \  
  --limit 5
```

输出：

```
{  
  "Events": [  
    {  
      "EventId": "a37b6892-5d07-4d3b-8b47-80244ecf66b9",  
      "ResourceId": "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "EventCode": "FLEET_STATE_ACTIVE",  
      "Message": "Fleet fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 changed  
state to ACTIVE",  
      "EventTime": 1579649342.191  
    },  
    {  
      "EventId": "67da4ec9-92a3-4d95-886a-5d6772c24063",  
      "ResourceId": "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "EventCode": "FLEET_STATE_ACTIVATING",  
      "Message": "Fleet fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 changed  
state to ACTIVATING",  
      "EventTime": 1579649321.427  
    },  
    {  
      "EventId": "23813a46-a9e6-4a53-8847-f12e6a8381ac",  
      "ResourceId": "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "EventCode": "FLEET_STATE_BUILDING",  
      "Message": "Fleet fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 changed  
state to BUILDING",  
      "EventTime": 1579649321.243  
    },  
    {  
      "EventId": "3bf217d0-1d44-42f9-9202-433ed475d2e8",  
      "ResourceId": "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "EventCode": "FLEET_STATE_VALIDATING",  
      "Message": "Fleet fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 changed  
state to VALIDATING",  
      "EventTime": 1579649197.449  
    },  
  ]  
}
```

```

    {
      "EventId": "2ecd0130-5986-44eb-99a7-62df27741084",
      "ResourceId": "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "EventCode": "FLEET_VALIDATION_LAUNCH_PATH_NOT_FOUND",
      "Message": "Failed to find a valid path",
      "EventTime": 1569319075.839,
      "PreSignedLogUrl": "https://gamelift-event-logs-prod-
us-west-2.s3.us-west-2.amazonaws.com/logs/fleet-83422059-8329-42a2-
a4d6-c4444386a6f8/events/2ecd0130-5986-44eb-99a7-62df27741084/
FLEET_VALIDATION_LAUNCH_PATH_NOT_FOUND.txt?X-Amz-Security-
Token=IQoJb3JpZ2luX2VjEB8aCXVzLXdlc3QtMiJHMEUCIHV5K%2FLPx8h310D
%2FAvx0%2FZxsDy5XA3cJ0wPdu3T0eBa%2FAiEA1yovokcZYy%2FV4CWW6l26aFyiSH0
%2Bxz%2FBMAhEHYHMqncqkQMImp%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F
%2FARAAGgw3NDEwNjE0TixNzEiDI8rsZtzLzlwEDQhXSrlAtl5Ae
%2Fgo6FCIzqXPbXfB0nSvFYqeDlriZarEpKqKrUt8mXQv9iqHResqCph9AKo49lwgSYTT2QoSxnrD7%2FUgv
%2BZm2pVuczvUkUA0fxc6s0GxpjIAzdIE%2F5P%2FB7B9M%2BVZ
%2F9KF82hbJi0HTE6Y7BjKsEgFCvk4UXILhfjtan9iQl8%2F21ZTurAcJbm7Y5tuLF9SWSK3%2BEa7VX0cCK4D401sMj
%2FIaXoHkNvg0RVTa0hIqdvpDQlsSBNdqTXbjHTu6fETE9Y9Ky%2BiJK5KiUG
%2F59GjCpDcvS1FqKeLUEmKT7wysGmvjMc2n%2Fr
%2F9VxQfte7w9srXwLLAQuwhiXAAyI5ICMz5JvzjzQwTqD4CHTVKUUDwL
%2BRZzbuuqkJ0bZml02CkRGp%2B74RTAzLbWptVqZTIIfzctiCTmWxb
%2FmKyELRYsVLrwNJ%2BGJ7%2BCrN0RC%2FjlgfLYIZyeAqjPgAu5HjgX
%2BM7jCo9M7wBTInAXK0FQuf9dvA84SuwX0JFp17LYGjrHMKv0qC3GfbTMrZ6kzeNV9awKCpXB2Gnx9z2KvI1JddqirWV
%2F9C6%2B4jIZPME3jXmZcEHqqw5uvAVF7aeIavtUZU8pxpDIWT0YE4p3Kriy2AA7ziCRKtVfjv839InyLk8LUjsioWk
%2BYUq8%2FDTL1Lxqj1S%2Fi04TI0Wo7i1Ao%2FKKWWF4guuNDexj8E00ynSp1yImB
%2BZf2Fua3044W4eEXAMPLE33333&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20170621T231808Z&X-Amz-SignedHeaders=host&X-Amz-Expires=900&X-Amz-
Credential=AKIAIOSFODNN7EXAMPLE%2F20170621%2Fus-west-2%2Fs3%2Faws4_request&X-Amz-
Signature=wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY"
    }
  ],
  "NextToken":
  "eyJhd3NBZjY2NvdW50SWQ1O3sic3I6IjMwMjc3NjAxNjM5OCJ9LCJidWlsZElkIjpw7InMi0iJidWlsZC01NWYxZTZmMS"
}

```

有关更多信息，请参阅 Amazon GameLift 开发者指南中的 [调试 GameLift 队列问题](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeFleetEvents](#) 中的。

## describe-fleet-port-settings

以下代码示例演示了如何使用 `describe-fleet-port-settings`。

## AWS CLI

### 查看队列的入站连接权限

以下describe-fleet-port-settings示例检索指定队列的连接设置。

```
aws gamelift describe-fleet-port-settings \  
  --fleet-id arn:aws:gamelift:us-west-2::fleet/fleet-a1b2c3d4-5678-90ab-cdef-  
  EXAMPLE11111
```

输出：

```
{  
  "InboundPermissions": [  
    {  
      "FromPort": 33400,  
      "ToPort": 33500,  
      "IpRange": "0.0.0.0/0",  
      "Protocol": "UDP"  
    },  
    {  
      "FromPort": 1900,  
      "ToPort": 2000,  
      "IpRange": "0.0.0.0/0",  
      "Protocol": "TCP"  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon GameLift 开发者指南》中的[设置 GameLift 队列](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFleetPortSettings](#)中的。

## describe-fleet-utilization

以下代码示例演示了如何使用 describe-fleet-utilization。

## AWS CLI

### 示例 1：查看车队列表的使用数据

以下describe-fleet-utilization示例检索一个指定队列的当前使用情况信息。

```
aws gamelift describe-fleet-utilization \  
  --fleet-ids arn:aws:gamelift:us-west-2::fleet/fleet-a1b2c3d4-5678-90ab-cdef-  
  EXAMPLE111111
```

输出：

```
{  
  "FleetUtilization": [  
    {  
      "FleetId": "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE111111",  
      "ActiveServerProcessCount": 100,  
      "ActiveGameSessionCount": 62,  
      "CurrentPlayerSessionCount": 329,  
      "MaximumPlayerSessionCount": 1000  
    }  
  ]  
}
```

示例 2：请求所有车队的使用数据

以下内容 `describe-fleet-utilization` 返回所有处于任何状态的舰队的车队使用数据。此示例使用分页参数一次返回两个队列的数据。

```
aws gamelift describe-fleet-utilization \  
  --limit 2
```

输出：

```
{  
  "FleetUtilization": [  
    {  
      "FleetId": "fleet-1111aaaa-22bb-33cc-44dd-5555eeee66ff",  
      "ActiveServerProcessCount": 100,  
      "ActiveGameSessionCount": 13,  
      "CurrentPlayerSessionCount": 98,  
      "MaximumPlayerSessionCount": 1000  
    },  
    {  
      "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
      "ActiveServerProcessCount": 100,  
      "ActiveGameSessionCount": 62,  
      "CurrentPlayerSessionCount": 98,  
      "MaximumPlayerSessionCount": 1000  
    }  
  ]  
}
```

```

        "CurrentPlayerSessionCount": 329,
        "MaximumPlayerSessionCount": 1000
    }
],
"NextToken":
"eyJhd3NBW52NvdW50SWQiOmsicyI6IjMwMjc3NjAxNjM5OCJ9LCJidWlsZElkIjpw7InMiOiJidWlsZC01NWYxZTZmMS1"
}

```

再次调用该命令，将该NextToken值作为参数传递给--next-token参数以查看接下来的两个结果。

```

aws gamelift describe-fleet-utilization \
  --limit 2 \
  --next-token
eyJhd3NBW52NvdW50SWQiOmsicyI6IjMwMjc3NjAxNjM5OCJ9LCJidWlsZElkIjpw7InMiOiJidWlsZC01NWYxZTZmMS1

```

重复此操作，直到响应不再在输出中包含NextToken值。

有关更多信息，请参阅《Amazon GameLift 开发者指南》中的[舰队GameLift 指标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFleetUtilization](#)中的。

## describe-game-session-queues

以下代码示例演示了如何使用 describe-game-session-queues。

### AWS CLI

#### 查看游戏会话队列

以下describe-game-session-queues示例检索两个指定队列的属性。

```

aws gamelift describe-game-session-queues \
  --names MegaFrogRace-NA MegaFrogRace-EU

```

输出：

```

{
  "GameSessionQueues": [{
    "Destinations": [{

```

```

        "DestinationArn": "arn:aws:gamelift:us-west-2::fleet/fleet-
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    },
    {
        "DestinationArn": "arn:aws:gamelift:us-west-2::fleet/fleet-
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
    }
],
"Name": "MegaFrogRace-NA",
"TimeoutInSeconds": 600,
"GameSessionQueueArn": "arn:aws:gamelift:us-west-2::gamesessionqueue/
MegaFrogRace-NA",
"PlayerLatencyPolicies": [{
    "MaximumIndividualPlayerLatencyMilliseconds": 200
},
{
    "MaximumIndividualPlayerLatencyMilliseconds": 100,
    "PolicyDurationSeconds": 60
}
],
"FilterConfiguration": {
    "AllowedLocations": ["us-west-2", "ap-south-1", "us-east-1"]
},
"PriorityConfiguration": {
    "PriorityOrder": ["LOCATION", "FLEET_TYPE", "DESTINATION"],
    "LocationOrder": ["us-west-2", "ap-south-1", "us-east-1"]
}
},
{
    "Destinations": [{
        "DestinationArn": "arn:aws:gamelift:eu-west-3::fleet/fleet-
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
    }],
    "Name": "MegaFrogRace-EU",
    "TimeoutInSeconds": 600,
    "GameSessionQueueArn": "arn:aws:gamelift:us-west-2::gamesessionqueue/
MegaFrogRace-EU"
}
]
}

```

有关更多信息，请参阅 Amazon GameLift 开发者指南中的[使用多区域队列](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeGameSessionQueues](#)中的。

## describe-runtime-configuration

以下代码示例演示了如何使用 describe-runtime-configuration。

### AWS CLI

#### 请求队列的运行时配置

以下 describe-runtime-configuration 示例检索有关指定队列的当前运行时配置的详细信息。

```
aws gamelift describe-runtime-configuration \  
  --fleet-id fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

#### 输出：

```
{  
  "RuntimeConfiguration": {  
    "ServerProcesses": [  
      {  
        "LaunchPath": "C:\\game\\Bin64.Release.Dedicated  
\\MegaFrogRace_Server.exe",  
        "Parameters": "+gamelift_start_server",  
        "ConcurrentExecutions": 3  
      },  
      {  
        "LaunchPath": "C:\\game\\Bin64.Release.Dedicated  
\\MegaFrogRace_Server.exe",  
        "Parameters": "+gamelift_start_server +debug",  
        "ConcurrentExecutions": 1  
      }  
    ],  
    "MaxConcurrentGameSessionActivations": 2147483647,  
    "GameSessionActivationTimeoutSeconds": 300  
  }  
}
```

有关更多信息，请参阅 Amazon GameLift 开发者指南中的在[队列上运行多个进程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeRuntimeConfiguration](#) 中的。



## list-builds

以下代码示例演示了如何使用 `list-builds`。

### AWS CLI

#### 示例 1：获取自定义游戏版本列表

以下 `list-builds` 示例检索当前区域中所有游戏服务器版本的属性。示例请求说明了如何使用分页参数 `Limit` 和 `NextToken` 按顺序检索结果。第一个命令检索前两个版本。由于有两个以上的可用结果，因此响应中包含 `aNextToken`，表示有更多结果可用。

```
aws gamelift list-builds \  
  --limit 2
```

输出：

```
{  
  "Builds": [  
    {  
      "BuildArn": "arn:aws:gamelift:us-west-2::build/build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "BuildId": "build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "CreationTime": 1495664528.723,  
      "Name": "My_Game_Server_Build_One",  
      "OperatingSystem": "WINDOWS_2012",  
      "SizeOnDisk": 8567781,  
      "Status": "READY",  
      "Version": "12345.678"  
    },  
    {  
      "BuildArn": "arn:aws:gamelift:us-west-2::build/build-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
      "BuildId": "build-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
      "CreationTime": 1495528748.555,  
      "Name": "My_Game_Server_Build_Two",  
      "OperatingSystem": "AMAZON_LINUX_2",  
      "SizeOnDisk": 8567781,  
      "Status": "FAILED",  
      "Version": "23456.789"  
    }  
  ],  
}
```

```
    "NextToken":  
    "eyJhd3NBWY2NvdW50SWQiOnsicyI6IjMwMjc3NjAxNjM5OCJ9LCJidWlsZE1kIjp7InMiOiJidWlsZC01NWYxZTZmMS1  
  }
```

然后，您可以使用 `--next-token` 参数再次调用该命令，如下所示，以查看接下来的两个版本。

```
aws gamelift list-builds \  
  --limit 2  
  --next-token  
  eyJhd3NBWY2NvdW50SWQiOnsicyI6IjMwMjc3NjAxNjM5OCJ9LCJidWlsZE1kIjp7InMiOiJidWlsZC01NWYxZTZmMS1
```

重复此操作，直到响应中不包含 `NextToken` 值。

### 示例 2：获取处于失败状态的自定义游戏版本列表

以下 `list-builds` 示例检索当前区域中所有当前状态为 `FAILED` 的游戏服务器版本的属性。

```
aws gamelift list-builds \  
  --status FAILED
```

输出：

```
{  
  "Builds": [  
    {  
      "BuildArn": "arn:aws:gamelift:us-west-2::build/build-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
      "BuildId": "build-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
      "CreationTime": 1495528748.555,  
      "Name": "My_Game_Server_Build_Two",  
      "OperatingSystem": "AMAZON_LINUX_2",  
      "SizeOnDisk": 8567781,  
      "Status": "FAILED",  
      "Version": "23456.789"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListBuilds](#) 中的。

## list-fleets

以下代码示例演示了如何使用 `list-fleets`。

### AWS CLI

示例 1：获取一个地区内所有舰队的列表

以下 `list-fleets` 示例显示了当前区域中所有舰队的舰队 ID。此示例使用分页参数一次检索两个舰队 ID。响应中包含一个 `next-token` 属性，该属性表示还有更多结果需要检索。

```
aws gamelift list-fleets \  
  --limit 2
```

输出：

```
{  
  "FleetIds": [  
    "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"  
  ],  
  "NextToken":  
    "eyJhd3NBZ2NvdW50SWQiOnsicyI6IjMwMjc3NjAxNjM5OCJ9LCJidWlsZEIkIjp7InMiOiJidWlsZC01NWYxZTZmMS"  
}
```

您可以在下一个命令中传递上一个响应中的 `NextToken` 值，如下所示，以获得接下来的两个结果。

```
aws gamelift list-fleets \  
  --limit 2 \  
  --next-token  
eyJhd3NBZ2NvdW50SWQiOnsicyI6IjMwMjc3NjAxNjM5OCJ9LCJidWlsZEIkIjp7InMiOiJidWlsZC00NDRLZjQxZS1
```

示例 2：获取具有特定版本或脚本的区域内所有舰队的列表

以下 `list-builds` 示例检索使用指定游戏版本部署的队列的 ID。如果您使用的是实时服务器，则可以提供脚本 ID 来代替构建 ID。由于此示例未指定 `limit` 参数，因此结果最多可以包含 16 个舰队 ID。

```
aws gamelift list-fleets \  
  --build-id build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
  "FleetIds": [
    "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE44444"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFleets](#)中的。

## request-upload-credentials

以下代码示例演示了如何使用 request-upload-credentials。

### AWS CLI

刷新上传版本的访问凭证

以下create-build示例获取新的有效访问凭证，用于将 GameLift 构建文件上传到 Amazon S3 位置。证书的有效期限有限。您可以从对原始CreateBuild请求的响应中获得构建 ID。

```
aws gamelift request-upload-credentials \
  --build-id build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
  "StorageLocation": {
    "Bucket": "gamelift-builds-us-west-2",
    "Key": "123456789012/build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  },
  "UploadCredentials": {
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY",
    "SessionToken": "AgoGb3JpZ2luENZ...EXAMPLETOKEN=="
  }
}
```

有关更多信息，请参阅《Amazon GameLift 开发者指南》 GameLift中的[将自定义服务器版本上传到](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RequestUploadCredentials](#)中的。

## start-fleet-actions

以下代码示例演示了如何使用 `start-fleet-actions`。

### AWS CLI

要重新启动队列自动扩展活动

以下 `start-fleet-actions` 示例恢复使用为指定队列定义但因调 `stop-fleet-actions` 用 ``而停止的所有扩展策略。启动后，扩展策略会立即开始跟踪各自的指标。

```
aws gamelift start-fleet-actions \  
  --fleet-id fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --actions AUTO_SCALING
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartFleetActions](#)中的。

## stop-fleet-actions

以下代码示例演示了如何使用 `stop-fleet-actions`。

### AWS CLI

停止队列的自动扩展活动

以下 `stop-fleet-actions` 示例停止使用为指定队列定义的所有扩展策略。策略暂停后，除非您手动调整，否则队列容量将保持不变。

```
aws gamelift start-fleet-actions \  
  --fleet-id fleet-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --actions AUTO_SCALING
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopFleetActions](#)中的。

## update-build

以下代码示例演示了如何使用 `update-build`。

## AWS CLI

### 更新自定义游戏版本

以下update-build示例更改了与指定编译资源关联的名称和版本信息。返回的构建对象验证更改是否已成功完成。

```
aws gamelift update-build \  
  --build-id build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --name MegaFrogRaceServer.NA.east \  
  --build-version 12345.east
```

输出：

```
{  
  "Build": {  
    "BuildArn": "arn:aws:gamelift:us-west-2::build/build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "BuildId": "build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "CreationTime": 1496708916.18,  
    "Name": "MegaFrogRaceServer.NA.east",  
    "OperatingSystem": "AMAZON_LINUX_2",  
    "SizeOnDisk": 1304924,  
    "Status": "READY",  
    "Version": "12345.east"  
  }  
}
```

有关更多信息，请参阅《Amazon GameLift 开发者指南》中的“[更新您的构建文件](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateBuild](#)中的。

## update-game-session-queue

以下代码示例演示了如何使用 update-game-session-queue。

## AWS CLI

### 更新游戏会话队列配置

以下update-game-session-queue示例添加了一个新的目标并更新了现有游戏会话队列的玩家延迟策略。

```
aws gamelift update-game-session-queue \  
  --name MegaFrogRace-NA \  
  --destinations file://destinations.json \  
  --player-latency-policies file://latency-policies.json
```

destinations.json 的内容 :

```
{  
  "Destinations": [  
    {"DestinationArn": "arn:aws:gamelift:us-west-2::fleet/  
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d"},  
    {"DestinationArn": "arn:aws:gamelift:us-east-1::fleet/  
fleet-5c6d3c4d-5e6f-7a8b-9c0d-1e2f3a4b5a2b"},  
    {"DestinationArn": "arn:aws:gamelift:us-east-1::alias/  
alias-11aa22bb-3c4d-5e6f-000a-1111aaaa22bb"}  
  ]  
}
```

latency-policies.json 的内容 :

```
{  
  "PlayerLatencyPolicies": [  
    {"MaximumIndividualPlayerLatencyMilliseconds": 200},  
    {"MaximumIndividualPlayerLatencyMilliseconds": 150, "PolicyDurationSeconds":  
120},  
    {"MaximumIndividualPlayerLatencyMilliseconds": 100, "PolicyDurationSeconds":  
120}  
  ]  
}
```

输出 :

```
{  
  "GameSessionQueue": {  
    "Destinations": [  
      {"DestinationArn": "arn:aws:gamelift:us-west-2::fleet/  
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d"},  
      {"DestinationArn": "arn:aws:gamelift:us-east-1::fleet/  
fleet-5c6d3c4d-5e6f-7a8b-9c0d-1e2f3a4b5a2b"},  
      {"DestinationArn": "arn:aws:gamelift:us-east-1::alias/  
alias-11aa22bb-3c4d-5e6f-000a-1111aaaa22bb"}  
    ]  
  }  
}
```

```

    ],
    "GameSessionQueueArn": "arn:aws:gamelift:us-
west-2:111122223333:gamesessionqueue/MegaFrogRace-NA",
    "Name": "MegaFrogRace-NA",
    "TimeoutInSeconds": 600,
    "PlayerLatencyPolicies": [
      {"MaximumIndividualPlayerLatencyMilliseconds": 200},
      {"MaximumIndividualPlayerLatencyMilliseconds": 150,
"PolicyDurationSeconds": 120},
      {"MaximumIndividualPlayerLatencyMilliseconds": 100,
"PolicyDurationSeconds": 120}
    ]
  }
}

```

有关更多信息，请参阅 Amazon GameLift 开发者指南中的[使用多区域队列](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateGameSessionQueue](#)中的。

## upload-build

以下代码示例演示了如何使用 upload-build。

### AWS CLI

示例 1：上传 Linux 游戏服务器版本

以下 upload-build 示例将 Linux 游戏服务器编译文件从文件目录上传到 GameLift 服务并创建构建资源。

```

aws gamelift upload-build \
  --name MegaFrogRaceServer.NA \
  --build-version 2.0.1 \
  --build-root ~/MegaFrogRace_Server/release-na \
  --operating-system AMAZON_LINUX_2
  --server-sdk-version 4.0.2

```

输出：

```

Uploading ~/MegaFrogRace_Server/release-na: 16.0 KiB / 74.6 KiB (21.45%)
Uploading ~/MegaFrogRace_Server/release-na: 32.0 KiB / 74.6 KiB (42.89%)
Uploading ~/MegaFrogRace_Server/release-na: 48.0 KiB / 74.6 KiB (64.34%)

```



```
Uploading ~/MegaFrogRace_Server/release-na: 64.0 KiB / 74.6 KiB (85.79%)
Uploading ~/MegaFrogRace_Server/release-na: 74.6 KiB / 74.6 KiB (100.00%)
Successfully uploaded ~/MegaFrogRace_Server/release-na to AWS GameLift
Build ID: build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

## 示例 2：上传 Windows 游戏服务器版本

以下upload-build示例将 Windows 游戏服务器编译文件从目录上传到 GameLift 服务并创建构建记录。

```
aws gamelift upload-build \
  --name MegaFrogRaceServer.NA \
  --build-version 2.0.1 \
  --build-root C:\MegaFrogRace_Server\release-na \
  --operating-system WINDOWS_2012
  --server-sdk-version 4.0.2
```

输出：

```
Uploading C:\MegaFrogRace_Server\release-na: 16.0 KiB / 74.6 KiB (21.45%)
Uploading C:\MegaFrogRace_Server\release-na: 32.0 KiB / 74.6 KiB (42.89%)
Uploading C:\MegaFrogRace_Server\release-na: 48.0 KiB / 74.6 KiB (64.34%)
Uploading C:\MegaFrogRace_Server\release-na: 64.0 KiB / 74.6 KiB (85.79%)
Uploading C:\MegaFrogRace_Server\release-na: 74.6 KiB / 74.6 KiB (100.00%)
Successfully uploaded C:\MegaFrogRace_Server\release-na to AWS GameLift
Build ID: build-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

有关更多信息，请参阅《Amazon GameLift 开发者指南》GameLift中的[将自定义服务器版本上传到](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UploadBuild](#)中的。

## 使用全球加速器示例 AWS CLI

以下代码示例向您展示了如何使用与 Global Accelerator AWS Command Line Interface 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## add-custom-routing-endpoints

以下代码示例演示了如何使用 add-custom-routing-endpoints。

AWS CLI

将 VPC 子网终端节点添加到自定义路由加速器的终端节点组中

以下add-custom-routing-endpoints示例将 VPC 子网终端节点添加到自定义路由加速器的终端节点组中。

```
aws globalaccelerator add-custom-routing-endpoints \  
  --endpoint-group-arn  
  arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-  
abcd-1234abcdefggh/listener/0123vxyz/endpoint-group/4321abcd \  
  --endpoint-configurations "EndpointId=subnet-1234567890abcdef0"
```

输出：

```
{  
  "EndpointDescriptions": [  
    {  
      "EndpointId": "subnet-1234567890abcdef0"  
    }  
  ],  
  
  "EndpointGroupArn": "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-  
abcd-1234-abcd-1234abcdefggh/listener/0123vxyz/endpoint-group/4321abcd"  
}
```

有关更多信息，请参阅《[全球加速器开发者指南](#)》中的 [AWS 全球加速器中用于自定义路由加速器的 VPC 子网终端节点](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddCustomRoutingEndpoints](#)中的。

## advertise-byoip-cidr

以下代码示例演示了如何使用 advertise-byoip-cidr。

### AWS CLI

#### 宣传地址范围

以下advertise-byoip-cidr示例请求 AWS 宣传您为 AWS 资源配置的地址范围。

```
aws globalaccelerator advertise-byoip-cidr \  
  --cidr 198.51.100.0/24
```

输出：

```
{  
  "ByoipCidr": {  
    "Cidr": "198.51.100.0/24",  
    "State": "PENDING_ADVERTISING"  
  }  
}
```

有关更多信息，请参阅《[AWS 全球加速器开发者指南](#)》中的[自带AWS 全球加速器的 IP 地址](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdvertiseByoipCidr](#)中的。

## allow-custom-routing-traffic

以下代码示例演示了如何使用 allow-custom-routing-traffic。

### AWS CLI

允许流向自定义路由加速器的 VPC 子网中的特定 Amazon EC2 实例目的地

以下allow-custom-routing-traffic示例指定允许流量流向某些 Amazon EC2 实例（目标）IP 地址，自定义路由加速器中的 VPC 子网终端节点可以接收流量。

```
aws globalaccelerator allow-custom-routing-traffic \  
  --target-ip-addresses 198.51.100.0/24
```

```
--endpoint-group-arn
arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh/listener/0123vxyz/endpoint-group/ab88888example \
--endpoint-id subnet-abcd123example \
--destination-addresses "172.31.200.6" "172.31.200.7" \
--destination-ports 80 81
```

此命令不生成任何输出。

有关更多信息，请参阅《全球加速器开发者指南》中的 [AWS 全球加速器中用于自定义路由加速器的 VPC 子网终端节点](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AllowCustomRoutingTraffic](#) 中的。

## create-accelerator

以下代码示例演示了如何使用 `create-accelerator`。

### AWS CLI

#### 创建加速器

以下 `create-accelerator` 示例创建了一个带有两个标签和两个 BYOIP 静态 IP 地址的加速器。您必须指定要创建或更新加速器的 US-West-2 (Oregon) 区域。

```
aws globalaccelerator create-accelerator \
  --name ExampleAccelerator \
  --tags Key="Name",Value="Example Name" Key="Project",Value="Example Project" \
  --ip-addresses 192.0.2.250 198.51.100.52
```

输出：

```
{
  "Accelerator": {
    "AcceleratorArn":
      "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
      abcd-1234abcdefgh",
    "IpAddressType": "IPV4",
    "Name": "ExampleAccelerator",
    "Enabled": true,
    "Status": "IN_PROGRESS",
    "IpSets": [
```

```
{
  "IpAddresses": [
    "192.0.2.250",
    "198.51.100.52"
  ],
  "IpFamily": "IPv4"
},
{
  "DnsName": "a1234567890abcdef.awsglobalaccelerator.com",
  "CreatedTime": 1542394847.0,
  "LastModifiedTime": 1542394847.0
}
}
```

有关更多信息，请参阅《[AWS 全球加速器](#)开发者指南》中的AWS 全球加速器。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAccelerator](#)中的。

## create-custom-routing-accelerator

以下代码示例演示了如何使用 create-custom-routing-accelerator。

### AWS CLI

#### 创建自定义路由加速器

以下create-custom-routing-accelerator示例使用Name和标签创建自定义路由加速器Project。

```
aws globalaccelerator create-custom-routing-accelerator \
  --name ExampleCustomRoutingAccelerator \
  --tags Key="Name",Value="Example Name" Key="Project",Value="Example Project" \
  --ip-addresses 192.0.2.250 198.51.100.52
```

输出：

```
{
  "Accelerator": {
    "AcceleratorArn":
      "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
      abcd-1234abcdefgh",
    "IpAddressType": "IPV4",
```

```
    "Name": "ExampleCustomRoutingAccelerator",
    "Enabled": true,
    "Status": "IN_PROGRESS",
    "IpSets": [
      {
        "IpAddresses": [
          "192.0.2.250",
          "198.51.100.52"
        ],
        "IpFamily": "IPv4"
      }
    ],
    "DnsName": "a1234567890abcdef.awsglobalaccelerator.com",
    "CreatedTime": 1542394847.0,
    "LastModifiedTime": 1542394847.0
  }
}
```

有关更多信息，请参阅《[全球加速器开发者指南](#)》中的 [AWS 全球加速器中的 AWS 自定义路由加速器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateCustomRoutingAccelerator](#) 中的。

## create-custom-routing-endpoint-group

以下代码示例演示了如何使用 create-custom-routing-endpoint-group。

### AWS CLI

为自定义路由加速器创建终端节点组

以下 create-custom-routing-endpoint-group 示例为自定义路由加速器创建终端节点组。

```
aws globalaccelerator create-custom-routing-endpoint-group \
  --listener-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh/listener/0123vxyz \
  --endpoint-group-region us-east-2 \
  --destination-configurations "FromPort=80,ToPort=81,Protocols=TCP,UDP"
```

输出：

```
{
  "EndpointGroup": {
```

```

    "EndpointGroupArn":
      "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh/listener/0123vxyz/endpoint-group/4321abcd",
    "EndpointGroupRegion": "us-east-2",
    "DestinationDescriptions": [
      {
        "FromPort": 80,
        "ToPort": 81,
        "Protocols": [
          "TCP",
          "UDP"
        ]
      }
    ],
    "EndpointDescriptions": []
  }
}

```

有关更多信息，请参阅《[全球加速器开发者指南](#)》中的 [AWS 全球加速器中自定义路由加速器的终端节点组](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateCustomRoutingEndpointGroup](#) 中的。

## create-custom-routing-listener

以下代码示例演示了如何使用 create-custom-routing-listener。

### AWS CLI

为自定义路由加速器创建监听器

以下 create-custom-routing-listener 示例为自定义路由加速器创建端口范围为 5000 到 10000 的侦听器。

```

aws globalaccelerator create-custom-routing-listener \
  --accelerator-arn arn:aws:globalaccelerator::123456789012:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh \
  --port-ranges FromPort=5000,ToPort=10000

```

输出：

```
{
```

```
"Listener": {
  "PortRange": [
    "FromPort": 5000,
    "ToPort": 10000
  ],
  "ListenerArn":
  "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh/listener/0123vxyz"
}
```

有关更多信息，请参阅 [《AWS 全球加速器开发者指南》中的 Global Accelerator 中的自定义路由加速器监听器](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateCustomRoutingListener](#) 中的。

## create-endpoint-group

以下代码示例演示了如何使用 create-endpoint-group。

### AWS CLI

#### 创建终端节点组

以下 create-endpoint-group 示例创建了一个包含一个终端节点的终端节点组。

```
aws globalaccelerator create-endpoint-group \
  --listener-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh/listener/0123vxyz \
  --endpoint-group-region us-east-1 \
  --endpoint-configurations EndpointId=i-1234567890abcdef0,Weight=128
```

输出：

```
{
  "EndpointGroup": {
    "TrafficDialPercentage": 100.0,
    "EndpointDescriptions": [
      {
        "Weight": 128,
        "EndpointId": "i-1234567890abcdef0"
      }
    ]
  }
}
```



```

    ],
    "EndpointGroupArn":
    "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh/listener/0123vxyz/endpoint-group/098765zyxwvu",
    "EndpointGroupRegion": "us-east-1"
  }
}

```

有关更多信息，请参阅《[AWS 全球加速器开发者指南](#)》中的[AWS 全球加速器中的端点组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateEndpointGroup](#)中的。

## create-listener

以下代码示例演示了如何使用 create-listener。

### AWS CLI

#### 创建监听器

以下create-listener示例创建了一个具有两个端口的监听器。

```

aws globalaccelerator create-listener \
  --accelerator-arn arn:aws:globalaccelerator::123456789012:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh \
  --port-ranges FromPort=80,ToPort=80 FromPort=81,ToPort=81 \
  --protocol TCP

```

输出：

```

{
  "Listener": {
    "PortRanges": [
      {
        "ToPort": 80,
        "FromPort": 80
      },
      {
        "ToPort": 81,
        "FromPort": 81
      }
    ]
  },

```

```
    "ClientAffinity": "NONE",
    "Protocol": "TCP",
    "ListenerArn":
      "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
      abcd-1234abcdefgh/listener/0123vxyz"
  }
}
```

有关更多信息，请参阅《AWS 全球加速器开发者指南》中的 [AWS 全球加速器中的监听器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateListener](#) 中的。

## deny-custom-routing-traffic

以下代码示例演示了如何使用 deny-custom-routing-traffic。

### AWS CLI

指定无法在自定义路由加速器中接收流量的目标地址

以下 deny-custom-routing-traffic 示例指定了无法接收自定义路由加速器流量的子网终端节点中的一个或多个目标地址。要指定多个目的地址，请用空格分隔各个地址。成功 deny-custom-routing-traffic 呼叫没有回应。

```
aws globalaccelerator deny-custom-routing-traffic \
  --endpoint-group-arn
  "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
  abcd-1234abcdefgh/listener/0123vxyz/endpoint-group/ab8888example" \
  --endpoint-id "subnet-abcd123example" \
  --destination-addresses "198.51.100.52"
```

此命令不生成任何输出。

有关更多信息，请参阅《全球加速器开发者指南》中的 [AWS 全球加速器中用于自定义路由加速器的 VPC 子网终端节点](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DenyCustomRoutingTraffic](#) 中的。

## deprovision-byoip-cidr

以下代码示例演示了如何使用 deprovision-byoip-cidr。

## AWS CLI

### 取消配置地址范围

以下deprovision-byoip-cidr示例释放了您预配置的用于 AWS 资源的指定地址范围。

```
aws globalaccelerator deprovision-byoip-cidr \  
  --cidr "198.51.100.0/24"
```

输出：

```
{  
  "ByoipCidr": {  
    "Cidr": "198.51.100.0/24",  
    "State": "PENDING_DEPROVISIONING"  
  }  
}
```

有关更多信息，请参阅《[全球加速器开发者指南](#)》中的“[AWS 在全球加速器中AWS 自带 IP 地址](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeprovisionByoipCidr](#)中的。

## describe-accelerator-attributes

以下代码示例演示了如何使用 describe-accelerator-attributes。

## AWS CLI

### 描述加速器的属性

以下describe-accelerator-attributes示例检索加速器的属性详细信息。

```
aws globalaccelerator describe-accelerator-attributes \  
  --accelerator-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-  
abcd-1234-abcd-1234abcdefg
```

输出：

```
{
```

```
"AcceleratorAttributes": {
  "FlowLogsEnabled": true
  "FlowLogsS3Bucket": flowlogs-abc
  "FlowLogsS3Prefix": bucketprefix-abc
}
```

有关更多信息，请参阅《[AWS 全球加速器开发者指南](#)》中的AWS 全球加速器。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAcceleratorAttributes](#)中的。

## describe-accelerator

以下代码示例演示了如何使用 describe-accelerator。

### AWS CLI

#### 描述加速器

以下describe-accelerator示例检索有关指定加速器的详细信息。

```
aws globalaccelerator describe-accelerator \
  --accelerator-arn arn:aws:globalaccelerator::123456789012:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh
```

输出：

```
{
  "Accelerator": {
    "AcceleratorArn":
"arn:aws:globalaccelerator::123456789012:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh",
    "IpAddressType": "IPV4",
    "Name": "ExampleAccelerator",
    "Enabled": true,
    "Status": "IN_PROGRESS",
    "IpSets": [
      {
        "IpAddresses": [
          "192.0.2.250",
          "198.51.100.52"
        ]
      }
    ]
  }
}
```

```
        "IpFamily": "IPv4"
      }
    ],
    "DnsName": "a1234567890abcdef.awsglobalaccelerator.com",
    "CreatedTime": 1542394847,
    "LastModifiedTime": 1542395013
  }
}
```

有关更多信息，请参阅《[AWS 全球加速器开发者指南](#)》中的AWS 全球加速器。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAccelerator](#)中的。

## describe-custom-routing-accelerator-attributes

以下代码示例演示了如何使用 describe-custom-routing-accelerator-attributes。

### AWS CLI

描述自定义路由加速器的属性

以下describe-custom-routing-accelerator-attributes示例描述了自定义路由加速器的属性。

```
aws globalaccelerator describe-custom-routing-accelerator-attributes \
  --accelerator-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh
```

输出：

```
{
  "AcceleratorAttributes": {
    "FlowLogsEnabled": false
  }
}
```

有关更多信息，请参阅《[全球加速器开发者指南](#)》中的 [AWS 全球加速器中的AWS 自定义路由加速器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCustomRoutingAcceleratorAttributes](#)中的。

## describe-custom-routing-accelerator

以下代码示例演示了如何使用 `describe-custom-routing-accelerator`。

### AWS CLI

#### 描述自定义路由加速器

以下 `describe-custom-routing-accelerator` 示例检索有关指定自定义路由加速器的详细信息。

```
aws globalaccelerator describe-custom-routing-accelerator \
  --accelerator-arn arn:aws:globalaccelerator::123456789012:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh
```

输出：

```
{
  "Accelerator": {
    "AcceleratorArn":
    "arn:aws:globalaccelerator::123456789012:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh",
    "IpAddressType": "IPV4",
    "Name": "ExampleCustomRoutingAccelerator",
    "Enabled": true,
    "Status": "IN_PROGRESS",
    "IpSets": [
      {
        "IpAddresses": [
          "192.0.2.250",
          "198.51.100.52"
        ],
        "IpFamily": "IPv4"
      }
    ],
    "DnsName": "a1234567890abcdef.awsglobalaccelerator.com",
    "CreatedTime": 1542394847,
    "LastModifiedTime": 1542395013
  }
}
```

有关更多信息，请参阅《[全球加速器开发者指南](#)》中的 [AWS 全球加速器中的 AWS 自定义路由加速器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCustomRoutingAccelerator](#)中的。

## describe-custom-routing-endpoint-group

以下代码示例演示了如何使用 describe-custom-routing-endpoint-group。

### AWS CLI

描述自定义路由加速器的终端节点组

以下describe-custom-routing-endpoint-group示例描述了自定义路由加速器的终端节点组。

```
aws globalaccelerator describe-custom-routing-endpoint-group \
  --endpoint-group-arn
  arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
  abcd-1234abcdefgh/listener/6789vxyz/endpoint-group/ab8888example
```

输出：

```
{
  "EndpointGroup": {
    "EndpointGroupArn":
    "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
    abcd-1234abcdefgh/listener/6789vxyz/endpoint-group/ab8888example",
    "EndpointGroupRegion": "us-east-2",
    "DestinationDescriptions": [
      {
        "FromPort": 5000,
        "ToPort": 10000,
        "Protocols": [
          "UDP"
        ]
      }
    ],
    "EndpointDescriptions": [
      {
        "EndpointId": "subnet-1234567890abcdef0"
      }
    ]
  }
}
```

有关更多信息，请参阅《[全球加速器开发者指南](#)》中的 [AWS 全球加速器中自定义路由加速器的终端节点组](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCustomRoutingEndpointGroup](#)中的。

## describe-custom-routing-listener

以下代码示例演示了如何使用 describe-custom-routing-listener。

### AWS CLI

描述自定义路由加速器的监听器

以下describe-custom-routing-listener示例描述了自定义路由加速器的监听器。

```
aws globalaccelerator describe-custom-routing-listener \
  --listener-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh/listener/abcdef1234
```

输出：

```
{
  "Listener": {
    "PortRanges": [
      "FromPort": 5000,
      "ToPort": 10000
    ],
    "ListenerArn":
    "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh/listener/abcdef1234"
  }
}
```

有关更多信息，请参阅《[AWS 全球加速器开发者指南](#)》中的 [Global Accelerator 中的自定义路由加速器监听器](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCustomRoutingListener](#)中的。

## describe-endpoint-group

以下代码示例演示了如何使用 describe-endpoint-group。



## AWS CLI

### 描述终端节点组

以下describe-endpoint-group示例检索具有以下终端节点的终端节点组的详细信息：Amazon EC2 实例、ALB 和 NLB。

```
aws globalaccelerator describe-endpoint-group \  
  --endpoint-group-arn \  
  arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-  
abcd-1234abcdefg/ listener/6789vxyz-vxyz-6789-vxyz-6789lmnopqrs/endpoint-group/  
ab8888example
```

输出：

```
{  
  "EndpointGroup": {  
    "TrafficDialPercentage": 100.0,  
    "EndpointDescriptions": [  
      {  
        "Weight": 128,  
        "EndpointId": "i-1234567890abcdef0"  
      },  
      {  
        "Weight": 128,  
        "EndpointId": "arn:aws:elasticloadbalancing:us-  
east-1:000123456789:loadbalancer/app/ALBTesting/alb01234567890xyz"  
      },  
      {  
        "Weight": 128,  
        "EndpointId": "arn:aws:elasticloadbalancing:us-  
east-1:000123456789:loadbalancer/net/NLBTesting/alb01234567890qrs"  
      }  
    ],  
    "EndpointGroupArn":  
    "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-  
abcd-1234abcdefg/ listener/6789vxyz-vxyz-6789-vxyz-6789lmnopqrs/endpoint-  
group/4321abcd-abcd-4321-abcd-4321abcdefg",  
    "EndpointGroupRegion": "us-east-1"  
  }  
}
```

有关更多信息，请参阅 [《AWS 全球加速器开发者指南》](#) 中的 [AWS 全球加速器中的端点组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEndpointGroup](#)中的。

## describe-listener

以下代码示例演示了如何使用 describe-listener。

### AWS CLI

#### 描述听众

以下describe-listener示例描述了一个监听器。

```
aws globalaccelerator describe-listener \  
  --listener-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-  
abcd-1234-abcd-1234abcdefgh/listener/abcdef1234
```

输出：

```
{  
  "Listener": {  
    "ListenerArn":  
    "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-  
abcd-1234abcdefgh/listener/abcdef1234",  
    "PortRanges": [  
      {  
        "FromPort": 80,  
        "ToPort": 80  
      }  
    ],  
    "Protocol": "TCP",  
    "ClientAffinity": "NONE"  
  }  
}
```

有关更多信息，请参阅《AWS 全球加速器开发者指南》中的 AWS 全球加速器[中的监听器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeListener](#)中的。

## list-accelerators

以下代码示例演示了如何使用 list-accelerators。

## AWS CLI

### 列出您的加速器

以下`list-accelerators`示例列出了您 AWS 账户中的加速器。该账户有两个加速器。

```
aws globalaccelerator list-accelerators
```

输出：

```
{
  "Accelerators": [
    {
      "AcceleratorArn":
"arn:aws:globalaccelerator::012345678901:accelerator/5555abcd-abcd-5555-
abcd-5555EXAMPLE1",
      "Name": "TestAccelerator",
      "IpAddressType": "IPV4",
      "Enabled": true,
      "IpSets": [
        {
          "IpFamily": "IPv4",
          "IpAddresses": [
            "192.0.2.250",
            "198.51.100.52"
          ]
        }
      ]
    },
    {
      "AcceleratorArn":
"arn:aws:globalaccelerator::888888888888:accelerator/8888abcd-abcd-8888-
abcd-8888EXAMPLE2",
      "Name": "ExampleAccelerator",
      "IpAddressType": "IPV4",
      "Enabled": true,
      "IpSets": [
        {
          "IpFamily": "IPv4",
```



```
  ]
}
```

有关更多信息，请参阅《[全球加速器开发者指南](#)》中的“[AWS 在全球加速器中AWS 自带 IP 地址](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListByoipCidr](#)中的。

## **list-custom-routing-accelerators**

以下代码示例演示了如何使用 list-custom-routing-accelerators。

### AWS CLI

列出您的自定义路由加速器

以下list-custom-routing-accelerators示例列出了 AWS 账户中的自定义路由加速器。

```
aws globalaccelerator list-custom-routing-accelerators
```

输出：

```
{
  "Accelerators": [
    {
      "AcceleratorArn":
      "arn:aws:globalaccelerator::012345678901:accelerator/5555abcd-abcd-5555-
abcd-5555EXAMPLE1",
      "Name": "TestCustomRoutingAccelerator",
      "IpAddressType": "IPV4",
      "Enabled": true,
      "IpSets": [
        {
          "IpFamily": "IPv4",
          "IpAddresses": [
            "192.0.2.250",
            "198.51.100.52"
          ]
        }
      ],
      "DnsName": "5a5a5a5a5a5a5a5a.awsglobalaccelerator.com",
      "Status": "DEPLOYED",
      "CreatedTime": 1552424416.0,
    }
  ]
}
```

```
        "LastModifiedTime": 1569375641.0
      },
      {
        "AcceleratorArn":
"arn:aws:globalaccelerator::888888888888:accelerator/8888abcd-abcd-8888-
abcd-8888EXAMPLE2",
        "Name": "ExampleCustomRoutingAccelerator",
        "IpAddressType": "IPv4",
        "Enabled": true,
        "IpSets": [
          {
            "IpFamily": "IPv4",
            "IpAddresses": [
              "192.0.2.100",
              "198.51.100.10"
            ]
          }
        ],
        "DnsName": "6a6a6a6a6a6a6a6a.awsglobalaccelerator.com",
        "Status": "DEPLOYED",
        "CreatedTime": 1575585564.0,
        "LastModifiedTime": 1579809243.0
      },
    ]
  }
}
```

有关更多信息，请参阅《[全球加速器开发者指南](#)》中的 [AWS 全球加速器中的 AWS 自定义路由加速器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListCustomRoutingAccelerators](#) 中的。

## list-custom-routing-endpoint-groups

以下代码示例演示了如何使用 list-custom-routing-endpoint-groups。

### AWS CLI

列出自定义路由加速器中监听器的终端节点组

以下 list-custom-routing-endpoint-groups 示例列出了自定义路由加速器中监听器的终端节点组。

```
aws globalaccelerator list-custom-routing-endpoint-groups \
```

```
--listener-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh/listener/abcdef1234
```

输出：

```
{
  "EndpointGroups": [
    {
      "EndpointGroupArn":
      "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh/listener/abcdef1234/endpoint-group/ab88888example",
      "EndpointGroupRegion": "eu-central-1",
      "DestinationDescriptions": [
        {
          "FromPort": 80,
          "ToPort": 80,
          "Protocols": [
            "TCP",
            "UDP"
          ]
        }
      ]
      "EndpointDescriptions": [
        {
          "EndpointId": "subnet-abcd123example"
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅《[全球加速器开发者指南](#)》中的 [AWS 全球加速器中自定义路由加速器的终端节点组](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCustomRoutingEndpointGroups](#)中的。

## list-custom-routing-listeners

以下代码示例演示了如何使用 list-custom-routing-listeners。

### AWS CLI

列出自定义路由加速器的监听器

以下 `list-custom-routing-listeners` 示例列出了自定义路由加速器的监听器。

```
aws globalaccelerator list-custom-routing-listeners \  
  --accelerator-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-  
abcd-1234-abcd-1234abcdefgh
```

输出：

```
{  
  "Listeners": [  
    {  
      "ListenerArn":  
"arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-  
abcd-1234abcdefgh/listener/abcdef1234",  
      "PortRanges": [  
        {  
          "FromPort": 5000,  
          "ToPort": 10000  
        }  
      ],  
      "Protocol": "TCP"  
    }  
  ]  
}
```

有关更多信息，请参阅 [《AWS 全球加速器开发者指南》](#) 中的 [Global Accelerator 中的自定义路由加速器监听器](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListCustomRoutingListeners](#) 中的。

## **list-custom-routing-port-mappings-by-destination**

以下代码示例演示了如何使用 `list-custom-routing-port-mappings-by-destination`。

AWS CLI

列出特定自定义路由加速器目标的端口映射

以下 `list-custom-routing-port-mappings-by-destination` 示例提供了自定义路由加速器的特定目标 EC2 服务器（目标地址）的端口映射。

```
aws globalaccelerator list-custom-routing-port-mappings-by-destination \  
  --accelerator-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-  
abcd-1234-abcd-1234abcdefgh
```



```
--endpoint-id subnet-abcd123example \  
--destination-address 198.51.100.52
```

输出：

```
{  
  "DestinationPortMappings": [  
    {  
      "AcceleratorArn":  
        "arn:aws:globalaccelerator::402092451327:accelerator/24ea29b8-  
d750-4489-8919-3095f3c4b0a7",  
      "AcceleratorSocketAddresses": [  
        {  
          "IpAddress": "192.0.2.250",  
          "Port": 65514  
        },  
        {  
          "IpAddress": "192.10.100.99",  
          "Port": 65514  
        }  
      ],  
      "EndpointGroupArn":  
        "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-  
abcd-1234abcdefgh/listener/0123vxyz/endpoint-group/ab88888example",  
      "EndpointId": "subnet-abcd123example",  
      "EndpointGroupRegion": "us-west-2",  
      "DestinationSocketAddress": {  
        "IpAddress": "198.51.100.52",  
        "Port": 80  
      },  
      "IpAddressType": "IPv4",  
      "DestinationTrafficState": "ALLOW"  
    }  
  ]  
}
```

有关更多信息，[请参阅《全球加速器开发者指南》中的自定义路由加速器 AWS 在全球加速器中的 AWS 工作原理。](#)

- 有关 API 的详细信息，[请参阅AWS CLI 命令参考ListCustomRoutingPortMappingsByDestination](#)中的。

## list-custom-routing-port-mappings

以下代码示例演示了如何使用 `list-custom-routing-port-mappings`。

### AWS CLI

在自定义路由加速器中列出端口映射

以下 `list-custom-routing-port-mappings` 示例提供了自定义路由加速器中端口映射的部分列表。

```
aws globalaccelerator list-custom-routing-port-mappings \
  --accelerator-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh
```

输出：

```
{
  "PortMappings": [
    {
      "AcceleratorPort": 40480,
      "EndpointGroupArn":
      "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh/listener/0123vxyz/endpoint-group/098765zyxwvu",
      "EndpointId": "subnet-1234567890abcdef0",
      "DestinationSocketAddress": {
        "IpAddress": "192.0.2.250",
        "Port": 80
      },
      "Protocols": [
        "TCP",
        "UDP"
      ],
      "DestinationTrafficState": "ALLOW"
    }
  ]
}
```

```

        "Port": 80
      },
      "Protocols": [
        "TCP",
        "UDP"
      ],
      "DestinationTrafficState": "ALLOW"
    }
  ]
}

```

有关更多信息，[请参阅《全球加速器开发者指南》中的自定义路由加速器 AWS 在全球加速器中的 AWS 工作原理。](#)

- 有关 API 的详细信息，[请参阅 AWS CLI 命令参考 `ListCustomRoutingPortMappings` 中的。](#)

## list-endpoint-groups

以下代码示例演示了如何使用 `list-endpoint-groups`。

### AWS CLI

#### 列出终端节点组

以下 `list-endpoint-groups` 示例列出了监听器的终端节点组。该监听器有两个终端节点组。

```

aws globalaccelerator --region us-west-2 list-endpoint-groups \
  --listener-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh/listener/abcdef1234

```

#### 输出：

```

{
  "EndpointGroups": [
    {
      "EndpointGroupArn":
"arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh/listener/abcdef1234/endpoint-group/ab88888example",
      "EndpointGroupRegion": "eu-central-1",
      "EndpointDescriptions": [],
      "TrafficDialPercentage": 100.0,
      "HealthCheckPort": 80,
    }
  ]
}

```

```

        "HealthCheckProtocol": "TCP",
        "HealthCheckIntervalSeconds": 30,
        "ThresholdCount": 3
    }
  {
    "EndpointGroupArn":
    "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh/listener/abcdef1234/endpoint-group/ab99999example",
    "EndpointGroupRegion": "us-east-1",
    "EndpointDescriptions": [],
    "TrafficDialPercentage": 50.0,
    "HealthCheckPort": 80,
    "HealthCheckProtocol": "TCP",
    "HealthCheckIntervalSeconds": 30,
    "ThresholdCount": 3
  }
]
}

```

有关更多信息，请参阅 [《AWS 全球加速器开发者指南》中的AWS 全球加速器中的端点组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListEndpointGroups](#)中的。

## list-listeners

以下代码示例演示了如何使用 list-listeners。

### AWS CLI

列出听众

以下list-listeners示例列出了加速器的监听器。

```

aws globalaccelerator list-listeners \
  --accelerator-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh

```

输出：

```

{
  "Listeners": [
    {

```

```
    "ListenerArn":
      "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh/listener/abcdef1234",
      "PortRanges": [
        {
          "FromPort": 80,
          "ToPort": 80
        }
      ],
      "Protocol": "TCP",
      "ClientAffinity": "NONE"
    }
  ]
}
```

有关更多信息，请参阅《AWS 全球加速器开发者指南》中的 [AWS 全球加速器中的监听器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListListeners](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出加速器的标签

以下 `list-tags-for-resource` 示例列出了特定加速器的标签。

```
aws globalaccelerator list-tags-for-resource \
  --accelerator-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh
```

输出：

```
{
  "Tags": [
    {
      "Key": "Project",
      "Value": "A123456"
    }
  ]
}
```

```
}
```

有关更多信息，请参阅《全球加速器开发者指南》中的“[AWS AWS 在全球加速器中添加标签](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## provision-byoip-cidr

以下代码示例演示了如何使用 provision-byoip-cidr。

### AWS CLI

#### 配置地址范围

以下provision-byoip-cidr示例预置了用于您的 AWS 资源的指定地址范围。

```
aws globalaccelerator provision-byoip-cidr \  
  --cidr 192.0.2.250/24 \  
  --cidr-authorization-context Message="$text_message",Signature="$signed_message"
```

输出：

```
{  
  "ByoipCidr": {  
    "Cidr": "192.0.2.250/24",  
    "State": "PENDING_PROVISIONING"  
  }  
}
```

有关更多信息，请参阅《全球加速器开发者指南》中的“[AWS 在全球加速器中AWS 自带 IP 地址](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ProvisionByoipCidr](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

#### 标记加速器

以下tag-resource示例向加速器添加标签 Name 和 Project，以及每个标签的相应值。

```
aws globalaccelerator tag-resource \  
  --resource-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-  
abcd-1234-abcd-1234abcdefgh \  
  --tags Key="Name",Value="Example Name" Key="Project",Value="Example Project"
```

此命令不生成任何输出。

有关更多信息，请参阅《全球加速器开发者指南》中的“AWS AWS 在全球加速器[中添加标签](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从加速器中移除标签

以下untag-resource示例从加速器中移除标签“名称”和“项目”。

```
aws globalaccelerator untag-resource \  
  --resource-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-  
abcd-1234-abcd-1234abcdefgh \  
  --tag-keys Key="Name" Key="Project"
```

此命令不生成任何输出。

有关更多信息，请参阅《全球加速器开发者指南》中的“AWS AWS 在全球加速器[中添加标签](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-accelerator-attributes

以下代码示例演示了如何使用 update-accelerator-attributes。

### AWS CLI

更新加速器的属性

以下update-accelerator-attributes示例更新加速器以启用流日志。您必须指定要创建或更新加速器属性的US-West-2 (Oregon)区域。

```
aws globalaccelerator update-accelerator-attributes \  
  --accelerator-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-  
abcd-1234-abcd-1234abcdefgh \  
  --flow-logs-enabled \  
  --flow-logs-s3-bucket flowlogs-abc \  
  --flow-logs-s3-prefix bucketprefix-abc
```

输出：

```
{  
  "AcceleratorAttributes": {  
    "FlowLogsEnabled": true  
    "FlowLogsS3Bucket": flowlogs-abc  
    "FlowLogsS3Prefix": bucketprefix-abc  
  }  
}
```

有关更多信息，请参阅《[AWS 全球加速器](#)开发者指南》中的AWS 全球加速器。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAcceleratorAttributes](#)中的。

## update-accelerator

以下代码示例演示了如何使用 update-accelerator。

### AWS CLI

#### 更新加速器

以下update-accelerator示例修改加速器以将加速器名称更改为。ExampleAcceleratorNew您必须指定要创建或更新加速器的US-West-2 (Oregon)区域。

```
aws globalaccelerator update-accelerator \  
  --accelerator-arn arn:aws:globalaccelerator::123456789012:accelerator/1234abcd-  
abcd-1234-abcd-1234abcdefgh \  
  --name ExampleAcceleratorNew
```

输出：

```
{  
  "Accelerator": {
```



```
    "AcceleratorArn":
      "arn:aws:globalaccelerator::123456789012:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh",
      "IpAddressType": "IPv4",
      "Name": "ExampleAcceleratorNew",
      "Enabled": true,
      "Status": "IN_PROGRESS",
      "IpSets": [
        {
          "IpAddresses": [
            "192.0.2.250",
            "198.51.100.52"
          ],
          "IpFamily": "IPv4"
        }
      ],
      "DnsName": "a1234567890abcdef.awsglobalaccelerator.com",
      "CreatedTime": 1232394847,
      "LastModifiedTime": 1232395654
    }
  }
```

有关更多信息，请参阅《[AWS 全球加速器](#)开发者指南》中的AWS 全球加速器。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAccelerator](#)中的。

## update-custom-routing-accelerator-attributes

以下代码示例演示了如何使用 update-custom-routing-accelerator-attributes。

### AWS CLI

更新自定义路由加速器的属性

以下update-custom-routing-accelerator-attributes示例更新自定义路由加速器以启用流日志。

```
aws globalaccelerator update-custom-routing-accelerator-attributes \
  --accelerator-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh \
  --flow-logs-enabled \
  --flow-logs-s3-bucket flowlogs-abc \
  --flow-logs-s3-prefix bucketprefix-abc
```

输出：

```
{
  "AcceleratorAttributes": {
    "FlowLogsEnabled": true
    "FlowLogsS3Bucket": flowlogs-abc
    "FlowLogsS3Prefix": bucketprefix-abc
  }
}
```

有关更多信息，请参阅《[全球加速器开发者指南](#)》中的 [AWS 全球加速器中的AWS 自定义路由加速器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateCustomRoutingAcceleratorAttributes](#)中的。

## update-custom-routing-accelerator

以下代码示例演示了如何使用 update-custom-routing-accelerator。

### AWS CLI

#### 更新自定义路由加速器

以下update-custom-routing-accelerator示例修改自定义路由加速器以更改加速器名称。

```
aws globalaccelerator --region us-west-2 update-custom-routing-accelerator \
  --accelerator-arn arn:aws:globalaccelerator::123456789012:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh \
  --name ExampleCustomRoutingAcceleratorNew
```

输出：

```
{
  "Accelerator": {
    "AcceleratorArn":
    "arn:aws:globalaccelerator::123456789012:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh",
    "IpAddressType": "IPV4",
    "Name": "ExampleCustomRoutingAcceleratorNew",
    "Enabled": true,
    "Status": "IN_PROGRESS",
```

```
    "IpSets": [
      {
        "IpAddresses": [
          "192.0.2.250",
          "198.51.100.52"
        ],
        "IpFamily": "IPv4"
      }
    ],
    "DnsName": "a1234567890abcdef.awsglobalaccelerator.com",
    "CreatedTime": 1232394847,
    "LastModifiedTime": 1232395654
  }
}
```

有关更多信息，请参阅《全球[加速器开发者指南](#)》中的 [AWS 全球加速器中的AWS 自定义路由加速器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateCustomRoutingAccelerator](#)中的。

## update-custom-routing-listener

以下代码示例演示了如何使用 `update-custom-routing-listener`。

### AWS CLI

#### 更新自定义路由加速器的监听器

以下`update-custom-routing-listener`示例更新侦听器以更改端口范围。

```
aws globalaccelerator update-custom-routing-listener \
  --listener-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-
abcd-1234-abcd-1234abcdefgh/listener/0123vxyz \
  --port-ranges FromPort=10000,ToPort=20000
```

输出：

```
{
  "Listener": {
    "ListenerArn":
    "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh/listener/0123vxyz
    "PortRanges": [
```

```

        {
            "FromPort": 10000,
            "ToPort": 20000
        }
    ],
    "Protocol": "TCP"
}

```

有关更多信息，请参阅 [《AWS 全球加速器开发者指南》](#) 中的 [Global Accelerator 中的自定义路由加速器监听器](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateCustomRoutingListener](#) 中的。

## update-endpoint-group

以下代码示例演示了如何使用 update-endpoint-group。

### AWS CLI

#### 更新终端节点组

以下 update-endpoint-group 示例向终端节点组添加了三个终端节点：弹性 IP 地址、ALB 和 NLB。

```

aws globalaccelerator update-endpoint-group \
  --endpoint-group-arn
  arn:aws:globalaccelerator::123456789012:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh/listener/6789vxyz-vxyz-6789-vxyz-6789lmnopqrs/endpoint-group/
ab8888example \
  --endpoint-configurations \
    EndpointId=eipalloc-eip01234567890abc,Weight=128 \
    EndpointId=arn:aws:elasticloadbalancing:us-east-1:000123456789:loadbalancer/
app/ALBTesting/alb01234567890xyz,Weight=128 \
    EndpointId=arn:aws:elasticloadbalancing:us-east-1:000123456789:loadbalancer/
net/NLBTesting/alb01234567890qrs,Weight=128

```

输出：

```

{
  "EndpointGroup": {
    "TrafficDialPercentage": 100,
    "EndpointDescriptions": [

```

```
    {
      "Weight": 128,
      "EndpointId": "eip01234567890abc"
    },
    {
      "Weight": 128,
      "EndpointId": "arn:aws:elasticloadbalancing:us-
east-1:000123456789:loadbalancer/app/ALBTesting/alb01234567890xyz"
    },
    {
      "Weight": 128,
      "EndpointId": "arn:aws:elasticloadbalancing:us-
east-1:000123456789:loadbalancer/net/NLBTesting/alb01234567890qrs"
    }
  ],
  "EndpointGroupArn":
  "arn:aws:globalaccelerator::123456789012:accelerator/1234abcd-abcd-1234-
abcd-1234abcdefgh/listener/6789vxyz-vxyz-6789-vxyz-6789lmnopqrs/endpoint-
group/4321abcd-abcd-4321-abcd-4321abcdefg",
  "EndpointGroupRegion": "us-east-1"
}
}
```

有关更多信息，请参阅 [《AWS 全球加速器开发者指南》](#) 中的 [AWS 全球加速器中的端点组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateEndpointGroup](#) 中的。

## update-listener

以下代码示例演示了如何使用 update-listener。

### AWS CLI

#### 更新监听器

以下 update-listener 示例更新侦听器以将端口更改为 100。

```
aws globalaccelerator update-listener \  
  --listener-arn arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-  
abcd-1234-abcd-1234abcdefgh/listener/0123vxyz \  
  --port-ranges FromPort=100,ToPort=100
```

输出：

```
{
  "Listener": {
    "ListenerArn":
      "arn:aws:globalaccelerator::012345678901:accelerator/1234abcd-abcd-1234-
      abcd-1234abcdefgh/listener/0123vxyz
    "PortRanges": [
      {
        "FromPort": 100,
        "ToPort": 100
      }
    ],
    "Protocol": "TCP",
    "ClientAffinity": "NONE"
  }
}
```

有关更多信息，请参阅《AWS 全球加速器开发者指南》中的 [AWS 全球加速器中的监听器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateListener](#) 中的。

## withdraw-byoip-cidr

以下代码示例演示了如何使用 `withdraw-byoip-cidr`。

### AWS CLI

#### 撤回地址范围

以下 `withdraw-byoip-cidr` 示例从 AWS 全球加速器中撤回了您之前发布的用于您的 AWS 资源的地址范围。

```
aws globalaccelerator withdraw-byoip-cidr \
  --cidr 192.0.2.250/24
```

输出：

```
{
  "ByoipCidr": {
    "Cidr": "192.0.2.250/24",
    "State": "PENDING_WITHDRAWING"
  }
}
```

有关更多信息，请参阅《[全球加速器开发者指南](#)》中的“[AWS 在全球加速器中AWS 自带 IP 地址](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[WithdrawByoipCidr](#)中的。

## AWS Glue 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Glue。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### batch-stop-job-run

以下代码示例演示了如何使用 batch-stop-job-run。

AWS CLI

停止作业运行

以下batch-stop-job-run示例停止作业运行。

```
aws glue batch-stop-job-run \  
  --job-name "my-testing-job" \  
  --job-run-id jr_852f1de1f29fb62e0ba4166c33970803935d87f14f96cfdee5089d5274a61d3f
```

输出：

```
{  
  "SuccessfulSubmissions": [  
    {
```

```

        "JobName": "my-testing-job",
        "JobRunId":
"jr_852f1de1f29fb62e0ba4166c33970803935d87f14f96cfdee5089d5274a61d3f"
    }
],
"Errors": [],
"ResponseMetadata": {
    "RequestId": "66bd6b90-01db-44ab-95b9-6aeff0e73d88",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
        "date": "Fri, 16 Oct 2020 20:54:51 GMT",
        "content-type": "application/x-amz-json-1.1",
        "content-length": "148",
        "connection": "keep-alive",
        "x-amzn-requestid": "66bd6b90-01db-44ab-95b9-6aeff0e73d88"
    },
    "RetryAttempts": 0
}
}

```

有关更多信息，请参阅《AWS Glue 开发人员指南》中的[任务运行](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[BatchStopJobRun](#)中的。

## create-connection

以下代码示例演示了如何使用 create-connection。

### AWS CLI

为 Glue 数据 AWS 存储创建连接

以下 create-connection 示例在 AWS Glue 数据目录中创建一个连接，该连接为 Kafka 数据存储提供连接信息。

```

aws glue create-connection \
  --connection-input '{ \
    "Name": "conn-kafka-custom", \
    "Description": "kafka connection with ssl to custom kafka", \
    "ConnectionType": "KAFKA", \
    "ConnectionProperties": { \
      "KAFKA_BOOTSTRAP_SERVERS": "<Kafka-broker-server-url>:<SSL-Port>", \
      "KAFKA_SSL_ENABLED": "true", \

```



```
    "KAFKA_CUSTOM_CERT": "s3://bucket/prefix/cert-file.pem" \
  }, \
  "PhysicalConnectionRequirements":{ \
    "SubnetId":"subnet-1234", \
    "SecurityGroupIdList":["sg-1234"], \
    "AvailabilityZone":"us-east-1a"} \
}' \
--region us-east-1
--endpoint https://glue.us-east-1.amazonaws.com
```

此命令不生成任何输出。

有关更多信息，请参阅《Glue 开发者指南》中的 AWS “在 Glue 数据目录中定义连接”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateConnection](#) 中的。

## create-database

以下代码示例演示了如何使用 create-database。

### AWS CLI

#### 创建数据库

以下 create-database 示例在 Glue 数据目录 AWS 中创建了一个数据库。

```
aws glue create-database \
  --database-input "{\"Name\":\"tempdb\"}" \
  --profile my_profile \
  --endpoint https://glue.us-east-1.amazonaws.com
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Glue 开发人员指南》中的 [在数据目录中定义数据库](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDatabase](#) 中的。

## create-job

以下代码示例演示了如何使用 create-job。

### AWS CLI

#### 创建用于转换数据的任务

以下 `create-job` 示例创建了一个运行存储在 S3 中的脚本的流式处理任务。

```
aws glue create-job \  
  --name my-testing-job \  
  --role AWSGlueServiceRoleDefault \  
  --command '{ \  
    "Name": "gluestreaming", \  
    "ScriptLocation": "s3://DOC-EXAMPLE-BUCKET/folder/" \  
  }' \  
  --region us-east-1 \  
  --output json \  
  --default-arguments '{ \  
    "--job-language":"scala", \  
    "--class":"GlueApp" \  
  }' \  
  --profile my-profile \  
  --endpoint https://glue.us-east-1.amazonaws.com
```

`test_script.scala` 的内容：

```
import com.amazonaws.services.glue.ChoiceOption  
import com.amazonaws.services.glue.GlueContext  
import com.amazonaws.services.glue.MappingSpec  
import com.amazonaws.services.glue.ResolveSpec  
import com.amazonaws.services.glue.errors.CallSite  
import com.amazonaws.services.glue.util.GlueArgParser  
import com.amazonaws.services.glue.util.Job  
import com.amazonaws.services.glue.util.JsonOptions  
import org.apache.spark.SparkContext  
import scala.collection.JavaConverters._  
  
object GlueApp {  
  def main(sysArgs: Array[String]) {  
    val spark: SparkContext = new SparkContext()  
    val glueContext: GlueContext = new GlueContext(spark)  
    // @params: [JOB_NAME]  
    val args = GlueArgParser.getResolvedOptions(sysArgs,  
Seq("JOB_NAME").toArray)  
    Job.init(args("JOB_NAME"), glueContext, args.asJava)  
    // @type: DataSource  
    // @args: [database = "tempdb", table_name = "s3-source", transformation_ctx  
= "datasource0"]  
    // @return: datasource0
```

```

        // @inputs: []
        val datasource0 = glueContext.getCatalogSource(database = "tempdb",
        tableName = "s3-source", redshiftTmpDir = "", transformationContext =
        "datasource0").getDynamicFrame()
        // @type: ApplyMapping
        // @args: [mapping = [("sensorid", "int", "sensorid", "int"),
        ("currenttemperature", "int", "currenttemperature", "int"), ("status", "string",
        "status", "string")], transformation_ctx = "applymapping1"]
        // @return: applymapping1
        // @inputs: [frame = datasource0]
        val applymapping1 = datasource0.applyMapping(mappings = Seq(("sensorid",
        "int", "sensorid", "int"), ("currenttemperature", "int", "currenttemperature",
        "int"), ("status", "string", "status", "string")), caseSensitive = false,
        transformationContext = "applymapping1")
        // @type: SelectFields
        // @args: [paths = ["sensorid", "currenttemperature", "status"],
        transformation_ctx = "selectfields2"]
        // @return: selectfields2
        // @inputs: [frame = applymapping1]
        val selectfields2 = applymapping1.selectFields(paths = Seq("sensorid",
        "currenttemperature", "status"), transformationContext = "selectfields2")
        // @type: ResolveChoice
        // @args: [choice = "MATCH_CATALOG", database = "tempdb", table_name = "my-
        s3-sink", transformation_ctx = "resolvechoice3"]
        // @return: resolvechoice3
        // @inputs: [frame = selectfields2]
        val resolvechoice3 = selectfields2.resolveChoice(choiceOption =
        Some(ChoiceOption("MATCH_CATALOG")), database = Some("tempdb"), tableName =
        Some("my-s3-sink"), transformationContext = "resolvechoice3")
        // @type: DataSink
        // @args: [database = "tempdb", table_name = "my-s3-sink",
        transformation_ctx = "datasink4"]
        // @return: datasink4
        // @inputs: [frame = resolvechoice3]
        val datasink4 = glueContext.getCatalogSink(database = "tempdb",
        tableName = "my-s3-sink", redshiftTmpDir = "", transformationContext =
        "datasink4").writeDynamicFrame(resolvechoice3)
        Job.commit()
    }
}

```

输出：

```
{
  "Name": "my-testing-job"
}
```

有关更多信息，请参阅《Glue 开发者指南》中的“[在 AWS Glue 中 AWS 创作作业](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateJob](#) 中的。

## create-table

以下代码示例演示了如何使用 create-table。

### AWS CLI

示例 1：为 Kinesis 数据流创建表

以下 create-table 示例在 Glue 数据目录中 AWS 创建了一个描述 Kinesis 数据流的表。

```
aws glue create-table \
  --database-name tempdb \
  --table-input '{"Name":"test-kinesis-input", "StorageDescriptor":{ \
    "Columns":[ \
      {"Name":"sensorid", "Type":"int"}, \
      {"Name":"currenttemperature", "Type":"int"}, \
      {"Name":"status", "Type":"string"} \
    ], \
    "Location":"my-testing-stream", \
    "Parameters":{ \
      "typeOfData":"kinesis","streamName":"my-testing-stream", \
      "kinesisUrl":"https://kinesis.us-east-1.amazonaws.com" \
    }, \
    "SerdeInfo":{ \
      "SerializationLibrary":"org.openx.data.jsonserde.JsonSerDe"} \
  }, \
  "Parameters":{ \
    "classification":"json"} \
}' \
  --profile my-profile \
  --endpoint https://glue.us-east-1.amazonaws.com
```

此命令不生成任何输出。

有关更多信息，请参阅《Glue 开发者指南》中的 AWS “在 [Glue 数据目录中定义表](#)”。

## 示例 2：为 Kafka 数据存储库创建表

以下create-table示例在 Glue 数据目录中 AWS 创建了一个描述 Kafka 数据存储的表。

```
aws glue create-table \  
  --database-name tempdb \  
  --table-input '{"Name":"test-kafka-input", "StorageDescriptor":{ \  
    "Columns":[ \  
      {"Name":"sensorid", "Type":"int"}, \  
      {"Name":"currenttemperature", "Type":"int"}, \  
      {"Name":"status", "Type":"string"} \  
    ], \  
    "Location":"glue-topic", \  
    "Parameters":{ \  
      "typeOfData":"kafka","topicName":"glue-topic", \  
      "connectionName":"my-kafka-connection"} \  
    }, \  
    "SerdeInfo":{ \  
      "SerializationLibrary":"org.apache.hadoop.hive.serde2.OpenCSVSerde"} \  
  } \  
  , \  
  "Parameters":{ \  
    "separatorChar":"," } \  
}' \  
  --profile my-profile \  
  --endpoint https://glue.us-east-1.amazonaws.com
```

此命令不生成任何输出。

有关更多信息，请参阅《Glue 开发者指南》中的 AWS “在 Glue 数据目录中定义表”。

## 示例 3：为 AWS S3 数据存储创建表

以下create-table示例在 Glue 数据目录 AWS 中创建了一个描述 AWS 简单存储服务 (AWS S3) 数据存储的表。

```
aws glue create-table \  
  --database-name tempdb \  
  --table-input '{"Name":"s3-output", "StorageDescriptor":{ \  
    "Columns":[ \  
      {"Name":"s1", "Type":"string"}, \  
      {"Name":"s2", "Type":"int"}, \  
      {"Name":"s3", "Type":"string"} \  
    ] \  
  }'
```

```
    ], \
    "Location": "s3://bucket-path/", \
    "SerdeInfo": { \
        "SerializationLibrary": "org.openx.data.jsonserde.JsonSerDe" \
    }, \
    "Parameters": { \
        "classification": "json" \
    } \
}' \
--profile my-profile \
--endpoint https://glue.us-east-1.amazonaws.com
```

此命令不生成任何输出。

有关更多信息，请参阅《Glue 开发者指南》中的 AWS “在 [Glue 数据目录中定义表](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateTable](#) 中的。

## delete-job

以下代码示例演示了如何使用 delete-job。

### AWS CLI

#### 删除任务

以下 delete-job 示例删除了不再需要的任务。

```
aws glue delete-job \
  --job-name my-testing-job
```

输出：

```
{
  "JobName": "my-testing-job"
}
```

有关更多信息，请参阅《Glue 开发者指南》中的 “在 [AWS Glue 控制台 AWS 上处理作业](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteJob](#) 中的。

## get-databases

以下代码示例演示了如何使用 get-databases。

## AWS CLI

在 Glue 数据目录中列出部分或全部 AWS 数据库的定义

以下 `get-databases` 示例返回有关数据目录中数据库的信息。

```
aws glue get-databases
```

输出：

```
{
  "DatabaseList": [
    {
      "Name": "default",
      "Description": "Default Hive database",
      "LocationUri": "file:/spark-warehouse",
      "CreateTime": 1602084052.0,
      "CreateTableDefaultPermissions": [
        {
          "Principal": {
            "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
          },
          "Permissions": [
            "ALL"
          ]
        }
      ],
      "CatalogId": "111122223333"
    },
    {
      "Name": "flights-db",
      "CreateTime": 1587072847.0,
      "CreateTableDefaultPermissions": [
        {
          "Principal": {
            "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
          },
          "Permissions": [
            "ALL"
          ]
        }
      ],
      "CatalogId": "111122223333"
    }
  ]
}
```

```
    },
    {
      "Name": "legislators",
      "CreateTime": 1601415625.0,
      "CreateTableDefaultPermissions": [
        {
          "Principal": {
            "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
          },
          "Permissions": [
            "ALL"
          ]
        }
      ],
      "CatalogId": "111122223333"
    },
    {
      "Name": "tempdb",
      "CreateTime": 1601498566.0,
      "CreateTableDefaultPermissions": [
        {
          "Principal": {
            "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
          },
          "Permissions": [
            "ALL"
          ]
        }
      ],
      "CatalogId": "111122223333"
    }
  ]
}
```

有关更多信息，请参阅《AWS Glue 开发人员指南》中的[在数据目录中定义数据库](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDatabases](#)中的。

## get-job-run

以下代码示例演示了如何使用 get-job-run。



## AWS CLI

### 获取有关任务运行的信息

以下 `get-job-run` 示例检索有关任务运行的信息。

```
aws glue get-job-run \  
  --job-name "Combine legislators data" \  
  --run-id jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e
```

输出：

```
{  
  "JobRun": {  
    "Id": "jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e",  
    "Attempt": 0,  
    "JobName": "Combine legislators data",  
    "StartedOn": 1602873931.255,  
    "LastModifiedOn": 1602874075.985,  
    "CompletedOn": 1602874075.985,  
    "JobRunState": "SUCCEEDED",  
    "Arguments": {  
      "--enable-continuous-cloudwatch-log": "true",  
      "--enable-metrics": "",  
      "--enable-spark-ui": "true",  
      "--job-bookmark-option": "job-bookmark-enable",  
      "--spark-event-logs-path": "s3://aws-glue-assets-111122223333-us-east-1/  
sparkHistoryLogs/"  
    },  
    "PredecessorRuns": [],  
    "AllocatedCapacity": 10,  
    "ExecutionTime": 117,  
    "Timeout": 2880,  
    "MaxCapacity": 10.0,  
    "WorkerType": "G.1X",  
    "NumberOfWorkers": 10,  
    "LogGroupName": "/aws-glue/jobs",  
    "GlueVersion": "2.0"  
  }  
}
```

有关更多信息，请参阅《AWS Glue 开发人员指南》中的[任务运行](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetJobRun](#) 中的。

## get-job-runs

以下代码示例演示了如何使用 `get-job-runs`。

### AWS CLI

获取有关任务的所有任务运行的信息

以下 `get-job-runs` 示例检索有关任务的任务运行的信息。

```
aws glue get-job-runs \  
  --job-name "my-testing-job"
```

输出：

```
{  
  "JobRuns": [  
    {  
      "Id":  
"jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e",  
      "Attempt": 0,  
      "JobName": "my-testing-job",  
      "StartedOn": 1602873931.255,  
      "LastModifiedOn": 1602874075.985,  
      "CompletedOn": 1602874075.985,  
      "JobRunState": "SUCCEEDED",  
      "Arguments": {  
        "--enable-continuous-cloudwatch-log": "true",  
        "--enable-metrics": "",  
        "--enable-spark-ui": "true",  
        "--job-bookmark-option": "job-bookmark-enable",  
        "--spark-event-logs-path": "s3://aws-glue-assets-111122223333-us-  
east-1/sparkHistoryLogs/"  
      },  
      "PredecessorRuns": [],  
      "AllocatedCapacity": 10,  
      "ExecutionTime": 117,  
      "Timeout": 2880,  
      "MaxCapacity": 10.0,  
      "WorkerType": "G.1X",  
      "NumberOfWorkers": 10,  
      "LogGroupName": "/aws-glue/jobs",  
      "GlueVersion": "2.0"  
    },  
  ],  
}
```

```

    {
      "Id":
"jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_2",
      "Attempt": 2,
      "PreviousRunId":
"jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_1",
      "JobName": "my-testing-job",
      "StartedOn": 1602811168.496,
      "LastModifiedOn": 1602811282.39,
      "CompletedOn": 1602811282.39,
      "JobRunState": "FAILED",
      "ErrorMessage": "An error occurred while calling
o122.pyWriteDynamicFrame.
          Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
AccessDenied;
          Request ID: 021AAB703DB20A2D;
          S3 Extended Request ID: teZk24Y09TkXzBvMPG502L5VJBhe9DJuWA9/
TXtuG0qfByajkfl/Tlqt5JBGdEGpigAqzdMDM/U=)",
      "PredecessorRuns": [],
      "AllocatedCapacity": 10,
      "ExecutionTime": 110,
      "Timeout": 2880,
      "MaxCapacity": 10.0,
      "WorkerType": "G.1X",
      "NumberOfWorkers": 10,
      "LogGroupName": "/aws-glue/jobs",
      "GlueVersion": "2.0"
    },
    {
      "Id":
"jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_1",
      "Attempt": 1,
      "PreviousRunId":
"jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f",
      "JobName": "my-testing-job",
      "StartedOn": 1602811020.518,
      "LastModifiedOn": 1602811138.364,
      "CompletedOn": 1602811138.364,
      "JobRunState": "FAILED",
      "ErrorMessage": "An error occurred while calling
o122.pyWriteDynamicFrame.
          Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
AccessDenied;
          Request ID: 2671D37856AE7ABB;

```

```
S3 Extended Request ID: RLJCJw20brV
+PpC6Gp0RahyF2fp9f1B5SSb2bTGPnUSPVizLXR11PN3QZ1db+v1o9qRVktNYbW8=)",
  "PredecessorRuns": [],
  "AllocatedCapacity": 10,
  "ExecutionTime": 113,
  "Timeout": 2880,
  "MaxCapacity": 10.0,
  "WorkerType": "G.1X",
  "NumberOfWorkers": 10,
  "LogGroupName": "/aws-glue/jobs",
  "GlueVersion": "2.0"
}
]
}
```

有关更多信息，请参阅《AWS Glue 开发人员指南》中的[任务运行](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetJobRuns](#) 中的。

## get-job

以下代码示例演示了如何使用 get-job。

### AWS CLI

检索有关任务的信息

以下 get-job 示例检索有关任务的信息。

```
aws glue get-job \
  --job-name my-testing-job
```

输出：

```
{
  "Job": {
    "Name": "my-testing-job",
    "Role": "Glue_DefaultRole",
    "CreatedOn": 1602805698.167,
    "LastModifiedOn": 1602805698.167,
    "ExecutionProperty": {
      "MaxConcurrentRuns": 1
    },
  },
}
```

```

    "Command": {
      "Name": "gluestreaming",
      "ScriptLocation": "s3://janetst-bucket-01/Scripts/test_script.scala",
      "PythonVersion": "2"
    },
    "DefaultArguments": {
      "--class": "GlueApp",
      "--job-language": "scala"
    },
    "MaxRetries": 0,
    "AllocatedCapacity": 10,
    "MaxCapacity": 10.0,
    "GlueVersion": "1.0"
  }
}

```

有关更多信息，请参阅《AWS Glue 开发人员指南》中的[任务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetJob](#) 中的。

## get-plan

以下代码示例演示了如何使用 get-plan。

### AWS CLI

获取生成的代码，用于将数据从源表映射到目标表

以下内容 get-plan 检索生成的代码，用于将列从数据源映射到数据目标。

```

aws glue get-plan --mapping '[ \
  { \
    "SourcePath": "sensorid", \
    "SourceTable": "anything", \
    "SourceType": "int", \
    "TargetPath": "sensorid", \
    "TargetTable": "anything", \
    "TargetType": "int" \
  }, \
  { \
    "SourcePath": "currenttemperature", \
    "SourceTable": "anything", \
    "SourceType": "int", \

```

```

        "TargetPath":"currenttemperature", \
        "TargetTable":"anything", \
        "TargetType":"int" \
    }, \
    { \
        "SourcePath":"status", \
        "SourceTable":"anything", \
        "SourceType":"string", \
        "TargetPath":"status", \
        "TargetTable":"anything", \
        "TargetType":"string" \
    ]]' \
--source '{ \
    "DatabaseName":"tempdb", \
    "TableName":"s3-source" \
}' \
--sinks '[' \
    { \
        "DatabaseName":"tempdb", \
        "TableName":"my-s3-sink" \
    }]' \
--language "scala"
--endpoint https://glue.us-east-1.amazonaws.com
--output "text"

```

输出：

```

import com.amazonaws.services.glue.ChoiceOption
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.ResolveSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    // @params: [JOB_NAME]

```

```

    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
    // @type: DataSource
    // @args: [database = "tempdb", table_name = "s3-source", transformation_ctx =
"datasource0"]
    // @return: datasource0
    // @inputs: []
    val datasource0 = glueContext.getCatalogSource(database = "tempdb",
tableName = "s3-source", redshiftTmpDir = "", transformationContext =
"datasource0").getDynamicFrame()
    // @type: ApplyMapping
    // @args: [mapping = [("sensorid", "int", "sensorid", "int"),
("currenttemperature", "int", "currenttemperature", "int"), ("status", "string",
"status", "string")], transformation_ctx = "applymapping1"]
    // @return: applymapping1
    // @inputs: [frame = datasource0]
    val applymapping1 = datasource0.applyMapping(mappings = Seq(("sensorid",
"int", "sensorid", "int"), ("currenttemperature", "int", "currenttemperature",
"int"), ("status", "string", "status", "string")), caseSensitive = false,
transformationContext = "applymapping1")
    // @type: SelectFields
    // @args: [paths = ["sensorid", "currenttemperature", "status"],
transformation_ctx = "selectfields2"]
    // @return: selectfields2
    // @inputs: [frame = applymapping1]
    val selectfields2 = applymapping1.selectFields(paths = Seq("sensorid",
"currenttemperature", "status"), transformationContext = "selectfields2")
    // @type: ResolveChoice
    // @args: [choice = "MATCH_CATALOG", database = "tempdb", table_name = "my-s3-
sink", transformation_ctx = "resolvechoice3"]
    // @return: resolvechoice3
    // @inputs: [frame = selectfields2]
    val resolvechoice3 = selectfields2.resolveChoice(choiceOption =
Some(ChoiceOption("MATCH_CATALOG")), database = Some("tempdb"), tableName =
Some("my-s3-sink"), transformationContext = "resolvechoice3")
    // @type: DataSink
    // @args: [database = "tempdb", table_name = "my-s3-sink", transformation_ctx =
"datasink4"]
    // @return: datasink4
    // @inputs: [frame = resolvechoice3]
    val datasink4 = glueContext.getCatalogSink(database = "tempdb",
tableName = "my-s3-sink", redshiftTmpDir = "", transformationContext =
"datasink4").writeDynamicFrame(resolvechoice3)
    Job.commit()

```

```
}  
}
```

有关更多信息，请参阅 Glue 开发者指南中的 [AWS 在 GI AWS u e 中编辑脚本](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetPlan](#) 中的。

## get-tables

以下代码示例演示了如何使用 get-tables。

### AWS CLI

列出指定数据库中的部分或全部表的定义

以下 get-tables 示例返回有关指定数据库中表的信息。

```
aws glue get-tables --database-name 'tempdb'
```

输出：

```
{  
  "TableList": [  
    {  
      "Name": "my-s3-sink",  
      "DatabaseName": "tempdb",  
      "CreateTime": 1602730539.0,  
      "UpdateTime": 1602730539.0,  
      "Retention": 0,  
      "StorageDescriptor": {  
        "Columns": [  
          {  
            "Name": "sensorid",  
            "Type": "int"  
          },  
          {  
            "Name": "currenttemperature",  
            "Type": "int"  
          },  
          {  
            "Name": "status",  
            "Type": "string"  
          }  
        ]  
      }  
    }  
  ]  
}
```



```
    ],
    "Location": "s3://janetst-bucket-01/test-s3-output/",
    "Compressed": false,
    "NumberOfBuckets": 0,
    "SerdeInfo": {
      "SerializationLibrary": "org.openx.data.jsonserde.JsonSerDe"
    },
    "SortColumns": [],
    "StoredAsSubDirectories": false
  },
  "Parameters": {
    "classification": "json"
  },
  "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
  "IsRegisteredWithLakeFormation": false,
  "CatalogId": "007436865787"
},
{
  "Name": "s3-source",
  "DatabaseName": "tempdb",
  "CreateTime": 1602730658.0,
  "UpdateTime": 1602730658.0,
  "Retention": 0,
  "StorageDescriptor": {
    "Columns": [
      {
        "Name": "sensorid",
        "Type": "int"
      },
      {
        "Name": "currenttemperature",
        "Type": "int"
      },
      {
        "Name": "status",
        "Type": "string"
      }
    ]
  },
  "Location": "s3://janetst-bucket-01/",
  "Compressed": false,
  "NumberOfBuckets": 0,
  "SortColumns": [],
  "StoredAsSubDirectories": false
},
```

```
"Parameters": {
  "classification": "json"
},
"CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
"IsRegisteredWithLakeFormation": false,
"CatalogId": "007436865787"
},
{
  "Name": "test-kinesis-input",
  "DatabaseName": "tempdb",
  "CreateTime": 1601507001.0,
  "UpdateTime": 1601507001.0,
  "Retention": 0,
  "StorageDescriptor": {
    "Columns": [
      {
        "Name": "sensorid",
        "Type": "int"
      },
      {
        "Name": "currenttemperature",
        "Type": "int"
      },
      {
        "Name": "status",
        "Type": "string"
      }
    ],
    "Location": "my-testing-stream",
    "Compressed": false,
    "NumberOfBuckets": 0,
    "SerdeInfo": {
      "SerializationLibrary": "org.openx.data.jsonserde.JsonSerDe"
    },
    "SortColumns": [],
    "Parameters": {
      "kinesisUrl": "https://kinesis.us-east-1.amazonaws.com",
      "streamName": "my-testing-stream",
      "typeOfData": "kinesis"
    },
    "StoredAsSubDirectories": false
  },
  "Parameters": {
    "classification": "json"
  }
}
```

```
    },  
    "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",  
    "IsRegisteredWithLakeFormation": false,  
    "CatalogId": "007436865787"  
  }  
]  
}
```

有关更多信息，请参阅《Glue 开发者指南》中的 AWS “在 Glue 数据目录中定义表”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetTables](#) 中的。

## start-crawler

以下代码示例演示了如何使用 start-crawler。

### AWS CLI

启动爬网程序

以下 start-crawler 示例启动了一个爬网程序。

```
aws glue start-crawler --name my-crawler
```

输出：

```
None
```

有关更多信息，请参阅《AWS Glue 开发人员指南》中的 [定义爬网程序](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartCrawler](#) 中的。

## start-job-run

以下代码示例演示了如何使用 start-job-run。

### AWS CLI

开始运行任务

以下 start-job-run 示例启动了一个任务。

```
aws glue start-job-run \  
  --job-name my-job
```

输出：

```
{  
  "JobRunId":  
  "jr_22208b1f44eb5376a60569d4b21dd20fcb8621e1a366b4e7b2494af764b82ded"  
}
```

有关更多信息，请参阅《AWS Glue 开发人员指南》中的[编写任务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[StartJobRun](#)中的。

## GuardDuty 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 GuardDuty。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **accept-invitation**

以下代码示例演示了如何使用 `accept-invitation`。

AWS CLI

接受邀请成为当前地区的 GuardDuty 成员账户

以下 `accept-invitation` 示例说明如何接受邀请，成为当前地区的 GuardDuty 成员账户。

```
aws guardduty accept-invitation \  
  --detector-id 12abc34d567e8fa901bc2d34eexample \  
  --master-id 123456789111 \  
  --invitation-id d6b94fb03a66ff665f7db8764example
```

此命令不生成任何输出。

有关更多信息，请参阅《GuardDuty 用户指南》中的[通过邀请管理 GuardDuty 账户](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AcceptInvitation](#) 中的。

## archive-findings

以下代码示例演示了如何使用 `archive-findings`。

### AWS CLI

存档当前区域的调查结果

此示例说明如何存档当前区域的调查结果。

```
aws guardduty archive-findings \  
  --detector-id 12abc34d567e8fa901bc2d34eexample \  
  --finding-ids d6b94fb03a66ff665f7db8764example 3eb970e0de00c16ec14e6910fexample
```

此命令不生成任何输出。

有关更多信息，请参阅《GuardDuty 用户指南》中的[通过邀请管理 GuardDuty 账户](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ArchiveFindings](#) 中的。

## create-detector

以下代码示例演示了如何使用 `create-detector`。

### AWS CLI

GuardDuty 在当前区域启用

此示例说明如何在当前区域创建新的探测器并将其启用 GuardDuty。：

```
aws guardduty create-detector \  
  --enable
```

输出：

```
{  
  "DetectorId": "b6b992d6d2f48e64bc59180bfexample"  
}
```

有关更多信息，请参阅GuardDuty 用户指南 GuardDuty中的[启用 Amazon](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDetector](#)中的。

## create-filter

以下代码示例演示了如何使用 create-filter。

### AWS CLI

为当前区域创建新过滤器

此示例创建了一个过滤器，该过滤器与所有 portscan 结果相匹配，例如根据特定图像创建的结果。：

```
aws guardduty create-filter \  
  --detector-id b6b992d6d2f48e64bc59180bfexample \  
  --action ARCHIVE \  
  --name myFilter \  
  --finding-criteria '{"Criterion": {"type": {"Eq": ["Recon:EC2/  
Portscan"]},"resource.instanceDetails.imageId": {"Eq": ["ami-0a7a207083example"]}}}'
```

输出：

```
{  
  "Name": "myFilter"  
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[筛选结果](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFilter](#)中的。

## create-ip-set

以下代码示例演示了如何使用 `create-ip-set`。

### AWS CLI

#### 创建可信 IP 集

以下 `create-ip-set` 示例在当前区域中创建并激活可信 IP 集。

```
aws guardduty create-ip-set \  
  --detector-id 12abc34d567e8fa901bc2d34eexample \  
  --name new-ip-set \  
  --format TXT \  
  --location s3://AWSDOC-EXAMPLE-BUCKET/customtrustlist.csv \  
  --activate
```

输出：

```
{  
  "IpSetId": "d4b94fc952d6912b8f3060768example"  
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的“[使用可信 IP 列表和威胁列表](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreatelPSet](#) 中的。

## create-members

以下代码示例演示了如何使用 `create-members`。

### AWS CLI

将新成员与您在当前区域 GuardDuty 的主账户关联。

此示例说明如何关联成员账户，使其由当前账户作为 GuardDuty 主账户进行管理。

```
aws guardduty create-members \  
  --detector-id b6b992d6d2f48e64bc59180bfexample \  
  --account-details AccountId=111122223333,Email=first+member@example.com \  
  AccountId=111111111111 ,Email=another+member@example.com
```

输出：

```
{
  "UnprocessedAccounts": []
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[管理多个账户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateMembers](#)中的。

## create-publishing-destination

以下代码示例演示了如何使用 create-publishing-destination。

### AWS CLI

创建要将当前区域的 GuardDuty 结果导出到的发布目的地。

此示例说明如何为 GuardDuty 调查结果创建发布目标。

```
aws guardduty create-publishing-destination \
  --detector-id b6b992d6d2f48e64bc59180bfexample \
  --destination-type S3 \
  --destination-properties
  DestinationArn=arn:aws:s3:::yourbucket,KmsKeyArn=arn:aws:kms:us-
  west-1:111122223333:key/84cee9c5-dea1-401a-ab6d-e1de7example
```

输出：

```
{
  "DestinationId": "46b99823849e1bbc242dfbe3cexample"
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[导出调查结果](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePublishingDestination](#)中的。

## create-sample-findings

以下代码示例演示了如何使用 create-sample-findings。

### AWS CLI

在当前区域创建样本 GuardDuty 调查结果。



此示例说明如何创建所提供类型的样本查找结果。

```
aws guardduty create-sample-findings \  
  --detector-id b6b992d6d2f48e64bc59180bfexample \  
  --finding-types UnauthorizedAccess:EC2/TorClient UnauthorizedAccess:EC2/TorRelay
```

此命令不生成任何输出。

有关更多信息，请参阅《GuardDuty 用户指南》中的[调查结果示例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSampleFindings](#)中的。

## create-threat-intel-set

以下代码示例演示了如何使用 create-threat-intel-set。

### AWS CLI

在当前区域创建新的威胁情报。

此示例说明如何上传设置为的威胁情报 GuardDuty 并立即将其激活。

```
aws guardduty create-threat-intel-set \  
  --detector-id b6b992d6d2f48e64bc59180bfexample \  
  --name myThreatSet \  
  --format TXT \  
  --location s3://EXAMPLEBUCKET/threatlist.csv \  
  --activate
```

输出：

```
{  
  "ThreatIntelSetId": "20b9a4691aeb33506b808878cexample"  
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[可信 IP 和威胁列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateThreatIntelSet](#)中的。

## decline-invitations

以下代码示例演示了如何使用 decline-invitations。

## AWS CLI

拒绝邀请当前区域的其他账户管理 Guardduty。

此示例说明如何拒绝会员邀请。

```
aws guardduty decline-invitations \  
  --account-ids 111122223333
```

输出：

```
{  
  "UnprocessedAccounts": []  
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[通过邀请管理 GuardDuty 账户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeclineInvitations](#)中的。

## delete-detector

以下代码示例演示了如何使用 delete-detector。

### AWS CLI

删除当前区域中的探测器并将其禁用 GuardDuty。

此示例说明如何删除探测器，如果成功，则将在与该探测器关联的区域 GuardDuty 中禁用。

```
aws guardduty delete-detector \  
  --detector-id b6b992d6d2f48e64bc59180bfexample
```

此命令不生成任何输出。

有关更多信息，请参阅《GuardDuty 用户指南》GuardDuty中的[暂停或禁用](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDetector](#)中的。

## delete-filter

以下代码示例演示了如何使用 delete-filter。

## AWS CLI

删除当前区域中的现有过滤器

此示例说明如何创建删除过滤器。

```
aws guardduty delete-filter \  
  --detector-id b6b992d6d2f48e64bc59180bfexample \  
  --filter-name byebyeFilter
```

此命令不生成任何输出。

有关更多信息，请参阅《GuardDuty 用户指南》中的[筛选结果](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFilter](#)中的。

## disable-organization-admin-account

以下代码示例演示了如何使用 disable-organization-admin-account。

## AWS CLI

删除组织 GuardDuty 内委托管理员的账户

此示例说明如何移除委派管理员的账户 GuardDuty。

```
aws guardduty disable-organization-admin-account \  
  --admin-account-id 111122223333
```

此命令不生成任何输出。

有关更多信息，请参阅《GuardDuty 用户指南》中的[管理 AWS 组织账户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableOrganizationAdminAccount](#)中的。

## disassociate-from-master-account

以下代码示例演示了如何使用 disassociate-from-master-account。

## AWS CLI

取消与当前地区当前主账户的关联

以下 `disassociate-from-master-account` 示例取消您的账户与当前区域中当前 GuardDuty 主账户的关联。AWS

```
aws guardduty disassociate-from-master-account \  
  --detector-id d4b040365221be2b54a6264dcexample
```

此命令不生成任何输出。

有关更多信息，请参阅《GuardDuty 用户指南》中的[“了解 GuardDuty 主账户和成员账户之间的关系”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateFromMasterAccount](#) 中的。

## get-detector

以下代码示例演示了如何使用 `get-detector`。

### AWS CLI

检索特定探测器的详细信息

以下 `get-detector` 示例显示了指定检测器的配置详细信息。

```
aws guardduty get-detector \  
  --detector-id 12abc34d567e8fa901bc2d34eexample
```

输出：

```
{  
  "Status": "ENABLED",  
  "ServiceRole": "arn:aws:iam::111122223333:role/aws-service-role/  
guardduty.amazonaws.com/AWSServiceRoleForAmazonGuardDuty",  
  "Tags": {},  
  "FindingPublishingFrequency": "SIX_HOURS",  
  "UpdatedAt": "2018-11-07T03:24:22.938Z",  
  "CreatedAt": "2017-12-22T22:51:31.940Z"  
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[概念和术语](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDetector](#) 中的。

## get-findings

以下代码示例演示了如何使用 `get-findings`。

### AWS CLI

示例 1：检索特定发现的详细信息

以下 `get-findings` 示例检索指定查找结果的完整 JSON 查找结果详细信息。

```
aws guardduty get-findings \  
  --detector-id 12abc34d567e8fa901bc2d34eexample \  
  --finding-id 1ab92989eaf0e742df4a014d5example
```

输出：

```
{  
  "Findings": [  
    {  
      "Resource": {  
        "ResourceType": "AccessKey",  
        "AccessKeyDetails": {  
          "UserName": "testuser",  
          "UserType": "IAMUser",  
          "PrincipalId": "AIDACKCEVSQ6C2EXAMPLE",  
          "AccessKeyId": "ASIASZ4SI7REEEXAMPLE"  
        }  
      },  
      "Description": "APIs commonly used to discover the users, groups,  
policies and permissions in an account, was invoked by IAM principal testuser under  
unusual circumstances. Such activity is not typically seen from this principal.",  
      "Service": {  
        "Count": 5,  
        "Archived": false,  
        "ServiceName": "guardduty",  
        "EventFirstSeen": "2020-05-26T22:02:24Z",  
        "ResourceRole": "TARGET",  
        "EventLastSeen": "2020-05-26T22:33:55Z",  
        "DetectorId": "d4b040365221be2b54a6264dcexample",  
        "Action": {  
          "ActionType": "AWS_API_CALL",  
          "AwsApiCallAction": {  
            "RemoteIpDetails": {
```

```

        "GeoLocation": {
            "Lat": 51.5164,
            "Lon": -0.093
        },
        "City": {
            "CityName": "London"
        },
        "IpAddressV4": "52.94.36.7",
        "Organization": {
            "Org": "Amazon.com",
            "Isp": "Amazon.com",
            "Asn": "16509",
            "AsnOrg": "AMAZON-02"
        },
        "Country": {
            "CountryName": "United Kingdom"
        }
    },
    "Api": "ListPolicyVersions",
    "ServiceName": "iam.amazonaws.com",
    "CallerType": "Remote IP"
}
}
},
"Title": "Unusual user permission reconnaissance activity by testuser.",
"Type": "Recon:IAMUser/UserPermissions",
"Region": "us-east-1",
"Partition": "aws",
"Arn": "arn:aws:guardduty:us-east-1:111122223333:detector/
d4b040365221be2b54a6264dcexample/finding/1ab92989eaf0e742df4a014d5example",
"UpdatedAt": "2020-05-26T22:55:21.703Z",
"SchemaVersion": "2.0",
"Severity": 5,
"Id": "1ab92989eaf0e742df4a014d5example",
"CreatedAt": "2020-05-26T22:21:48.385Z",
"AccountId": "111122223333"
}
]
}

```

有关更多信息，请参阅《GuardDuty 用户指南》中的[调查结果](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFindings](#)中的。

## get-ip-set

以下代码示例演示了如何使用 `get-ip-set`。

### AWS CLI

要列出，请获取有关指定可信 IP 集的详细信息

以下`get-ip-set`示例显示了指定可信 IP 集的状态和详细信息。

```
aws guardduty get-ip-set \  
  --detector-id 12abc34d567e8fa901bc2d34eexample \  
  --ip-set-id d4b94fc952d6912b8f3060768example
```

输出：

```
{  
  "Status": "ACTIVE",  
  "Location": "s3://AWSDOC-EXAMPLE-BUCKET.s3-us-west-2.amazonaws.com/  
customlist.csv",  
  "Tags": {},  
  "Format": "TXT",  
  "Name": "test-ip-set"  
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[“使用可信 IP 列表和威胁列表”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetIpSet](#)中的。

## get-master-account

以下代码示例演示了如何使用 `get-master-account`。

### AWS CLI

在当前地区检索有关您的主账户的详细信息

以下`get-master-account`示例显示当前区域中与您的探测器关联的主账户的状态和详细信息。

```
aws guardduty get-master-account \  
  --detector-id 12abc34d567e8fa901bc2d34eexample
```

输出：

```
{
  "Master": {
    "InvitationId": "04b94d9704854a73f94e061e8example",
    "InvitedAt": "2020-06-09T22:23:04.970Z",
    "RelationshipStatus": "Enabled",
    "AccountId": "123456789111"
  }
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[“了解 GuardDuty 主账户和成员账户之间的关系”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMasterAccount](#)中的。

## list-detectors

以下代码示例演示了如何使用 list-detectors。

### AWS CLI

列出当前区域中的可用探测器

以下list-detectors示例列出了您当前 AWS 区域中可用的探测器。

```
aws guardduty list-detectors
```

输出：

```
{
  "DetectorIds": [
    "12abc34d567e8fa901bc2d34eexample"
  ]
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[概念和术语](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDetectors](#)中的。

## list-findings

以下代码示例演示了如何使用 list-findings。



## AWS CLI

示例 1：列出当前区域的所有调查结果

以下list-findings示例显示了当前区域的所有 FindingID 列表，按严重性从高到低排序。

```
aws guardduty list-findings \  
  --detector-id 12abc34d567e8fa901bc2d34eexample \  
  --sort-criteria '{"AttributeName": "severity", "OrderBy": "DESC"}'
```

输出：

```
{  
  "FindingIds": [  
    "04b8ab50fd29c64fc771b232dexample",  
    "5ab8ab50fd21373735c826d3aexample",  
    "90b93de7aba69107f05bbe60bexample",  
    ...  
  ]  
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[调查结果](#)。

示例 2：列出与特定查找条件相匹配的当前区域的查找结果

以下list-findings示例显示了与指定查找类型匹配的所有 FindingID 的列表。

```
aws guardduty list-findings \  
  --detector-id 12abc34d567e8fa901bc2d34eexample \  
  --finding-criteria '{"Criterion":{"type": {"Eq":["UnauthorizedAccess:EC2/  
SSHBruteForce"]}}}'
```

输出：

```
{  
  "FindingIds": [  
    "90b93de7aba69107f05bbe60bexample",  
    "6eb9430d7023d30774d6f05e3example",  
    "2eb91a2d060ac9a21963a5848example",  
    "44b8ab50fd2b0039a9e48f570example",  
    "9eb8ab4cd2b7e5b66ba4f5e96example",  
  ]  
}
```

```
    "e0b8ab3a38e9b0312cc390ceeexample"  
  ]  
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[调查结果](#)。

示例 3：列出与 JSON 文件中定义的一组特定查找条件相匹配的当前区域的查找结果

以下list-findings示例显示了所有未存档的 FindingID 的列表，这些用户涉及在 JSON 文件中指定的名为“testuser”的 IAM 用户。

```
aws guardduty list-findings \  
  --detector-id 12abc34d567e8fa901bc2d34eexample \  
  --finding-criteria file://myfile.json
```

myfile.json 的内容：

```
{"Criterion": {  
  "resource.accessKeyDetails.userName": {  
    "Eq": [  
      "testuser"  
    ]  
  },  
  "service.archived": {  
    "Eq": [  
      "false"  
    ]  
  }  
}}
```

输出：

```
{  
  "FindingIds": [  
    "1ab92989eaf0e742df4a014d5example"  
  ]  
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[调查结果](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFindings](#)中的。

## list-invitations

以下代码示例演示了如何使用 `list-invitations`。

### AWS CLI

列出有关您成为当前地区成员账户的邀请的详细信息

以下 `list-invitations` 示例列出了您成为当前地区 GuardDuty 成员账户的邀请的详细信息和状态。

```
aws guardduty list-invitations
```

输出：

```
{
  "Invitations": [
    {
      "InvitationId": "d6b94fb03a66ff665f7db8764example",
      "InvitedAt": "2020-06-10T17:56:38.221Z",
      "RelationshipStatus": "Invited",
      "AccountId": "123456789111"
    }
  ]
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[通过邀请管理 GuardDuty 账户](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListInvitations](#) 中的。

## list-ip-sets

以下代码示例演示了如何使用 `list-ip-sets`。

### AWS CLI

列出当前区域中的可信 IP 集

以下 `list-ip-sets` 示例列出了您当前 AWS 区域中的可信 IP 集。

```
aws guardduty list-ip-sets \
```

```
--detector-id 12abc34d567e8fa901bc2d34eexample
```

输出：

```
{
  "IpSetIds": [
    "d4b94fc952d6912b8f3060768example"
  ]
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的“[使用可信 IP 列表和威胁列表](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListIpSets](#) 中的。

## list-members

以下代码示例演示了如何使用 list-members。

### AWS CLI

列出当前区域的所有成员

以下 list-members 示例列出了当前区域的所有成员账户及其详细信息。

```
aws guardduty list-members \
  --detector-id 12abc34d567e8fa901bc2d34eexample
```

输出：

```
{
  "Members": [
    {
      "RelationshipStatus": "Enabled",
      "InvitedAt": "2020-06-09T22:49:00.910Z",
      "MasterId": "123456789111",
      "DetectorId": "7ab8b2f61b256c87f793f6a86example",
      "UpdatedAt": "2020-06-09T23:08:22.512Z",
      "Email": "your+member@example.com",
      "AccountId": "123456789222"
    }
  ]
}
```

有关更多信息，请参阅《GuardDuty 用户指南》中的[“了解 GuardDuty 主账户和成员账户之间的关系”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListMembers](#)中的。

## update-ip-set

以下代码示例演示了如何使用 update-ip-set。

### AWS CLI

#### 更新可信 IP 集

以下update-ip-set示例说明如何更新可信 IP 集的详细信息。

```
aws guardduty update-ip-set \  
  --detector-id 12abc34d567e8fa901bc2d34eexample \  
  --ip-set-id d4b94fc952d6912b8f3060768example \  
  --location https://AWSDOC-EXAMPLE-BUCKET.s3-us-west-2.amazonaws.com/  
  customtrustlist2.csv
```

此命令不生成任何输出。

有关更多信息，请参阅《GuardDuty 用户指南》中的[“使用可信 IP 列表和威胁列表”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateIpSet](#)中的。

## AWS Health 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Health。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### describe-affected-entities

以下代码示例演示了如何使用 describe-affected-entities。

#### AWS CLI

列出受指定 AWS Health 事件影响的实体

以下 describe-affected-entities 示例列出了受指定 AWS Health 事件影响的实体。此事件是该 AWS 账户的账单通知。

```
aws health describe-affected-entities \
  --filter "eventArns=arn:aws:health:global::event/BILLING/
AWS_BILLING_NOTIFICATION/AWS_BILLING_NOTIFICATION_6ce1d874-e995-40e2-99cd-
EXAMPLE11145" \
  --region us-east-1
```

输出：

```
{
  "entities": [
    {
      "entityArn": "arn:aws:health:global:123456789012:entity/
EXAMPLEimSMoULmWHpb",
      "eventArn": "arn:aws:health:global::event/BILLING/
AWS_BILLING_NOTIFICATION/AWS_BILLING_NOTIFICATION_6ce1d874-e995-40e2-99cd-
EXAMPLE11145",
      "entityValue": "AWS_ACCOUNT",
      "awsAccountId": "123456789012",
      "lastUpdatedTime": 1588356454.08
    }
  ]
}
```

有关更多信息，请参阅 Health AWS h 用户指南中的[事件日志](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAffectedEntities](#) 中的。

### describe-event-details

以下代码示例演示了如何使用 describe-event-details。

## AWS CLI

列出有关 AWS Health 事件的信息

以下describe-event-details示例列出了有关指定 AWS Health 事件的信息。

```
aws health describe-event-details \  
  --event-arns "arn:aws:health:us-east-1::event/EC2/AWS_EC2_OPERATIONAL_ISSUE/  
AWS_EC2_OPERATIONAL_ISSUE_VKTXI_EXAMPLE111" \  
  --region us-east-1
```

输出：

```
{  
  "successfulSet": [  
    {  
      "event": {  
        "arn": "arn:aws:health:us-east-1::event/EC2/  
AWS_EC2_OPERATIONAL_ISSUE/AWS_EC2_OPERATIONAL_ISSUE_VKTXI_EXAMPLE111",  
        "service": "EC2",  
        "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",  
        "eventTypeCategory": "issue",  
        "region": "us-east-1",  
        "startTime": 1587462325.096,  
        "endTime": 1587464204.774,  
        "lastUpdatedTime": 1587464204.865,  
        "statusCode": "closed"  
      },  
      "eventDescription": {  
        "latestDescription": "[RESOLVED] Increased API Error Rates and  
Latencies\n\n[02:45 AM PDT] We are investigating increased API error rates and  
latencies in the US-EAST-1 Region.\n\n[03:16 AM PDT] Between 2:10 AM and 2:59 AM  
PDT we experienced increased API error rates and latencies in the US-EAST-1 Region.  
The issue has been resolved and the service is operating normally."  
      }  
    }  
  ],  
  "failedSet": []  
}
```

有关更多信息，请参阅 Health AWS h 用户指南中的[事件详情窗格](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEventDetails](#)中的。

## describe-events

以下代码示例演示了如何使用 describe-events。

### AWS CLI

示例 1：列出 Health AWS h 事件

以下 describe-events 示例列出了最近的 Health AWS 事件。

```
aws health describe-events \
  --region us-east-1
```

输出：

```
{
  "events": [
    {
      "arn": "arn:aws:health:us-west-1::event/ECS/AWS_ECS_OPERATIONAL_ISSUE/AWS_ECS_OPERATIONAL_ISSUE_KWQPY_EXAMPLE111",
      "service": "ECS",
      "eventTypeCode": "AWS_ECS_OPERATIONAL_ISSUE",
      "eventTypeCategory": "issue",
      "region": "us-west-1",
      "startTime": 1589077890.53,
      "endTime": 1589086345.597,
      "lastUpdatedTime": 1589086345.905,
      "statusCode": "closed",
      "eventScopeCode": "PUBLIC"
    },
    {
      "arn": "arn:aws:health:global::event/BILLING/AWS_BILLING_NOTIFICATION/AWS_BILLING_NOTIFICATION_6ce1d874-e995-40e2-99cd-EXAMPLE1118b",
      "service": "BILLING",
      "eventTypeCode": "AWS_BILLING_NOTIFICATION",
      "eventTypeCategory": "accountNotification",
      "region": "global",
      "startTime": 1588356000.0,
      "lastUpdatedTime": 1588356524.358,
      "statusCode": "open",
      "eventScopeCode": "ACCOUNT_SPECIFIC"
    },
    {
```



```
    "arn": "arn:aws:health:us-west-2::event/
    CLOUDFORMATION/AWS_CLOUDFORMATION_OPERATIONAL_ISSUE/
    AWS_CLOUDFORMATION_OPERATIONAL_ISSUE_OHTWY_EXAMPLE111",
    "service": "CLOUDFORMATION",
    "eventTypeCode": "AWS_CLOUDFORMATION_OPERATIONAL_ISSUE",
    "eventTypeCategory": "issue",
    "region": "us-west-2",
    "startTime": 1588279630.761,
    "endTime": 1588284650.0,
    "lastUpdatedTime": 1588284691.941,
    "statusCode": "closed",
    "eventScopeCode": "PUBLIC"
  },
  {
    "arn": "arn:aws:health:ap-northeast-1::event/LAMBDA/
    AWS_LAMBDA_OPERATIONAL_ISSUE/AWS_LAMBDA_OPERATIONAL_ISSUE_JZDND_EXAMPLE111",
    "service": "LAMBDA",
    "eventTypeCode": "AWS_LAMBDA_OPERATIONAL_ISSUE",
    "eventTypeCategory": "issue",
    "region": "ap-northeast-1",
    "startTime": 1587379534.08,
    "endTime": 1587391771.0,
    "lastUpdatedTime": 1587395689.316,
    "statusCode": "closed",
    "eventScopeCode": "PUBLIC"
  },
  {
    "arn": "arn:aws:health:us-east-1::event/EC2/AWS_EC2_OPERATIONAL_ISSUE/
    AWS_EC2_OPERATIONAL_ISSUE_COBXJ_EXAMPLE111",
    "service": "EC2",
    "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
    "eventTypeCategory": "issue",
    "region": "us-east-1",
    "startTime": 1586473044.284,
    "endTime": 1586479706.091,
    "lastUpdatedTime": 1586479706.153,
    "statusCode": "closed",
    "eventScopeCode": "PUBLIC"
  },
  {
    "arn": "arn:aws:health:global::event/SECURITY/AWS_SECURITY_NOTIFICATION/
    AWS_SECURITY_NOTIFICATION_42007387-8129-42da-8c88-EXAMPLE11139",
    "service": "SECURITY",
    "eventTypeCode": "AWS_SECURITY_NOTIFICATION",
```

```
    "eventTypeCategory": "accountNotification",
    "region": "global",
    "startTime": 1585674000.0,
    "lastUpdatedTime": 1585674004.132,
    "statusCode": "open",
    "eventScopeCode": "PUBLIC"
  },
  {
    "arn": "arn:aws:health:global::event/CLOUDFRONT/
AWS_CLOUDFRONT_OPERATIONAL_ISSUE/AWS_CLOUDFRONT_OPERATIONAL_ISSUE_FRQXG_EXAMPLE111",
    "service": "CLOUDFRONT",
    "eventTypeCode": "AWS_CLOUDFRONT_OPERATIONAL_ISSUE",
    "eventTypeCategory": "issue",
    "region": "global",
    "startTime": 1585610898.589,
    "endTime": 1585617671.0,
    "lastUpdatedTime": 1585620638.869,
    "statusCode": "closed",
    "eventScopeCode": "PUBLIC"
  },
  {
    "arn": "arn:aws:health:us-east-1::event/SES/AWS_SES_OPERATIONAL_ISSUE/
AWS_SES_OPERATIONAL_ISSUE_URNDF_EXAMPLE111",
    "service": "SES",
    "eventTypeCode": "AWS_SES_OPERATIONAL_ISSUE",
    "eventTypeCategory": "issue",
    "region": "us-east-1",
    "startTime": 1585342008.46,
    "endTime": 1585344017.0,
    "lastUpdatedTime": 1585344355.989,
    "statusCode": "closed",
    "eventScopeCode": "PUBLIC"
  },
  {
    "arn": "arn:aws:health:global::event/IAM/
AWS_IAM_OPERATIONAL_NOTIFICATION/
AWS_IAM_OPERATIONAL_NOTIFICATION_b6771c34-6ecd-4aea-9d3e-EXAMPLE1117e",
    "service": "IAM",
    "eventTypeCode": "AWS_IAM_OPERATIONAL_NOTIFICATION",
    "eventTypeCategory": "accountNotification",
    "region": "global",
    "startTime": 1584978300.0,
    "lastUpdatedTime": 1584978553.572,
    "statusCode": "open",
```

```

        "eventScopeCode": "ACCOUNT_SPECIFIC"
    },
    {
        "arn": "arn:aws:health:ap-southeast-2::event/EC2/
AWS_EC2_OPERATIONAL_ISSUE/AWS_EC2_OPERATIONAL_ISSUE_HNGHE_EXAMPLE111",
        "service": "EC2",
        "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
        "eventTypeCategory": "issue",
        "region": "ap-southeast-2",
        "startTime": 1583881487.483,
        "endTime": 1583885056.785,
        "lastUpdatedTime": 1583885057.052,
        "statusCode": "closed",
        "eventScopeCode": "PUBLIC"
    }
]
}

```

有关更多信息，请参阅 [《AWS 健康用户指南》中的 Person AWS al Health Dashboard 入门](#)。

示例 2：按服务和事件状态代码列出 He AWS alth 事件

以下describe-events示例列出了事件状态已关闭的亚马逊弹性计算云 (Amazon EC2) 的 AWS 健康事件。

```

aws health describe-events \
  --filter "services=EC2,eventStatusCodes=closed"

```

输出：

```

{
  "events": [
    {
      "arn": "arn:aws:health:us-east-1::event/EC2/AWS_EC2_OPERATIONAL_ISSUE/
AWS_EC2_OPERATIONAL_ISSUE_VKTXI_EXAMPLE111",
      "service": "EC2",
      "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
      "eventTypeCategory": "issue",
      "region": "us-east-1",
      "startTime": 1587462325.096,
      "endTime": 1587464204.774,
      "lastUpdatedTime": 1587464204.865,
    }
  ]
}

```

```
        "statusCode": "closed",
        "eventScopeCode": "PUBLIC"
    },
    {
        "arn": "arn:aws:health:us-east-1::event/EC2/AWS_EC2_OPERATIONAL_ISSUE/
AWS_EC2_OPERATIONAL_ISSUE_COBXJ_EXAMPLE111",
        "service": "EC2",
        "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
        "eventTypeCategory": "issue",
        "region": "us-east-1",
        "startTime": 1586473044.284,
        "endTime": 1586479706.091,
        "lastUpdatedTime": 1586479706.153,
        "statusCode": "closed",
        "eventScopeCode": "PUBLIC"
    },
    {
        "arn": "arn:aws:health:ap-southeast-2::event/EC2/
AWS_EC2_OPERATIONAL_ISSUE/AWS_EC2_OPERATIONAL_ISSUE_HNGHE_EXAMPLE111",
        "service": "EC2",
        "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
        "eventTypeCategory": "issue",
        "region": "ap-southeast-2",
        "startTime": 1583881487.483,
        "endTime": 1583885056.785,
        "lastUpdatedTime": 1583885057.052,
        "statusCode": "closed",
        "eventScopeCode": "PUBLIC"
    }
]
}
```

有关更多信息，请参阅 [《AWS 健康用户指南》](#) 中的 [Person AWS al Health Dashboard 入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEvents](#) 中的。

## HealthImaging 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 HealthImaging。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## copy-image-set

以下代码示例演示了如何使用 `copy-image-set`。

AWS CLI

例 1：复制没有目标的影像集。

以下 `copy-image-set` 代码示例可制作没有目标的影像集的副本。

```
aws medical-imaging copy-image-set \  
  --datastore-id 12345678901234567890123456789012 \  
  --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \  
  --copy-image-set-information '{"sourceImageSet": {"latestVersionId": "1" } }'
```

输出：

```
{  
  "destinationImageSetProperties": {  
    "latestVersionId": "2",  
    "imageSetWorkflowStatus": "COPYING",  
    "updatedAt": 1680042357.432,  
    "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",  
    "imageSetState": "LOCKED",  
    "createdAt": 1680042357.432  
  },  
  "sourceImageSetProperties": {  
    "latestVersionId": "1",  
    "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",  
    "updatedAt": 1680042357.432,  
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
```

```

    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436
  },
  "datastoreId": "12345678901234567890123456789012"
}

```

例 2：复制带有目标的影像集。

以下 `copy-image-set` 代码示例可制作带有目标的影像集的副本。

```

aws medical-imaging copy-image-set \
  --datastore-id 12345678901234567890123456789012 \
  --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \
  --copy-image-set-information '{"sourceImageSet": {"latestVersionId": "1" }},
  "destinationImageSet": { "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
  "latestVersionId": "1"} }'

```

输出：

```

{
  "destinationImageSetProperties": {
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "COPYING",
    "updatedAt": 1680042505.135,
    "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
    "imageSetState": "LOCKED",
    "createdAt": 1680042357.432
  },
  "sourceImageSetProperties": {
    "latestVersionId": "1",
    "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
    "updatedAt": 1680042505.135,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436
  },
  "datastoreId": "12345678901234567890123456789012"
}

```

有关更多信息，请参阅《AWS HealthImaging 开发者指南》中的[复制图像集](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CopyImageSet](#) 中的。

## create-datastore

以下代码示例演示了如何使用 create-datastore。

### AWS CLI

#### 创建数据存储

以下 create-datastore 代码示例创建名称为 my-datastore 的数据存储。

```
aws medical-imaging create-datastore \  
  --datastore-name "my-datastore"
```

输出：

```
{  
  "datastoreId": "12345678901234567890123456789012",  
  "datastoreStatus": "CREATING"  
}
```

有关更多信息，请参阅《AWS HealthImaging 开发人员指南》中的[创建数据存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDatastore](#)中的。

## delete-datastore

以下代码示例演示了如何使用 delete-datastore。

### AWS CLI

#### 删除数据存储

以下 delete-datastore 代码示例可删除数据存储。

```
aws medical-imaging delete-datastore \  
  --datastore-id "12345678901234567890123456789012"
```

输出：

```
{  
  "datastoreId": "12345678901234567890123456789012",  
  "datastoreStatus": "DELETING"  
}
```

```
}
```

有关更多信息，请参阅《AWS HealthImaging 开发人员指南》中的[删除数据存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDatastore](#)中的。

## delete-image-set

以下代码示例演示了如何使用 delete-image-set。

### AWS CLI

#### 删除映像集

以下 delete-image-set 代码示例可删除影像集。

```
aws medical-imaging delete-image-set \  
  --datastore-id 12345678901234567890123456789012 \  
  --image-set-id ea92b0d8838c72a3f25d00d13616f87e
```

输出：

```
{  
  "imageSetWorkflowStatus": "DELETING",  
  "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",  
  "imageSetState": "LOCKED",  
  "datastoreId": "12345678901234567890123456789012"  
}
```

有关更多信息，请参阅《AWS HealthImaging 开发者指南》中的[删除图像集](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteImageSet](#)中的。

## get-datastore

以下代码示例演示了如何使用 get-datastore。

### AWS CLI

#### 获取数据存储的属性

以下 get-datastore 代码示例可获取数据存储的属性。



```
aws medical-imaging get-datastore \  
  --datastore-id 12345678901234567890123456789012
```

输出：

```
{  
  "datastoreProperties": {  
    "datastoreId": "12345678901234567890123456789012",  
    "datastoreName": "TestDatastore123",  
    "datastoreStatus": "ACTIVE",  
    "datastoreArn": "arn:aws:medical-imaging:us-  
east-1:123456789012:datastore/12345678901234567890123456789012",  
    "createdAt": "2022-11-15T23:33:09.643000+00:00",  
    "updatedAt": "2022-11-15T23:33:09.643000+00:00"  
  }  
}
```

有关更多信息，请参阅《AWS HealthImaging 开发人员指南》中的[获取数据存储属性](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDatastore](#) 中的。

## get-dicom-import-job

以下代码示例演示了如何使用 `get-dicom-import-job`。

### AWS CLI

获取 DICOM 导入任务的属性

以下 `get-dicom-import-job` 代码示例可获取导入任务的属性。

```
aws medical-imaging get-dicom-import-job \  
  --datastore-id "12345678901234567890123456789012" \  
  --job-id "09876543210987654321098765432109"
```

输出：

```
{  
  "jobProperties": {  
    "jobId": "09876543210987654321098765432109",  
    "jobName": "my-job",
```

```

    "jobStatus": "COMPLETED",
    "datastoreId": "12345678901234567890123456789012",
    "dataAccessRoleArn": "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole",
    "endedAt": "2022-08-12T11:29:42.285000+00:00",
    "submittedAt": "2022-08-12T11:28:11.152000+00:00",
    "inputS3Uri": "s3://medical-imaging-dicom-input/dicom_input/",
    "outputS3Uri": "s3://medical-imaging-output/
job_output/12345678901234567890123456789012-
DicomImport-09876543210987654321098765432109/"
  }
}

```

有关更多信息，请参阅《AWS HealthImaging 开发人员指南》中的[获取导入任务属性](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》ImportJob 中的 [getDiCom](#)。

## get-image-frame

以下代码示例演示了如何使用 get-image-frame。

### AWS CLI

#### 获取影像集像素数据

以下 get-image-frame 代码示例可获取影像帧。

```

aws medical-imaging get-image-frame \
  --datastore-id "12345678901234567890123456789012" \
  --image-set-id "98765412345612345678907890789012" \
  --image-frame-information imageFrameId=3abf5d5d7ae72f80a0ec81b2c0de3ef4 \
  imageframe.jpg

```

注意：此代码示例不包括输出，因为该 GetImageFrame 操作将像素数据流返回到 imageframe.jpg 文件。有关解码和查看影像帧的信息，请参阅 HTJ2K 解码库。

有关更多信息，请参阅《AWS HealthImaging 开发者指南》中的[获取图像集像素数据](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetImageFrame](#) 中的。

## get-image-set-metadata

以下代码示例演示了如何使用 get-image-set-metadata。

## AWS CLI

### 例 1：获取没有版本的影像集元数据

以下 `get-image-set-metadata` 代码示例可获取未指定版本的影像集的元数据。

**注意：** `outfile` 是必需的参数。

```
aws medical-imaging get-image-set-metadata \  
  --datastore-id 12345678901234567890123456789012 \  
  --image-set-id ea92b0d8838c72a3f25d00d13616f87e \  
  studymetadata.json.gz
```

返回的元数据使用 `gzip` 压缩并存储在 `studymetadata.json.gz` 文件中。要查看返回的 JSON 对象的内容，必须先将其解压。

输出：

```
{  
  "contentType": "application/json",  
  "contentEncoding": "gzip"  
}
```

### 例 2：获取带有版本的影像集元数据

以下 `get-image-set-metadata` 代码示例可获取指定版本的影像集的元数据。

**注意：** `outfile` 是必需的参数。

```
aws medical-imaging get-image-set-metadata \  
  --datastore-id 12345678901234567890123456789012 \  
  --image-set-id ea92b0d8838c72a3f25d00d13616f87e \  
  --version-id 1 \  
  studymetadata.json.gz
```

返回的元数据使用 `gzip` 压缩并存储在 `studymetadata.json.gz` 文件中。要查看返回的 JSON 对象的内容，必须先将其解压。

输出：

```
{
```

```
"contentType": "application/json",
"contentEncoding": "gzip"
}
```

有关更多信息，请参阅《AWS HealthImaging 开发者指南》中的[获取图像集元数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetImageSetMetadata](#)中的。

## get-image-set

以下代码示例演示了如何使用 get-image-set。

### AWS CLI

#### 获取影像集属性

以下 get-image-set 代码示例可获取影像集的属性。

```
aws medical-imaging get-image-set \
  --datastore-id 12345678901234567890123456789012 \
  --image-set-id 18f88ac7870584f58d56256646b4d92b \
  --version-id 1
```

输出：

```
{
  "versionId": "1",
  "imageSetWorkflowStatus": "COPIED",
  "updatedAt": 1680027253.471,
  "imageSetId": "18f88ac7870584f58d56256646b4d92b",
  "imageSetState": "ACTIVE",
  "createdAt": 1679592510.753,
  "datastoreId": "12345678901234567890123456789012"
}
```

有关更多信息，请参阅《AWS HealthImaging 开发者指南》中的[获取图像集属性](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetImageSet](#)中的。

## list-datastores

以下代码示例演示了如何使用 list-datastores。

## AWS CLI

### 列出数据存储

以下 `list-datastores` 代码示例列出可用的数据存储。

```
aws medical-imaging list-datastores
```

输出：

```
{
  "datastoreSummaries": [
    {
      "datastoreId": "12345678901234567890123456789012",
      "datastoreName": "TestDatastore123",
      "datastoreStatus": "ACTIVE",
      "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
      "createdAt": "2022-11-15T23:33:09.643000+00:00",
      "updatedAt": "2022-11-15T23:33:09.643000+00:00"
    }
  ]
}
```

有关更多信息，请参阅AWS HealthImaging 开发者指南中的[列出数据存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDatastores](#)中的。

## **list-dicom-import-jobs**

以下代码示例演示了如何使用 `list-dicom-import-jobs`。

## AWS CLI

### 列出 DICOM 导入任务

以下 `list-dicom-import-jobs` 代码示例列出 DICOM 导入任务。

```
aws medical-imaging list-dicom-import-jobs \
  --datastore-id "12345678901234567890123456789012"
```

输出：

```
{
  "jobSummaries": [
    {
      "jobId": "09876543210987654321098765432109",
      "jobName": "my-job",
      "jobStatus": "COMPLETED",
      "datastoreId": "12345678901234567890123456789012",
      "dataAccessRoleArn": "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole",
      "endedAt": "2022-08-12T11:21:56.504000+00:00",
      "submittedAt": "2022-08-12T11:20:21.734000+00:00"
    }
  ]
}
```

有关更多信息，请参阅《AWS HealthImaging 开发人员指南》中的[列出导入任务](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》ImportJobs 中的 [ListDicom](#)。

## list-image-set-versions

以下代码示例演示了如何使用 `list-image-set-versions`。

### AWS CLI

列出影像集版本

以下 `list-image-set-versions` 代码示例列出了影像集的版本历史记录。

```
aws medical-imaging list-image-set-versions \
  --datastore-id 12345678901234567890123456789012 \
  --image-set-id ea92b0d8838c72a3f25d00d13616f87e
```

输出：

```
{
  "imageSetPropertiesList": [
    {
      "ImageSetWorkflowStatus": "UPDATED",
      "versionId": "4",
      "updatedAt": 1680029436.304,
```

```

    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "ACTIVE",
    "createdAt": 1680027126.436
  },
  {
    "ImageSetWorkflowStatus": "UPDATED",
    "versionId": "3",
    "updatedAt": 1680029163.325,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "ACTIVE",
    "createdAt": 1680027126.436
  },
  {
    "ImageSetWorkflowStatus": "COPY_FAILED",
    "versionId": "2",
    "updatedAt": 1680027455.944,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "ACTIVE",
    "message": "INVALID_REQUEST: Series of SourceImageSet and
DestinationImageSet don't match.",
    "createdAt": 1680027126.436
  },
  {
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "ACTIVE",
    "versionId": "1",
    "ImageSetWorkflowStatus": "COPIED",
    "createdAt": 1680027126.436
  }
]
}

```

有关更多信息，请参阅《AWS HealthImaging 开发者指南》中的[列出图像集版本](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListImageSetVersions](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 list-tags-for-resource。

### AWS CLI

#### 例 1：列出数据存储的资源标签

以下 `list-tags-for-resource` 代码示例列出数据存储的标签。

```
aws medical-imaging list-tags-for-resource \  
  --resource-arn "arn:aws:medical-imaging:us-  
east-1:123456789012:datastore/12345678901234567890123456789012"
```

输出：

```
{  
  "tags":{  
    "Deployment":"Development"  
  }  
}
```

例 2：列出影像集的资源标签

以下 `list-tags-for-resource` 代码示例列出影像集的标签。

```
aws medical-imaging list-tags-for-resource \  
  --resource-arn "arn:aws:medical-imaging:us-  
east-1:123456789012:datastore/12345678901234567890123456789012/  
imageset/18f88ac7870584f58d56256646b4d92b"
```

输出：

```
{  
  "tags":{  
    "Deployment":"Development"  
  }  
}
```

有关更多信息，请参阅《AWS HealthImaging 开发人员指南》AWS HealthImaging中的[使用为资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## search-image-sets

以下代码示例演示了如何使用 `search-image-sets`。



## AWS CLI

### 例 1：使用 EQUAL 运算符搜索影像集

以下 `search-image-sets` 代码示例使用 EQUAL 运算符根据特定值搜索影像集。

```
aws medical-imaging search-image-sets \  
  --datastore-id 12345678901234567890123456789012 \  
  --search-criteria file://search-criteria.json
```

### `search-criteria.json` 的内容

```
{  
  "filters": [{  
    "values": [{"DICOMPatientId" : "SUBJECT08701"}],  
    "operator": "EQUAL"  
  }]  
}
```

输出：

```
{  
  "imageSetsMetadataSummaries": [{  
    "imageSetId": "09876543210987654321098765432109",  
    "createdAt": "2022-12-06T21:40:59.429000+00:00",  
    "version": 1,  
    "DICOMTags": {  
      "DICOMStudyId": "2011201407",  
      "DICOMStudyDate": "19991122",  
      "DICOMPatientSex": "F",  
      "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",  
      "DICOMPatientBirthDate": "19201120",  
      "DICOMStudyDescription": "UNKNOWN",  
      "DICOMPatientId": "SUBJECT08701",  
      "DICOMPatientName": "Melissa844 Huel628",  
      "DICOMNumberOfStudyRelatedInstances": 1,  
      "DICOMStudyTime": "140728",  
      "DICOMNumberOfStudyRelatedSeries": 1  
    },  
    "updatedAt": "2022-12-06T21:40:59.429000+00:00"  
  }]  
}
```

## 示例 2：使用 DICOM 和 DICOM StudyDate 使用 BETWEEN 运算符搜索影像集 StudyTime

以下 `search-image-sets` 代码示例搜索在 1990 年 1 月 1 日 (12:00 AM) 至 2023 年 1 月 1 日 (12:00 AM) 之间生成的 DICOM 研究的影像集。

注意：DICOM StudyTime 是可选的。如果不存在，则上午 12:00（一天的开始）是提供用于筛选的日期的时间值。

```
aws medical-imaging search-image-sets \  
  --datastore-id 12345678901234567890123456789012 \  
  --search-criteria file://search-criteria.json
```

### search-criteria.json 的内容

```
{  
  "filters": [{  
    "values": [{  
      "DICOMStudyDateAndTime": {  
        "DICOMStudyDate": "19900101",  
        "DICOMStudyTime": "000000"  
      }  
    },  
    {  
      "DICOMStudyDateAndTime": {  
        "DICOMStudyDate": "20230101",  
        "DICOMStudyTime": "000000"  
      }  
    }  
  ]],  
  "operator": "BETWEEN"  
}]  
}
```

输出：

```
{  
  "imageSetsMetadataSummaries": [{  
    "imageSetId": "09876543210987654321098765432109",  
    "createdAt": "2022-12-06T21:40:59.429000+00:00",  
    "version": 1,  
    "DICOMTags": {  
      "DICOMStudyId": "2011201407",  
      "DICOMStudyDate": "19991122",  
    }  
  }  
]
```

```

        "DICOMPatientSex": "F",
        "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
        "DICOMPatientBirthDate": "19201120",
        "DICOMStudyDescription": "UNKNOWN",
        "DICOMPatientId": "SUBJECT08701",
        "DICOMPatientName": "Melissa844 Huel628",
        "DICOMNumberOfStudyRelatedInstances": 1,
        "DICOMStudyTime": "140728",
        "DICOMNumberOfStudyRelatedSeries": 1
    },
    "updatedAt": "2022-12-06T21:40:59.429000+00:00"
  ]
}

```

例 3：使用 CreatedAt，通过 BETWEEN 运算符搜索影像集（之前保留了时间研究）

以下 search-image-sets 代码示例搜索在 DICOM 研究保持在 UTC 时区时间范围 HealthImaging 之间的影像集。

注意：提供的 createdAt 仅作为示例（“1985-04-12T 23:20:50.52 Z”）。

```

aws medical-imaging search-image-sets \
  --datastore-id 12345678901234567890123456789012 \
  --search-criteria file://search-criteria.json

```

search-criteria.json 的内容

```

{
  "filters": [{
    "values": [{
      "createdAt": "1985-04-12T23:20:50.52Z"
    },
    {
      "createdAt": "2022-04-12T23:20:50.52Z"
    }
  ]],
  "operator": "BETWEEN"
}]
}

```

输出：

```

{
  "imageSetsMetadataSummaries": [{

```

```

    "imageSetId": "09876543210987654321098765432109",
    "createdAt": "2022-12-06T21:40:59.429000+00:00",
    "version": 1,
    "DICOMTags": {
      "DICOMStudyId": "2011201407",
      "DICOMStudyDate": "19991122",
      "DICOMPatientSex": "F",
      "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
      "DICOMPatientBirthDate": "19201120",
      "DICOMStudyDescription": "UNKNOWN",
      "DICOMPatientId": "SUBJECT08701",
      "DICOMPatientName": "Melissa844 Huel628",
      "DICOMNumberOfStudyRelatedInstances": 1,
      "DICOMStudyTime": "140728",
      "DICOMNumberOfStudyRelatedSeries": 1
    },
    "lastUpdatedAt": "2022-12-06T21:40:59.429000+00:00"
  ]
}

```

示例 4：在 DICOM SeriesInstance UID 上使用等号运算符搜索图像集，在 updateDat 上使用介于两者之间搜索图像集，然后在 updateDat 字段上按照 ASC 顺序对响应进行排序

以下 search-image-sets 代码示例在 DICOM SeriesInstance UID 上搜索具有等于运算符的影像集，在 updateDat 上使用介于两者之间的影像集，并在 updateDat 字段上按照 ASC 顺序对响应进行排序。

注意：以示例格式提供 updateDat ( "1985-04-12T 23:20:50.52 Z" )。

```

aws medical-imaging search-image-sets \
  --datastore-id 12345678901234567890123456789012 \
  --search-criteria file://search-criteria.json

```

search-criteria.json 的内容

```

{
  "filters": [{
    "values": [{
      "updatedAt": "2024-03-11T15:00:05.074000-07:00"
    }, {
      "updatedAt": "2024-03-11T16:00:05.074000-07:00"
    }
  ]],
  "operator": "BETWEEN"
}

```

```

    }, {
      "values": [{
        "DICOMSeriesInstanceUID": "1.2.840.99999999.84710745.943275268089"
      }],
      "operator": "EQUAL"
    }],
    "sort": {
      "sortField": "updatedAt",
      "sortOrder": "ASC"
    }
  }
}

```

输出：

```

{
  "imageSetsMetadataSummaries": [{
    "imageSetId": "09876543210987654321098765432109",
    "createdAt": "2022-12-06T21:40:59.429000+00:00",
    "version": 1,
    "DICOMTags": {
      "DICOMStudyId": "2011201407",
      "DICOMStudyDate": "19991122",
      "DICOMPatientSex": "F",
      "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
      "DICOMPatientBirthDate": "19201120",
      "DICOMStudyDescription": "UNKNOWN",
      "DICOMPatientId": "SUBJECT08701",
      "DICOMPatientName": "Melissa844 Huel628",
      "DICOMNumberOfStudyRelatedInstances": 1,
      "DICOMStudyTime": "140728",
      "DICOMNumberOfStudyRelatedSeries": 1
    },
    "lastUpdatedAt": "2022-12-06T21:40:59.429000+00:00"
  }]
}

```

有关更多信息，请参阅《AWS HealthImaging 开发者指南》中的[搜索图像集](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SearchImageSets](#)中的。

## start-dicom-import-job

以下代码示例演示了如何使用 start-dicom-import-job。

## AWS CLI

### 启动 DICOM 导入任务

以下 `start-dicom-import-job` 代码示例启动 DICOM 导入任务。

```
aws medical-imaging start-dicom-import-job \  
  --job-name "my-job" \  
  --datastore-id "12345678901234567890123456789012" \  
  --input-s3-uri "s3://medical-imaging-dicom-input/dicom_input/" \  
  --output-s3-uri "s3://medical-imaging-output/job_output/" \  
  --data-access-role-arn "arn:aws:iam::123456789012:role/ImportJobDataAccessRole"
```

输出：

```
{  
  "datastoreId": "12345678901234567890123456789012",  
  "jobId": "09876543210987654321098765432109",  
  "jobStatus": "SUBMITTED",  
  "submittedAt": "2022-08-12T11:28:11.152000+00:00"  
}
```

有关更多信息，请参阅《AWS HealthImaging 开发人员指南》中的[启动导入任务](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》ImportJob 中的 [StartDicom](#)。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

### AWS CLI

#### 例 1：标记数据存储

以下 `tag-resource` 代码示例可标记数据存储。

```
aws medical-imaging tag-resource \  
  --resource-arn "arn:aws:medical-imaging:us-  
east-1:123456789012:datastore/12345678901234567890123456789012" \  
  --tags '{"Deployment":"Development"}'
```

此命令不生成任何输出。

## 例 2：标记影像集

以下 `tag-resource` 代码示例可标记影像集。

```
aws medical-imaging tag-resource \  
  --resource-arn "arn:aws:medical-imaging:us-  
east-1:123456789012:datastore/12345678901234567890123456789012/  
imageset/18f88ac7870584f58d56256646b4d92b" \  
  --tags '{"Deployment":"Development"}'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS HealthImaging 开发人员指南》AWS HealthImaging 中的[使用为资源添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 `untag-resource`。

### AWS CLI

#### 例 1：取消标记数据存储

以下 `untag-resource` 代码示例可取消标记数据存储。

```
aws medical-imaging untag-resource \  
  --resource-arn "arn:aws:medical-imaging:us-  
east-1:123456789012:datastore/12345678901234567890123456789012" \  
  --tag-keys ['"Deployment"]'
```

此命令不生成任何输出。

#### 例 2：取消标记影像集

以下 `untag-resource` 代码示例可取消标记影像集。

```
aws medical-imaging untag-resource \  
  --resource-arn "arn:aws:medical-imaging:us-  
east-1:123456789012:datastore/12345678901234567890123456789012/  
imageset/18f88ac7870584f58d56256646b4d92b" \  
  --tag-keys ['"Deployment"]'
```

```
--tag-keys '["Deployment"]'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS HealthImaging 开发人员指南》AWS HealthImaging中的[使用为资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-image-set-metadata

以下代码示例演示了如何使用 update-image-set-metadata。

### AWS CLI

在影像集元数据中插入或更新属性

以下update-image-set-metadata代码示例在影像集元数据中插入或更新属性。

```
aws medical-imaging update-image-set-metadata \  
  --datastore-id 12345678901234567890123456789012 \  
  --image-set-id ea92b0d8838c72a3f25d00d13616f87e \  
  --latest-version-id 1 \  
  --update-image-set-metadata-updates file://metadata-updates.json
```

metadata-updates.json 的内容

```
{  
  "DICOMUpdates": {  
    "updatableAttributes":  
    "eyJTY2h1bWFWZXJzaW9uIjoxLjEsIlBhdGllbnQiOnsiRElDT00iOnsiUGF0aWVudE5hbWUiOiJNWf5NWCJ9fX0=" }  
  }  
}
```

注意：updatableAttributes 是一个采用 Base64 编码的 JSON 字符串。这是未编码的 JSON 字符串。

```
{ "SchemaVersion": 1.1, "Patient": { "DICOM": { "PatientName": "MX^MX" } } }
```

输出：

```
{
```



```

    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
  }

```

## 从影像集元数据中移除属性

以下 `update-image-set-metadata` 代码示例从影像集元数据中移除一个属性。

```

aws medical-imaging update-image-set-metadata \
  --datastore-id 12345678901234567890123456789012 \
  --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
  --latest-version-id 1 \
  --update-image-set-metadata-updates file://metadata-updates.json

```

## metadata-updates.json 的内容

```

{
  "DICOMUpdates": {
    "removableAttributes":
    "e1NjaGVtYVZlcnNpb246MS4xLFN0dWR50ntESUNPTTp7U3R1ZH1EZnJcm1wdGlvbWpDSEVTVH19fQo="
  }
}

```

注意：`removableAttributes` 是一个采用 Base64 编码的 JSON 字符串。这是未编码的 JSON 字符串。键和值必须与要删除的属性相匹配。

```
{ "SchemaVersion": "1.1", "Study": { "DICOM": { "StudyDescription": "CHEST" } } }
```

输出：

```

{
  "latestVersionId": "2",
  "imageSetWorkflowStatus": "UPDATING",
  "updatedAt": 1680042257.908,
  "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
  "imageSetState": "LOCKED",
  "createdAt": 1680027126.436,
  "datastoreId": "12345678901234567890123456789012"
}

```

```
}

```

## 从影像集元数据中移除实例

以下update-image-set-metadata代码示例从影像集元数据中移除实例。

```
aws medical-imaging update-image-set-metadata \
  --datastore-id 12345678901234567890123456789012 \
  --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
  --latest-version-id 1 \
  --update-image-set-metadata-updates file://metadata-updates.json

```

## metadata-updates.json 的内容

```
{
  "DICOMUpdates": {
    "removableAttributes":
      "eezEuMS4xLjEuMS4xLjEyMzQ1LjEyMzQ1Njc4OTAxMi4xMjMuMTIzNDU2Nzg5MDEyMzQuMTp7SW5zdGFuY2VzOnsxL
    }
  }
}

```

注意：removableAttributes 是一个采用 Base64 编码的 JSON 字符串。这是未编码的 JSON 字符串。

```
{"1.1.1.1.1.1.1.1.1.12345.123456789012.123.12345678901234.1" : {"实例" :
{"1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.12345.123456789012.123.12345678901234.1" : {}}}}}
```

输出：

```
{
  "latestVersionId": "2",
  "imageSetWorkflowStatus": "UPDATING",
  "updatedAt": 1680042257.908,
  "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
  "imageSetState": "LOCKED",
  "createdAt": 1680027126.436,
  "datastoreId": "12345678901234567890123456789012"
}

```

有关更多信息，请参阅《AWS HealthImaging 开发者指南》中的[更新图像集元数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateImageSetMetadata](#)中的。

## HealthLake 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 HealthLake。 AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-fhir-datastore**

以下代码示例演示了如何使用 create-fhir-datastore。

AWS CLI

创建 FHIR 数据存储。

以下create-fhir-datastore示例演示了如何在 Amazon 中创建新的数据存储 HealthLake。

```
aws healthlake create-fhir-datastore \  
  --region us-east-1 \  
  --datastore-type-version R4 \  
  --datastore-type-version R4 \  
  --datastore-name "FhirTestDatastore"
```

输出：

```
{  
  "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/  
(Datastore ID)/r4/",  
  "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/  
(Datastore ID)",
```

```
"DatastoreStatus": "CREATING",
  "DatastoreId": "(Datastore ID)"
}
```

有关更多信息，请参阅《亚马逊 HealthLake 开发者指南》中的[创建和监控 FHIR 数据存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFhirDatastore](#)中的。

## delete-fhir-datastore

以下代码示例演示了如何使用 delete-fhir-datastore。

### AWS CLI

#### 删除 FHIR 数据存储

以下delete-fhir-datastore示例演示如何删除 Amazon 中的数据存储及其所有内容 HealthLake。

```
aws healthlake delete-fhir-datastore \
  --datastore-id (Data Store ID) \
  --region us-east-1
```

输出：

```
{
  "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Datastore ID)/r4/",
  "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Datastore ID)",
  "DatastoreStatus": "DELETING",
  "DatastoreId": "(Datastore ID)"
}
```

有关更多信息，请参阅《< <https://docs.aws.amazon.com/healthlake/latest/devguide/working-with-FHIR-healthlake.html> > 亚马逊 HealthLake 开发者指南》中的创建和监控 FHIR 数据存储。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFhirDatastore](#)中的。

## describe-fhir-datastore

以下代码示例演示了如何使用 describe-fhir-datastore。

## AWS CLI

### 描述 FHIR 数据存储

以下describe-fhir-datastore示例演示了如何在 Amazon 中查找数据存储的属性 HealthLake。

```
aws healthlake describe-fhir-datastore \  
  --datastore-id "1f2f459836ac6c513ce899f9e4f66a59" \  
  --region us-east-1
```

输出：

```
{  
  "DatastoreProperties": {  
    "PreloadDataConfig": {  
      "PreloadDataType": "SYNTHEA"  
    },  
    "DatastoreName": "FhirTestDatastore",  
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/  
(Datastore ID)",  
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/  
(Datastore ID)/r4/",  
    "DatastoreStatus": "CREATING",  
    "DatastoreTypeVersion": "R4",  
    "DatastoreId": "(Datastore ID)"  
  }  
}
```

有关更多信息，请参阅《亚马逊 HealthLake 开发者指南》中的[创建和监控 FHIR 数据存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFhirDatastore](#)中的。

## describe-fhir-export-job

以下代码示例演示了如何使用 describe-fhir-export-job。

## AWS CLI

### 描述 FHIR 导出任务

以下describe-fhir-export-job示例说明如何在 Amazon HealthLake 中查找 FHIR 导出任务的属性。

```
aws healthlake describe-fhir-export-job \  
  --datastore-id (Datastore ID) \  
  --job-id 9b9a51943afaedd0a8c0c26c49135a31
```

输出：

```
{  
  "ExportJobProperties": {  
    "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",  
    "JobStatus": "IN_PROGRESS",  
    "JobId": "9009813e9d69ba7cf79bcb3468780f16",  
    "SubmitTime": 1609175692.715,  
    "OutputDataConfig": {  
      "S3Uri": "s3://(Bucket Name)/(Prefix  
Name)/59593b2d0367ce252b5e66bf5fd6b574-  
FHIR_EXPORT-9009813e9d69ba7cf79bcb3468780f16/"  
    },  
    "DatastoreId": "(Datastore ID)"  
  }  
}
```

有关更多信息，请参阅《Amazon HealthLake 开发者指南》中的从 FHIR 数据存储中导出文件。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeFhirExportJob](#) 中的。

## describe-fhir-import-job

以下代码示例演示了如何使用 `describe-fhir-import-job`。

### AWS CLI

描述 FHIR 导入任务

以下 `describe-fhir-import-job` 示例说明如何使用 Amazon HealthLake 学习 FHIR 导入任务的属性。

```
aws healthlake describe-fhir-import-job \  
  --datastore-id (Datastore ID) \  
  --job-id c145fbb27b192af392f8ce6e7838e34f \  
  --region us-east-1
```

输出：

```
{
  "ImportJobProperties": {
    "InputDataConfig": {
      "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
      { "arrayitem2": 2 }
    },
    "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
    "JobStatus": "COMPLETED",
    "JobId": "c145fbb27b192af392f8ce6e7838e34f",
    "SubmitTime": 1606272542.161,
    "EndTime": 1606272609.497,
    "DatastoreId": "(Datastore ID)"
  }
}
```

有关更多信息，请参阅《Amazon HealthLake 开发者指南》中的[将文件导入 FHIR 数据存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFhirImportJob](#)中的。

## list-fhir-datastores

以下代码示例演示了如何使用 list-fhir-datastores。

### AWS CLI

列出 FHIR 数据存储

以下list-fhir-datastores示例说明如何使用该命令以及用户如何根据亚马逊中的数据存储状态筛选结果 HealthLake。

```
aws healthlake list-fhir-datastores \
  --region us-east-1 \
  --filter DatastoreStatus=ACTIVE
```

输出：

```
{
  "DatastorePropertiesList": [
    {
      "PreloadDataConfig": {
        "PreloadDataType": "SYNTHEA"
      },

```

```

    "DatastoreName": "FhirTestDatastore",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Datastore ID>",
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
<Datastore ID>/r4/",
    "DatastoreStatus": "ACTIVE",
    "DatastoreTypeVersion": "R4",
    "CreatedAt": 1605574003.209,
    "DatastoreId": "<Datastore ID>"
  },
  {
    "DatastoreName": "Demo",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Datastore ID>",
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
<Datastore ID>/r4/",
    "DatastoreStatus": "ACTIVE",
    "DatastoreTypeVersion": "R4",
    "CreatedAt": 1603761064.881,
    "DatastoreId": "<Datastore ID>"
  }
]
}

```

有关更多信息，请参阅《亚马逊 HealthLake 开发者指南》中的[创建和监控 FHIR 数据存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListFhirDatastores](#) 中的。

## list-fhir-export-jobs

以下代码示例演示了如何使用 `list-fhir-export-jobs`。

### AWS CLI

列出所有 FHIR 导出任务

以下 `list-fhir-export-jobs` 示例说明如何使用命令查看与账户关联的导出任务列表。

```

aws healthlake list-fhir-export-jobs \
  --datastore-id (Datastore ID) \
  --submitted-before (DATE like 2024-10-13T19:00:00Z)\
  --submitted-after (DATE like 2020-10-13T19:00:00Z) \
  --job-name "FHIR-EXPORT" \

```



```
--job-status SUBMITTED \
--max-results (Integer between 1 and 500)
```

输出：

```
{
  "ExportJobProperties": {
    "OutputDataConfig": {
      "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
      "S3Configuration": {
        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
        "KmsKeyId" : "(KmsKey Id)"
      },
    },
    "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
    "JobStatus": "COMPLETED",
    "JobId": "c145fbb27b192af392f8ce6e7838e34f",
    "JobName": "FHIR-EXPORT",
    "SubmitTime": 1606272542.161,
    "EndTime": 1606272609.497,
    "DatastoreId": "(Datastore ID)"
  }
}
"NextToken": String
```

有关更多信息，请参阅《Amazon HealthLake 开发者指南》中的[从 FHIR 数据存储中导出文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListFhirExportJobs](#) 中的。

## list-fhir-import-jobs

以下代码示例演示了如何使用 `list-fhir-import-jobs`。

### AWS CLI

列出所有 FHIR 导入任务

以下 `list-fhir-import-jobs` 示例说明如何使用命令查看与账户关联的所有导入任务的列表。

```
aws healthlake list-fhir-import-jobs \
  --datastore-id (Datastore ID) \
  --submitted-before (DATE like 2024-10-13T19:00:00Z) \
  --submitted-after (DATE like 2020-10-13T19:00:00Z ) \
```

```
--job-name "FHIR-IMPORT" \  
--job-status SUBMITTED \  
-max-results (Integer between 1 and 500)
```

输出：

```
{  
  "ImportJobProperties": {  
    "OutputDataConfig": {  
      "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",  
      "S3Configuration": {  
        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",  
        "KmsKeyId" : "(KmsKey Id)"  
      },  
    },  
  },  
  "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",  
  "JobStatus": "COMPLETED",  
  "JobId": "c145fbb27b192af392f8ce6e7838e34f",  
  "JobName": "FHIR-IMPORT",  
  "SubmitTime": 1606272542.161,  
  "EndTime": 1606272609.497,  
  "DatastoreId": "(Datastore ID)"  
}  
"NextToken": String
```

有关更多信息，请参阅《亚马逊 HealthLake 开发者指南》中的[将文件导入 FHIR 数据存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListFhirImportJobs](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 list-tags-for-resource。

### AWS CLI

列出数据存储的标签

以下 list-tags-for-resource 示例列出了与指定数据存储关联的标签。：

```
aws healthlake list-tags-for-resource \  
  --resource-arn "arn:aws:healthlake:us-east-1:674914422125:datastore/  
  fhir/0725c83f4307f263e16fd56b6d8ebdbe" \  
  --max-results 1
```

```
--region us-east-1
```

输出：

```
{
  "tags": {
    "key": "value",
    "key1": "value1"
  }
}
```

有关更多信息，请参阅《亚马逊 HealthLake 开发者指南》中的[在亚马逊 HealthLake 中为资源添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListTagsForResource](#) 中的。

## start-fhir-export-job

以下代码示例演示了如何使用 start-fhir-export-job。

AWS CLI

启动 FHIR 导出任务

以下 start-fhir-export-job 示例说明如何使用 Amazon HealthLake 启动 FHIR 导出任务。

```
aws healthlake start-fhir-export-job \
  --output-data-config S3Uri="s3://(Bucket Name)/(Prefix Name)/" \
  --datastore-id (Datastore ID) \
  --data-access-role-arn arn:aws:iam::(AWS Account ID):role/(Role Name)
```

输出：

```
{
  "DatastoreId": "(Datastore ID)",
  "JobStatus": "SUBMITTED",
  "JobId": "9b9a51943afaedd0a8c0c26c49135a31"
}
```

有关更多信息，请参阅《Amazon HealthLake 开发者指南》中的从 FHIR 数据存储中[导出文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartFhirExportJob](#)中的。

## start-fhir-import-job

以下代码示例演示了如何使用 start-fhir-import-job。

### AWS CLI

#### 启动 FHIR 导入任务

以下start-fhir-import-job示例说明如何使用 Amazon HealthLake 启动 FHIR 导入任务。

```
aws healthlake start-fhir-import-job \  
  --input-data-config S3Uri="s3://(Bucket Name)/(Prefix Name)/" \  
  --datastore-id (Datastore ID) \  
  --data-access-role-arn "arn:aws:iam::(AWS Account ID):role/(Role Name)" \  
  --region us-east-1
```

输出：

```
{  
  "DatastoreId": "(Datastore ID)",  
  "JobStatus": "SUBMITTED",  
  "JobId": "c145fbb27b192af392f8ce6e7838e34f"  
}
```

有关更多信息，请参阅《亚马逊 HealthLake 开发者指南》中的“将文件导入 FHIR 数据存储”  
<https://docs.aws.amazon.com/healthlake/latest/devguide/import-datastore.html>。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartFhirImportJob](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

#### 向数据存储添加标签

以下tag-resource示例说明如何向数据存储添加标签。

```
aws healthlake tag-resource \  
  --resource-id (Datastore ID) \  
  --tag-key (Tag Key) \  
  --tag-value (Tag Value) \  
  --region us-east-1
```

```
--resource-arn "arn:aws:healthlake:us-east-1:691207106566:datastore/
fhir/0725c83f4307f263e16fd56b6d8ebdbe" \
--tags '[{"Key": "key1", "Value": "value1"}]' \
--region us-east-1
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊开发者指南》中的“向数据存储添加标签 < <https://docs.aws.amazon.com/healthlake/latest/devguide/add-a-tag.html>>’。HealthLake。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从数据存储中移除标签。

以下 untag-resource 示例说明如何从数据存储中移除标签。

```
aws healthlake untag-resource \
--resource-arn "arn:aws:healthlake:us-east-1:674914422125:datastore/fhir/
b91723d65c6fdeb1d26543a49d2ed1fa" \
--tag-keys '["key1"]' \
--region us-east-1
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon HealthLake 开发者指南》中的[从数据存储中移除标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## HealthOmics 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 HealthOmics。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## **abort-multipart-read-set-upload**

以下代码示例演示了如何使用 `abort-multipart-read-set-upload`。

AWS CLI

要停止分段读取集上传

以下`abort-multipart-read-set-upload`示例停止将分段读取集上传到您的 HealthOmics 序列存储中。

```
aws omics abort-multipart-read-set-upload \  
  --sequence-store-id 0123456789 \  
  --upload-id 1122334455
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS HealthOmics 用户指南》中的[直接上传到序列存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AbortMultipartReadSetUpload](#)中的。

## **accept-share**

以下代码示例演示了如何使用 `accept-share`。

AWS CLI

要接受分析存储数据的共享

以下`accept-share`示例接受一部分 HealthOmics 分析商店数据。

```
aws omics accept-share \  
----share-id "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a"
```

输出：

```
{  
  "status": "ACTIVATING"  
}
```

有关更多信息，请参阅AWS HealthOmics 用户指南中的[跨账户共享](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AcceptShare](#)中的。

## batch-delete-read-set

以下代码示例演示了如何使用 batch-delete-read-set。

### AWS CLI

删除多个读取集

以下batch-delete-read-set示例删除了两个读取集。

```
aws omics batch-delete-read-set \  
--sequence-store-id 1234567890 \  
--ids 1234567890 0123456789
```

如果删除任何指定的读取集时出错，则该服务会返回错误列表。

```
{  
  "errors": [  
    {  
      "code": "",  
      "id": "0123456789",  
      "message": "The specified readset does not exist."  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchDeleteReadSet](#)中的。

## cancel-annotation-import-job

以下代码示例演示了如何使用 `cancel-annotation-import-job`。

### AWS CLI

#### 取消注释导入任务

以下`cancel-annotation-import-job`示例取消了 ID 为 ID 04f57618-xmpl-4fd0-9349-e5a85aefb997 的注释导入任务。

```
aws omics cancel-annotation-import-job \  
  --job-id 04f57618-xmpl-4fd0-9349-e5a85aefb997
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analyt ics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelAnnotationImportJob](#)中的。

## cancel-run

以下代码示例演示了如何使用 `cancel-run`。

### AWS CLI

#### 取消跑步

以下`cancel-run`示例取消带有 ID 1234567 的运行。

```
aws omics cancel-run \  
  --id 1234567
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelRun](#)中的。

## cancel-variant-import-job

以下代码示例演示了如何使用 `cancel-variant-import-job`。



## AWS CLI

### 取消变体导入任务

以下cancel-variant-import-job示例取消了带有 ID 69cb65d6-xmpl-4a4a-9025-4565794b684e 的变体导入任务。

```
aws omics cancel-variant-import-job \  
  --job-id 69cb65d6-xmpl-4a4a-9025-4565794b684e
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analyt ics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelVariantImportJob](#)中的。

## complete-multipart-read-set-upload

以下代码示例演示了如何使用 complete-multipart-read-set-upload。

### AWS CLI

在上传完所有组件后，结束分段上传。

以下complete-multipart-read-set-upload示例将在所有组件都上传完毕后结束向序列存储的分段上传。

```
aws omics complete-multipart-read-set-upload \  
  --sequence-store-id 0123456789 \  
  --upload-id 1122334455 \  
  --parts '[{"checksum":"gaCBQMe+rpCFZxLpoP6gydBoXaKKDA/  
Vobh5zBDb4W4=","partNumber":1,"partSource":"SOURCE1"}]'
```

输出：

```
{  
  "readSetId": "0000000001"  
  "readSetId": "0000000002"  
  "readSetId": "0000000003"  
}
```

有关更多信息，请参阅《AWS HealthOmics 用户指南》中的[直接上传到序列存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CompleteMultipartReadSetUpload](#)中的。

## create-annotation-store-version

以下代码示例演示了如何使用 `create-annotation-store-version`。

### AWS CLI

创建注释存储库的新版本

以下 `create-annotation-store-version` 示例创建了注释存储库的新版本。

```
aws omics create-annotation-store-version \  
  --name my_annotation_store \  
  --version-name my_version
```

输出：

```
{  
  "creationTime": "2023-07-21T17:15:49.251040+00:00",  
  "id": "3b93cdef69d2",  
  "name": "my_annotation_store",  
  "reference": {  
    "referenceArn": "arn:aws:omics:us-  
west-2:555555555555:referenceStore/6505293348/reference/5987565360"  
  },  
  "status": "CREATING",  
  "versionName": "my_version"  
}
```

有关更多信息，请参阅AWS HealthOmics 用户指南中的[创建注释存储库的新版本](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAnnotationStoreVersion](#)中的。

## create-annotation-store

以下代码示例演示了如何使用 `create-annotation-store`。

### AWS CLI

示例 1：创建 VCF 注释存储库

以下 `create-annotation-store` 示例创建了 VCF 格式的注释存储库。

```
aws omics create-annotation-store \  
  --name my_annotation_store \  
  --format vcf
```

```
--name my_ann_store \  
--store-format VCF \  
--reference referenceArn=arn:aws:omics:us-  
west-2:123456789012:referenceStore/1234567890/reference/1234567890
```

输出：

```
{  
  "creationTime": "2022-11-23T22:48:39.226492Z",  
  "id": "0a91xmplc71f",  
  "name": "my_ann_store",  
  "reference": {  
    "referenceArn": "arn:aws:omics:us-  
west-2:123456789012:referenceStore/1234567890/reference/1234567890"  
  },  
  "status": "CREATING",  
  "storeFormat": "VCF"  
}
```

## 示例 2：创建 TSV 注释存储库

以下create-annotation-store示例创建 TSV 格式的注释存储库。

```
aws omics create-annotation-store \  
  --name tsv_ann_store \  
  --store-format TSV \  
  --reference referenceArn=arn:aws:omics:us-  
west-2:123456789012:referenceStore/1234567890/reference/1234567890 \  
  --store-options file://tsv-store-options.json
```

tsv-store-options.json为注释配置格式选项。

```
{  
  "tsvStoreOptions": {  
    "annotationType": "CHR_START_END_ZERO_BASE",  
    "formatToHeader": {  
      "CHR": "chromosome",  
      "START": "start",  
      "END": "end"  
    },  
    "schema": [  
      {
```

```

        "chromosome": "STRING"
      },
      {
        "start": "LONG"
      },
      {
        "end": "LONG"
      },
      {
        "name": "STRING"
      }
    ]
  }
}

```

输出：

```

{
  "creationTime": "2022-11-30T01:28:08.525586Z",
  "id": "861cxmpl96b0",
  "name": "tsv_ann_store",
  "reference": {
    "referenceArn": "arn:aws:omics:us-west-2:123456789012:referenceStore/1234567890/reference/1234567890"
  },
  "status": "CREATING",
  "storeFormat": "TSV",
  "storeOptions": {
    "tsvStoreOptions": {
      "annotationType": "CHR_START_END_ZERO_BASE",
      "formatToHeader": {
        "CHR": "chromosome",
        "END": "end",
        "START": "start"
      }
    },
    "schema": [
      {
        "chromosome": "STRING"
      },
      {
        "start": "LONG"
      }
    ]
  }
}

```

```
        "end": "LONG"
      },
      {
        "name": "STRING"
      }
    ]
  }
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analytics](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateAnnotationStore](#) 中的。

## create-multipart-read-set-upload

以下代码示例演示了如何使用 `create-multipart-read-set-upload`。

### AWS CLI

要开始分段读取，请先上传。

以下 `create-multipart-read-set-upload` 示例启动分段读取集上传。

```
aws omics create-multipart-read-set-upload \
  --sequence-store-id 0123456789 \
  --name HG00146 \
  --source-file-type FASTQ \
  --subject-id mySubject\
  --sample-id mySample\
  --description "FASTQ for HG00146"\
  --generated-from "1000 Genomes"
```

输出：

```
{
  "creationTime": "2022-07-13T23:25:20Z",
  "description": "FASTQ for HG00146",
  "generatedFrom": "1000 Genomes",
  "name": "HG00146",
  "sampleId": "mySample",
  "sequenceStoreId": "0123456789",
  "sourceFileType": "FASTQ",
```

```
"subjectId": "mySubject",
"uploadId": "1122334455"
}
```

有关更多信息，请参阅《AWS HealthOmics 用户指南》中的[直接上传到序列存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateMultipartReadSetUpload](#)中的。

## create-reference-store

以下代码示例演示了如何使用 create-reference-store。

### AWS CLI

#### 创建参考存储库

以下create-reference-store示例创建了一个参考存储库my-ref-store。

```
aws omics create-reference-store \
  --name my-ref-store
```

输出：

```
{
  "arn": "arn:aws:omics:us-west-2:123456789012:referenceStore/1234567890",
  "creationTime": "2022-11-22T22:13:25.947Z",
  "id": "1234567890",
  "name": "my-ref-store"
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateReferenceStore](#)中的。

## create-run-group

以下代码示例演示了如何使用 create-run-group。

### AWS CLI

#### 创建跑步组

以下create-run-group示例创建了一个名为的运行组cram-converter。

```
aws omics create-run-group \  
  --name cram-converter \  
  --max-cpus 20 \  
  --max-duration 600
```

输出：

```
{  
  "arn": "arn:aws:omics:us-west-2:123456789012:runGroup/1234567",  
  "id": "1234567",  
  "tags": {}  
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRunGroup](#)中的。

## create-sequence-store

以下代码示例演示了如何使用 create-sequence-store。

### AWS CLI

#### 创建序列存储

以下create-sequence-store示例创建了一个序列存储。

```
aws omics create-sequence-store \  
  --name my-seq-store
```

输出：

```
{  
  "arn": "arn:aws:omics:us-west-2:123456789012:sequenceStore/1234567890",  
  "creationTime": "2022-11-23T01:24:33.629Z",  
  "id": "1234567890",  
  "name": "my-seq-store"  
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSequenceStore](#)中的。

## create-share

以下代码示例演示了如何使用 create-share。

### AWS CLI

创建 HealthOmics 分析商店的共享

以下create-share示例说明如何创建可供账户外部订阅者接受的 HealthOmics 分析商店共享。

```
aws omics create-share \  
  --resource-arn "arn:aws:omics:us-west-2:555555555555:variantStore/  
omics_dev_var_store" \  
  --principal-subscriber "123456789012" \  
  --name "my_Share-123"
```

输出：

```
{  
  "shareId": "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a",  
  "name": "my_Share-123",  
  "status": "PENDING"  
}
```

有关更多信息，请参阅《AWS HealthOmics 用户指南》中的[跨账户共享](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateShare](#)中的。

## create-variant-store

以下代码示例演示了如何使用 create-variant-store。

### AWS CLI

创建多属性商店

以下create-variant-store示例创建了一个名为的变体商店my\_var\_store。

```
aws omics create-variant-store \  
  --name my_var_store \  
  --reference referenceArn=arn:aws:omics:us-  
west-2:123456789012:referenceStore/1234567890/reference/1234567890
```



输出：

```
{
  "creationTime": "2022-11-23T22:09:07.534499Z",
  "id": "02dexplcfd",
  "name": "my_var_store",
  "reference": {
    "referenceArn": "arn:aws:omics:us-
west-2:123456789012:referenceStore/1234567890/reference/1234567890"
  },
  "status": "CREATING"
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analyt ics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVariantStore](#)中的。

## create-workflow

以下代码示例演示了如何使用 create-workflow。

### AWS CLI

#### 创建工作流程

以下create-workflow示例创建了 WDL 工作流程。

```
aws omics create-workflow \
  --name cram-converter \
  --engine WDL \
  --definition-zip fileb://workflow-crambam.zip \
  --parameter-template file://workflow-params.json
```

workflow-crambam.zip是一个包含工作流程定义的 ZIP 存档。 workflow-params.json定义工作流程的运行时参数。

```
{
  "ref_fasta" : {
    "description": "Reference genome fasta file",
    "optional": false
  },
  "ref_fasta_index" : {
```

```
    "description": "Index of the reference genome fasta file",
    "optional": false
  },
  "ref_dict" : {
    "description": "dictionary file for 'ref_fasta'",
    "optional": false
  },
  "input_cram" : {
    "description": "The Cram file to convert to BAM",
    "optional": false
  },
  "sample_name" : {
    "description": "The name of the input sample, used to name the output BAM",
    "optional": false
  }
}
```

输出：

```
{
  "arn": "arn:aws:omics:us-west-2:123456789012:workflow/1234567",
  "id": "1234567",
  "status": "CREATING",
  "tags": {}
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateWorkflow](#)中的。

## delete-annotation-store-versions

以下代码示例演示了如何使用 delete-annotation-store-versions。

### AWS CLI

#### 删除注释库版本

以下delete-annotation-store-versions示例删除注释库版本。

```
aws omics delete-annotation-store-versions \
  --name my_annotation_store \
```

```
--versions my_version
```

输出：

```
{
  "errors": []
}
```

有关更多信息，请参阅AWS HealthOmics 用户指南中的[创建注释存储库的新版本](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAnnotationStoreVersions](#)中的。

## delete-annotation-store

以下代码示例演示了如何使用 delete-annotation-store。

### AWS CLI

删除注释存储库

以下delete-annotation-store示例删除名为的注释存储库my\_vcf\_store。

```
aws omics delete-annotation-store \
  --name my_vcf_store
```

输出：

```
{
  "status": "DELETING"
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analyt ics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAnnotationStore](#)中的。

## delete-reference-store

以下代码示例演示了如何使用 delete-reference-store。

### AWS CLI

删除参考资料库

以下delete-reference-store示例删除标识为 ID 的参考存储库1234567890。

```
aws omics delete-reference-store \  
  --id 1234567890
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteReferenceStore](#)中的。

## delete-reference

以下代码示例演示了如何使用 delete-reference。

### AWS CLI

删除参考文献

以下delete-reference示例删除引用。

```
aws omics delete-reference \  
  --reference-store-id 1234567890 \  
  --id 1234567890
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteReference](#)中的。

## delete-run-group

以下代码示例演示了如何使用 delete-run-group。

### AWS CLI

删除跑步组

以下delete-run-group示例删除标识为 ID 的运行组1234567。

```
aws omics delete-run-group \  
  --id 1234567
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRunGroup](#)中的。

## delete-run

以下代码示例演示了如何使用 delete-run。

### AWS CLI

要删除工作流程，请运行

以下delete-run示例删除带有 ID 的运行1234567。

```
aws omics delete-run \  
  --id 1234567
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRun](#)中的。

## delete-sequence-store

以下代码示例演示了如何使用 delete-sequence-store。

### AWS CLI

删除序列存储

以下delete-sequence-store示例删除 ID 为 ID 的序列存储1234567890。

```
aws omics delete-sequence-store \  
  --id 1234567890
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSequenceStore](#)中的。

## delete-share

以下代码示例演示了如何使用 delete-share。

### AWS CLI

删除共享的 HealthOmics 分析数据

以下delete-share示例删除了跨账户共享的分析数据。

```
aws omics delete-share \  
  --share-id "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a"
```

输出：

```
{  
  "status": "DELETING"  
}
```

有关更多信息，请参阅AWS HealthOmics 用户指南中的[跨账户共享](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteShare](#)中的。

## delete-variant-store

以下代码示例演示了如何使用 delete-variant-store。

### AWS CLI

#### 删除多属性商店

以下delete-variant-store示例删除名为的变体存储my\_var\_store。

```
aws omics delete-variant-store \  
  --name my_var_store
```

输出：

```
{  
  "status": "DELETING"  
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analyt ics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVariantStore](#)中的。

## delete-workflow

以下代码示例演示了如何使用 delete-workflow。

## AWS CLI

### 删除工作流程

以下delete-workflow示例删除了带有 ID 的工作流程1234567。

```
aws omics delete-workflow \  
  --id 1234567
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteWorkflow](#)中的。

## get-annotation-import-job

以下代码示例演示了如何使用 get-annotation-import-job。

## AWS CLI

### 查看注释导入作业

以下get-annotation-import-job示例获取有关注释导入任务的详细信息。

```
aws omics get-annotation-import-job \  
  --job-id 984162c7-xmpl-4d23-ab47-286f7950bfbf
```

输出：

```
{  
  "creationTime": "2022-11-30T01:40:11.017746Z",  
  "destinationName": "tsv_ann_store",  
  "id": "984162c7-xmpl-4d23-ab47-286f7950bfbf",  
  "items": [  
    {  
      "jobStatus": "COMPLETED",  
      "source": "s3://omics-artifacts-01d6xmpl4e72dd32/targetedregions.bed.gz"  
    }  
  ],  
  "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-serviceRole-  
W801XMPL7QZ",  
  "runLeftNormalization": false,  
  "status": "COMPLETED",  
  "updateTime": "2022-11-30T01:42:39.134009Z"
```

```
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analytics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAnnotationImportJob](#)中的。

## get-annotation-store-version

以下代码示例演示了如何使用 `get-annotation-store-version`。

### AWS CLI

检索注解存储版本的元数据

以下`get-annotation-store-version`示例检索所请求的注释存储版本的元数据。

```
aws omics get-annotation-store-version \  
  --name my_annotation_store \  
  --version-name my_version
```

输出：

```
{  
  "storeId": "4934045d1c6d",  
  "id": "2a3f4a44aa7b",  
  "status": "ACTIVE",  
  "versionArn": "arn:aws:omics:us-west-2:555555555555:annotationStore/  
my_annotation_store/version/my_version",  
  "name": "my_annotation_store",  
  "versionName": "my_version",  
  "creationTime": "2023-07-21T17:15:49.251040+00:00",  
  "updateTime": "2023-07-21T17:15:56.434223+00:00",  
  "statusMessage": "",  
  "versionSizeBytes": 0  
}
```

有关更多信息，请参阅AWS HealthOmics 用户指南中的[创建注释存储库的新版本](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAnnotationStoreVersion](#)中的。

## get-annotation-store

以下代码示例演示了如何使用 `get-annotation-store`。



## AWS CLI

### 查看注释存储库

以下`get-annotation-store`示例获取有关名为的注释存储的详细信息`my_ann_store`。

```
aws omics get-annotation-store \  
  --name my_ann_store
```

输出：

```
{  
  "creationTime": "2022-11-23T22:48:39.226492Z",  
  "id": "0a91xmplc71f",  
  "name": "my_ann_store",  
  "reference": {  
    "referenceArn": "arn:aws:omics:us-west-2:123456789012:referenceStore/1234567890/reference/1234567890"  
  },  
  "status": "CREATING",  
  "storeArn": "arn:aws:omics:us-west-2:123456789012:annotationStore/my_ann_store",  
  "storeFormat": "VCF",  
  "storeSizeBytes": 0,  
  "tags": {}  
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analyt ics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAnnotationStore](#)中的。

## get-read-set-activation-job

以下代码示例演示了如何使用 `get-read-set-activation-job`。

### AWS CLI

#### 查看读取集激活作业

以下`get-read-set-activation-job`示例获取有关读取集激活作业的详细信息。

```
aws omics get-read-set-activation-job \  
  --sequence-store-id 1234567890 \  
  --id 1234567890
```

输出：

```
{
  "completionTime": "2022-12-06T22:33:42.828Z",
  "creationTime": "2022-12-06T22:32:45.213Z",
  "id": "1234567890",
  "sequenceStoreId": "1234567890",
  "sources": [
    {
      "readSetId": "1234567890",
      "status": "FINISHED",
      "statusMessage": "No activation needed as read set is already in
ACTIVATING or ACTIVE state."
    }
  ],
  "status": "COMPLETED",
  "statusMessage": "The job completed successfully."
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetReadSetActivationJob](#) 中的。

## get-read-set-export-job

以下代码示例演示了如何使用 `get-read-set-export-job`。

AWS CLI

查看读取集导出作业

以下 `get-read-set-export-job` 示例获取有关读取集导出任务的详细信息。

```
aws omics get-read-set-export-job \
  --sequence-store-id 1234567890 \
  --id 1234567890
```

输出：

```
{
  "completionTime": "2022-12-06T22:39:14.491Z",
  "creationTime": "2022-12-06T22:37:18.612Z",
  "destination": "s3://omics-artifacts-01d6xmpl4e72dd32/read-set-export/",
}
```

```
"id": "1234567890",
"sequenceStoreId": "1234567890",
"status": "COMPLETED",
"statusMessage": "The job is submitted and will start soon."
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetReadSetExportJob](#) 中的。

## get-read-set-import-job

以下代码示例演示了如何使用 `get-read-set-import-job`。

### AWS CLI

查看读取集导入作业

以下 `get-read-set-import-job` 示例获取有关读取集导入任务的详细信息。

```
aws omics get-read-set-import-job \
  --sequence-store-id 1234567890 \
  --id 1234567890
```

输出：

```
{
  "creationTime": "2022-11-23T01:36:38.158Z",
  "id": "1234567890",
  "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-serviceRole-
W801XMPL7QZ",
  "sequenceStoreId": "1234567890",
  "sources": [
    {
      "name": "HG00100",
      "referenceArn": "arn:aws:omics:us-
west-2:123456789012:referenceStore/1234567890/reference/1234567890",
      "sampleId": "bam-sample",
      "sourceFileType": "BAM",
      "sourceFiles": {
        "source1": "s3://omics-artifacts-01d6xmpl4e72dd32/
HG00100.chrom20.ILLUMINA.bwa.GBR.low_coverage.20101123.bam",
        "source2": ""
      }
    }
  ]
}
```

```
    },
    "status": "IN_PROGRESS",
    "statusMessage": "The source job is currently in progress.",
    "subjectId": "bam-subject",
    "tags": {
      "aws:omics:sampleId": "bam-sample",
      "aws:omics:subjectId": "bam-subject"
    }
  },
  {
    "name": "HG00146",
    "referenceArn": "arn:aws:omics:us-
west-2:123456789012:referenceStore/1234567890/reference/1234567890",
    "sampleId": "fastq-sample",
    "sourceFileType": "FASTQ",
    "sourceFiles": {
      "source1": "s3://omics-artifacts-01d6xmpl4e72dd32/
SRR233106_1.filt.fastq.gz",
      "source2": "s3://omics-artifacts-01d6xmpl4e72dd32/
SRR233106_2.filt.fastq.gz"
    },
    "status": "IN_PROGRESS",
    "statusMessage": "The source job is currently in progress.",
    "subjectId": "fastq-subject",
    "tags": {
      "aws:omics:sampleId": "fastq-sample",
      "aws:omics:subjectId": "fastq-subject"
    }
  },
  {
    "name": "HG00096",
    "referenceArn": "arn:aws:omics:us-
west-2:123456789012:referenceStore/1234567890/reference/1234567890",
    "sampleId": "cram-sample",
    "sourceFileType": "CRAM",
    "sourceFiles": {
      "source1": "s3://omics-artifacts-01d6xmpl4e72dd32/
HG00096.alt_bwamem_GRCh38DH.20150718.GBR.low_coverage.cram",
      "source2": ""
    },
    "status": "IN_PROGRESS",
    "statusMessage": "The source job is currently in progress.",
    "subjectId": "cram-subject",
    "tags": {
```

```
        "aws:omics:sampleId": "cram-sample",
        "aws:omics:subjectId": "cram-subject"
      }
    ],
    "status": "IN_PROGRESS",
    "statusMessage": "The job is currently in progress."
  }
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetReadSetImportJob](#)中的。

## get-read-set-metadata

以下代码示例演示了如何使用 get-read-set-metadata。

### AWS CLI

#### 查看已读集

以下get-read-set-metadata示例获取有关读取集文件的详细信息。

```
aws omics get-read-set-metadata \
  --sequence-store-id 1234567890 \
  --id 1234567890
```

输出：

```
{
  "arn": "arn:aws:omics:us-west-2:123456789012:sequenceStore/1234567890/
readSet/1234567890",
  "creationTime": "2022-11-23T21:55:00.515Z",
  "fileType": "FASTQ",
  "files": {
    "source1": {
      "contentLength": 310054739,
      "partSize": 104857600,
      "totalParts": 3
    },
    "source2": {
      "contentLength": 307846621,
```

```
        "partSize": 104857600,
        "totalParts": 3
      }
    },
    "id": "1234567890",
    "name": "HG00146",
    "referenceArn": "arn:aws:omics:us-west-2:123456789012:referenceStore/1234567890/
reference/1234567890",
    "sampleId": "fastq-sample",
    "sequenceInformation": {
      "alignment": "UNALIGNED",
      "totalBaseCount": 677717384,
      "totalReadCount": 8917334
    },
    "sequenceStoreId": "1234567890",
    "status": "ACTIVE",
    "subjectId": "fastq-subject"
  }
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetReadSetMetadata](#)中的。

## get-read-set

以下代码示例演示了如何使用 get-read-set。

### AWS CLI

#### 下载读取集

以下get-read-set示例将读取集的第 3 部分下载为1234567890.3.bam。

```
aws omics get-read-set \
  --sequence-store-id 1234567890 \
  --id 1234567890 \
  --part-number 3 1234567890.3.bam
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetReadSet](#)中的。

## get-reference-import-job

以下代码示例演示了如何使用 `get-reference-import-job`。

### AWS CLI

查看参考文献导入作业

以下 `get-reference-import-job` 示例获取有关引用导入任务的详细信息。

```
aws omics get-reference-import-job \  
  --reference-store-id 1234567890 \  
  --id 1234567890
```

输出：

```
{  
  "creationTime": "2022-11-22T22:25:41.124Z",  
  "id": "1234567890",  
  "referenceStoreId": "1234567890",  
  "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-serviceRole-  
W801XMPL7QZ",  
  "sources": [  
    {  
      "name": "assembly-38",  
      "sourceFile": "s3://omics-artifacts-01d6xmpl4e72dd32/  
Homo_sapiens_assembly38.fasta",  
      "status": "IN_PROGRESS",  
      "statusMessage": "The source job is currently in progress."  
    }  
  ],  
  "status": "IN_PROGRESS",  
  "statusMessage": "The job is currently in progress."  
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetReferenceImportJob](#) 中的。

## get-reference-metadata

以下代码示例演示了如何使用 `get-reference-metadata`。

## AWS CLI

### 查看参考文献

以下`get-reference-metadata`示例获取有关参考文献的详细信息。

```
aws omics get-reference-metadata \  
  --reference-store-id 1234567890 \  
  --id 1234567890
```

输出：

```
{  
  "arn": "arn:aws:omics:us-west-2:123456789012:referenceStore/1234567890/  
reference/1234567890",  
  "creationTime": "2022-11-22T22:27:09.033Z",  
  "files": {  
    "index": {  
      "contentLength": 160928,  
      "partSize": 104857600,  
      "totalParts": 1  
    },  
    "source": {  
      "contentLength": 3249912778,  
      "partSize": 104857600,  
      "totalParts": 31  
    }  
  },  
  "id": "1234567890",  
  "md5": "7ff134953dcca8c8997453bbb80b6b5e",  
  "name": "assembly-38",  
  "referenceStoreId": "1234567890",  
  "status": "ACTIVE",  
  "updateTime": "2022-11-22T22:27:09.033Z"  
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetReferenceMetadata](#)中的。

## get-reference-store

以下代码示例演示了如何使用 `get-reference-store`。



## AWS CLI

### 查看参考资料库

以下`get-reference-store`示例获取有关参考存储的详细信息。

```
aws omics get-reference-store \  
  --id 1234567890
```

输出：

```
{  
  "arn": "arn:aws:omics:us-west-2:123456789012:referenceStore/1234567890",  
  "creationTime": "2022-09-23T23:27:20.364Z",  
  "id": "1234567890",  
  "name": "my-rstore-0"  
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetReferenceStore](#)中的。

## get-reference

以下代码示例演示了如何使用 `get-reference`。

## AWS CLI

### 下载基因组参考文献

以下`get-reference`示例将基因组的第 1 部分下载为`hg38.1.fa`。

```
aws omics get-reference \  
  --reference-store-id 1234567890 \  
  --id 1234567890 \  
  --part-number 1 hg38.1.fa
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetReference](#)中的。

## get-run-group

以下代码示例演示了如何使用 `get-run-group`。

### AWS CLI

#### 查看跑步组

以下`get-run-group`示例获取有关跑步组的详细信息。

```
aws omics get-run-group \  
  --id 1234567
```

输出：

```
{  
  "arn": "arn:aws:omics:us-west-2:123456789012:runGroup/1234567",  
  "creationTime": "2022-12-01T00:58:42.915219Z",  
  "id": "1234567",  
  "maxCpus": 20,  
  "maxDuration": 600,  
  "name": "cram-convert",  
  "tags": {}  
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRunGroup](#)中的。

## get-run-task

以下代码示例演示了如何使用 `get-run-task`。

### AWS CLI

#### 查看任务

以下`get-run-task`示例获取有关工作流程任务的详细信息。

```
aws omics get-run-task \  
  --id 1234567 \  
  --task-id 1234567
```

输出：

```
{
  "cpus": 1,
  "creationTime": "2022-11-30T23:13:00.718651Z",
  "logStream": "arn:aws:logs:us-west-2:123456789012:log-group:/aws/omics/WorkflowLog:log-stream:run/1234567/task/1234567",
  "memory": 15,
  "name": "CramToBamTask",
  "startTime": "2022-11-30T23:17:47.016Z",
  "status": "COMPLETED",
  "stopTime": "2022-11-30T23:18:21.503Z",
  "taskId": "1234567"
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetRunTask](#) 中的。

## get-run

以下代码示例演示了如何使用 `get-run`。

### AWS CLI

要查看工作流程，请运行

以下 `get-run` 示例获取有关工作流程运行的详细信息。

```
aws omics get-run \
  --id 1234567
```

输出：

```
{
  "arn": "arn:aws:omics:us-west-2:123456789012:run/1234567",
  "creationTime": "2022-11-30T22:58:22.615865Z",
  "digest":
  "sha256:c54bxmpl742dcc26f7fa1f10e37550ddd8f251f418277c0a58e895b801ed28cf",
  "id": "1234567",
  "name": "cram-to-bam",
  "outputUri": "s3://omics-artifacts-01d6xmpl4e72dd32/workflow-output/",
  "parameters": {
```

```

    "ref_dict": "s3://omics-artifacts-01d6xmpl4e72dd32/
Homo_sapiens_assembly38.dict",
    "ref_fasta_index": "s3://omics-artifacts-01d6xmpl4e72dd32/
Homo_sapiens_assembly38.fasta.fai",
    "ref_fasta": "s3://omics-artifacts-01d6xmpl4e72dd32/
Homo_sapiens_assembly38.fasta",
    "sample_name": "NA12878",
    "input_cram": "s3://omics-artifacts-01d6xmpl4e72dd32/NA12878.cram"
  },
  "resourceDigests": {
    "s3://omics-artifacts-01d6xmpl4e72dd32/Homo_sapiens_assembly38.fasta.fai":
"etag:f76371b113734a56cde236bc0372de0a",
    "s3://omics-artifacts-01d6xmpl4e72dd32/Homo_sapiens_assembly38.dict":
"etag:3884c62eb0e53fa92459ed9bfff133ae6",
    "s3://omics-artifacts-01d6xmpl4e72dd32/Homo_sapiens_assembly38.fasta":
"etag:e307d81c605fb91b7720a08f00276842-388",
    "s3://omics-artifacts-01d6xmpl4e72dd32/NA12878.cram":
"etag:a9f52976381286c6143b5cc681671ec6"
  },
  "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-serviceRole-
W801XMPL7QZ",
  "startedBy": "arn:aws:iam::123456789012:user/laptop-2020",
  "status": "STARTING",
  "tags": {},
  "workflowId": "1234567",
  "workflowType": "PRIVATE"
}

```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRun](#)中的。

## get-sequence-store

以下代码示例演示了如何使用 get-sequence-store。

### AWS CLI

#### 查看序列存储

以下get-sequence-store示例获取有关带有 ID 的序列存储的详细信息1234567890。

```
aws omics get-sequence-store \
```

```
--id 1234567890
```

输出：

```
{
  "arn": "arn:aws:omics:us-east-1:123456789012:sequenceStore/1234567890",
  "creationTime": "2022-11-23T19:55:48.376Z",
  "id": "1234567890",
  "name": "my-seq-store"
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSequenceStore](#)中的。

## get-share

以下代码示例演示了如何使用 get-share。

### AWS CLI

检索有关 HealthOmics 分析数据份额的元数据

以下get-share示例检索跨账户共享的分析数据的元数据。

```
aws omics get-share \
  --share-id "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a"
```

输出：

```
{
  "share": {
    "shareId": "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a",
    "name": "my_Share-123",
    "resourceArn": "arn:aws:omics:us-west-2:555555555555:variantStore/omics_dev_var_store",
    "principalSubscriber": "123456789012",
    "ownerId": "555555555555",
    "status": "PENDING"
  }
}
```

```
}
```

有关更多信息，请参阅AWS HealthOmics 用户指南中的[跨账户共享](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetShare](#)中的。

## get-variant-import-job

以下代码示例演示了如何使用 get-variant-import-job。

### AWS CLI

#### 查看变体导入任务

以下get-variant-import-job示例获取有关变体导入任务的详细信息。

```
aws omics get-variant-import-job \  
  --job-id edd7b8ce-xmpl-47e2-bc99-258cac95a508
```

输出：

```
{  
  "creationTime": "2022-11-23T22:42:50.037812Z",  
  "destinationName": "my_var_store",  
  "id": "edd7b8ce-xmpl-47e2-bc99-258cac95a508",  
  "items": [  
    {  
      "jobStatus": "IN_PROGRESS",  
      "source": "s3://omics-artifacts-01d6xmpl4e72dd32/  
Homo_sapiens_assembly38.known_indels.vcf.gz"  
    }  
  ],  
  "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-serviceRole-  
W801XMPL7QZ",  
  "runLeftNormalization": false,  
  "status": "IN_PROGRESS",  
  "updateTime": "2022-11-23T22:43:05.898309Z"  
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analyt ics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVariantImportJob](#)中的。

## get-variant-store

以下代码示例演示了如何使用 `get-variant-store`。

### AWS CLI

#### 查看多属性商店

以下`get-variant-store`示例获取有关变体商店的详细信息。

```
aws omics get-variant-store \  
  --name my_var_store
```

输出：

```
{  
  "creationTime": "2022-11-23T22:09:07.534499Z",  
  "id": "02dexmplcfdd",  
  "name": "my_var_store",  
  "reference": {  
    "referenceArn": "arn:aws:omics:us-  
west-2:123456789012:referenceStore/1234567890/reference/1234567890"  
  },  
  "status": "CREATING",  
  "storeArn": "arn:aws:omics:us-west-2:123456789012:variantStore/my_var_store",  
  "storeSizeBytes": 0,  
  "tags": {},  
  "updateTime": "2022-11-23T22:09:24.931711Z"  
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analyt ics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVariantStore](#)中的。

## get-workflow

以下代码示例演示了如何使用 `get-workflow`。

### AWS CLI

#### 查看工作流程

以下`get-workflow`示例获取有关带有 ID 的工作流的详细信息1234567。

```
aws omics get-workflow \  
  --id 1234567
```

输出：

```
{  
  "arn": "arn:aws:omics:us-west-2:123456789012:workflow/1234567",  
  "creationTime": "2022-11-30T22:33:16.225368Z",  
  "digest":  
  "sha256:c54bxmpl742dcc26f7fa1f10e37550ddd8f251f418277c0a58e895b801ed28cf",  
  "engine": "WDL",  
  "id": "1234567",  
  "main": "workflow-crambam.wdl",  
  "name": "cram-converter",  
  "parameterTemplate": {  
    "ref_dict": {  
      "description": "dictionary file for 'ref_fasta'"  
    },  
    "ref_fasta_index": {  
      "description": "Index of the reference genome fasta file"  
    },  
    "ref_fasta": {  
      "description": "Reference genome fasta file"  
    },  
    "input_cram": {  
      "description": "The Cram file to convert to BAM"  
    },  
    "sample_name": {  
      "description": "The name of the input sample, used to name the output  
BAM"  
    }  
  },  
  "status": "ACTIVE",  
  "statusMessage": "workflow-crambam.wdl\  
  workflow CramToBamFlow\  
call CramToBamTask\  
  call ValidateSamFile\  
  task CramToBamTask\  
  task  
ValidateSamFile\  
",  
  "tags": {},  
  "type": "PRIVATE"  
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetWorkflow](#)中的。

## list-annotation-import-jobs

以下代码示例演示了如何使用 list-annotation-import-jobs。

### AWS CLI

获取注释导入任务列表

以下list-annotation-import-jobs是注释导入任务的列表。

```
aws omics list-annotation-import-jobs
```

输出：

```
{
  "annotationImportJobs": [
    {
      "creationTime": "2022-11-30T01:39:41.478294Z",
      "destinationName": "gff_ann_store",
      "id": "18a9e792-xmpl-4869-a105-e5b602900444",
      "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-
serviceRole-W801XMPL7QZ",
      "runLeftNormalization": false,
      "status": "COMPLETED",
      "updateTime": "2022-11-30T01:47:09.145178Z"
    },
    {
      "creationTime": "2022-11-30T00:45:58.007838Z",
      "destinationName": "my_ann_store",
      "id": "4e9eafc8-xmpl-431e-a0b2-3bda27cb600a",
      "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-
serviceRole-W801XMPL7QZ",
      "runLeftNormalization": false,
      "status": "FAILED",
      "updateTime": "2022-11-30T00:47:01.706325Z"
    }
  ]
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analytics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAnnotationImportJobs](#)中的。

## list-annotation-store-versions

以下代码示例演示了如何使用 list-annotation-store-versions。

### AWS CLI

列出注释库的所有版本。

以下list-annotation-store-versions示例列出了注释存储库的所有现有版本。

```
aws omics list-annotation-store-versions \  
  --name my_annotation_store
```

输出：

```
{  
  "annotationStoreVersions": [  
    {  
      "storeId": "4934045d1c6d",  
      "id": "2a3f4a44aa7b",  
      "status": "CREATING",  
      "versionArn": "arn:aws:omics:us-west-2:555555555555:annotationStore/  
my_annotation_store/version/my_version_2",  
      "name": "my_annotation_store",  
      "versionName": "my_version_2",  
      "creationTime": "2023-07-21T17:20:59.380043+00:00",  
      "versionSizeBytes": 0  
    },  
    {  
      "storeId": "4934045d1c6d",  
      "id": "4934045d1c6d",  
      "status": "ACTIVE",  
      "versionArn": "arn:aws:omics:us-west-2:555555555555:annotationStore/  
my_annotation_store/version/my_version_1",  
      "name": "my_annotation_store",  
      "versionName": "my_version_1",  
      "creationTime": "2023-07-21T17:15:49.251040+00:00",  
      "updateTime": "2023-07-21T17:15:56.434223+00:00",  
      "statusMessage": "",  
      "versionSizeBytes": 0  
    }  
  ]  
}
```

```
    }  
  }  
}
```

有关更多信息，请参阅AWS HealthOmics 用户指南中的[创建注释存储库的新版本](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAnnotationStoreVersions](#)中的。

## list-annotation-stores

以下代码示例演示了如何使用 list-annotation-stores。

### AWS CLI

获取注释存储库列表

以下list-annotation-stores示例获取注释存储列表。

```
aws omics list-annotation-stores
```

输出：

```
{  
  "annotationStores": [  
    {  
      "creationTime": "2022-11-23T22:48:39.226492Z",  
      "id": "0a91xmplc71f",  
      "name": "my_ann_store",  
      "reference": {  
        "referenceArn": "arn:aws:omics:us-west-2:123456789012:referenceStore/1234567890/reference/1234567890"  
      },  
      "status": "ACTIVE",  
      "statusMessage": "",  
      "storeArn": "arn:aws:omics:us-west-2:123456789012:annotationStore/my_ann_store",  
      "storeFormat": "VCF",  
      "storeSizeBytes": 0,  
      "updateTime": "2022-11-23T22:53:27.372840Z"  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analytics](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAnnotationStores](#) 中的。

## list-multipart-read-set-uploads

以下代码示例演示了如何使用 `list-multipart-read-set-uploads`。

### AWS CLI

要列出所有分段读取设置上传及其状态。

以下 `list-multipart-read-set-uploads` 示例列出了所有分段读取集上传及其状态。

```
aws omics list-multipart-read-set-uploads \  
  --sequence-store-id 0123456789
```

输出：

```
{  
  "uploads":  
    [  
      {  
        "sequenceStoreId": "0123456789",  
        "uploadId": "8749584421",  
        "sourceFileType": "FASTQ",  
        "subjectId": "mySubject",  
        "sampleId": "mySample",  
        "generatedFrom": "1000 Genomes",  
        "name": "HG00146",  
        "description": "FASTQ for HG00146",  
        "creationTime": "2023-11-29T19:22:51.349298+00:00"  
      },  
      {  
        "sequenceStoreId": "0123456789",  
        "uploadId": "5290538638",  
        "sourceFileType": "BAM",  
        "subjectId": "mySubject",  
        "sampleId": "mySample",  
        "generatedFrom": "1000 Genomes",  
        "referenceArn": "arn:aws:omics:us-  
west-2:845448930428:referenceStore/8168613728/reference/2190697383",  
        "name": "HG00146",
```

```

        "description": "BAM for HG00146",
        "creationTime": "2023-11-29T19:23:33.116516+00:00"
    },
    {
        "sequenceStoreId": "0123456789",
        "uploadId": "4174220862",
        "sourceFileType": "BAM",
        "subjectId": "mySubject",
        "sampleId": "mySample",
        "generatedFrom": "1000 Genomes",
        "referenceArn": "arn:aws:omics:us-
west-2:845448930428:referenceStore/8168613728/reference/2190697383",
        "name": "HG00147",
        "description": "BAM for HG00147",
        "creationTime": "2023-11-29T19:23:47.007866+00:00"
    }
]
}

```

有关更多信息，请参阅《AWS HealthOmics 用户指南》中的[直接上传到序列存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListMultipartReadSetUploads](#) 中的。

## list-read-set-activation-jobs

以下代码示例演示了如何使用 `list-read-set-activation-jobs`。

### AWS CLI

获取读集激活任务列表

以下 `list-read-set-activation-jobs` 示例获取 id 为序列存储的激活任务列表 1234567890。

```
aws omics list-read-set-activation-jobs \
  --sequence-store-id 1234567890
```

输出：

```
{
  "activationJobs": [
    {
```

```
    "completionTime": "2022-12-06T22:33:42.828Z",
    "creationTime": "2022-12-06T22:32:45.213Z",
    "id": "1234567890",
    "sequenceStoreId": "1234567890",
    "status": "COMPLETED"
  },
  {
    "creationTime": "2022-12-06T22:35:10.100Z",
    "id": "1234567890",
    "sequenceStoreId": "1234567890",
    "status": "IN_PROGRESS"
  }
]
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListReadSetActivationJobs](#) 中的。

## list-read-set-export-jobs

以下代码示例演示了如何使用 `list-read-set-export-jobs`。

### AWS CLI

获取读取集导出任务列表

以下 `list-read-set-export-jobs` 示例获取 id 为序列存储的导出任务列表 1234567890。

```
aws omics list-read-set-export-jobs \
  --sequence-store-id 1234567890
```

输出：

```
{
  "exportJobs": [
    {
      "completionTime": "2022-12-06T22:39:14.491Z",
      "creationTime": "2022-12-06T22:37:18.612Z",
      "destination": "s3://omics-artifacts-01d6xmpl4e72dd32/read-set-export/",
      "id": "1234567890",
      "sequenceStoreId": "1234567890",
      "status": "COMPLETED"
    }
  ]
}
```

```
    },
    {
      "creationTime": "2022-12-06T22:38:04.871Z",
      "destination": "s3://omics-artifacts-01d6xmpl4e72dd32/read-set-export/",
      "id": "1234567890",
      "sequenceStoreId": "1234567890",
      "status": "IN_PROGRESS"
    }
  ]
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListReadSetExportJobs](#) 中的。

## list-read-set-import-jobs

以下代码示例演示了如何使用 `list-read-set-import-jobs`。

### AWS CLI

获取读取集导入任务列表

以下 `list-read-set-import-jobs` 示例获取 id 为序列存储的导入任务列表 1234567890。

```
aws omics list-read-set-import-jobs \
  --sequence-store-id 1234567890
```

输出：

```
{
  "importJobs": [
    {
      "completionTime": "2022-11-29T18:17:49.244Z",
      "creationTime": "2022-11-29T17:32:47.700Z",
      "id": "1234567890",
      "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-
serviceRole-W801XMPL7QZ",
      "sequenceStoreId": "1234567890",
      "status": "COMPLETED"
    },
    {
      "completionTime": "2022-11-23T22:01:34.090Z",
```

```
    "creationTime": "2022-11-23T21:52:43.289Z",
    "id": "1234567890",
    "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-
serviceRole-W801XMPL7QZ",
    "sequenceStoreId": "1234567890",
    "status": "COMPLETED_WITH_FAILURES"
  }
]
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListReadSetImportJobs](#) 中的。

## list-read-set-upload-parts

以下代码示例演示了如何使用 `list-read-set-upload-parts`。

### AWS CLI

列出序列存储请求的分段上传中的所有分段。

以下 `list-read-set-upload-parts` 示例列出了为序列存储请求的分段上传中的所有分段。

```
aws omics list-read-set-upload-parts \
  --sequence-store-id 0123456789 \
  --upload-id 1122334455 \
  --part-source SOURCE1
```

输出：

```
{
  "parts": [
    {
      "partNumber": 1,
      "partSize": 94371840,
      "file": "SOURCE1",
      "checksum":
"984979b9928ae8d8622286c4a9cd8e99d964a22d59ed0f5722e1733eb280e635",
      "lastUpdatedTime": "2023-02-02T20:14:47.533000+00:00"
    }
  ]
}
```



```
        "partSize": 10471840,
        "file": "SOURCE1",
        "checksum":
"984979b9928ae8d8622286c4a9cd8e99d964a22d59ed0f5722e1733eb280e635",
        "lastUpdatedTime": "2023-02-02T20:14:47.533000+00:00"
    }
]
}
```

有关更多信息，请参阅《AWS HealthOmics 用户指南》中的[直接上传到序列存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListReadSetUploadParts](#) 中的。

## list-read-sets

以下代码示例演示了如何使用 list-read-sets。

### AWS CLI

获取读取集列表

以下 list-read-sets 示例获取 id 为序列存储的读取集列表 1234567890。

```
aws omics list-read-sets \
  --sequence-store-id 1234567890
```

输出：

```
{
  "readSets": [
    {
      "arn": "arn:aws:omics:us-west-2:123456789012:sequenceStore/1234567890/
readSet/1234567890",
      "creationTime": "2022-11-23T21:55:00.515Z",
      "fileType": "FASTQ",
      "id": "1234567890",
      "name": "HG00146",
      "referenceArn": "arn:aws:omics:us-
west-2:123456789012:referenceStore/1234567890/reference/1234567890",
      "sampleId": "fastq-sample",
      "sequenceStoreId": "1234567890",
      "status": "ACTIVE",
```

```
        "subjectId": "fastq-subject"
      }
    ]
  }
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListReadSets](#)中的。

## list-reference-import-jobs

以下代码示例演示了如何使用 `list-reference-import-jobs`。

### AWS CLI

获取参考导入任务列表

以下`list-reference-import-jobs`示例获取了 ID 为 ID 的参考存储的参考文献导入任务列表1234567890。

```
aws omics list-reference-import-jobs \
  --reference-store-id 1234567890
```

输出：

```
{
  "importJobs": [
    {
      "completionTime": "2022-11-23T19:54:58.204Z",
      "creationTime": "2022-11-23T19:53:20.729Z",
      "id": "1234567890",
      "referenceStoreId": "1234567890",
      "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-
serviceRole-W801XMPL7QZ",
      "status": "COMPLETED"
    },
    {
      "creationTime": "2022-11-23T20:34:03.250Z",
      "id": "1234567890",
      "referenceStoreId": "1234567890",
      "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-
serviceRole-W801XMPL7QZ",

```

```
        "status": "IN_PROGRESS"
      }
    ]
  }
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListReferenceImportJobs](#)中的。

## list-reference-stores

以下代码示例演示了如何使用 list-reference-stores。

### AWS CLI

获取参考商店列表

以下list-reference-stores示例获取参考商店列表。

```
aws omics list-reference-stores
```

输出：

```
{
  "referenceStores": [
    {
      "arn": "arn:aws:omics:us-west-2:123456789012:referenceStore/1234567890",
      "creationTime": "2022-11-22T22:13:25.947Z",
      "id": "1234567890",
      "name": "my-ref-store"
    }
  ]
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListReferenceStores](#)中的。

## list-references

以下代码示例演示了如何使用 list-references。

## AWS CLI

### 获取参考文献清单

以下`list-references`示例获取标识为 `id` 的参考库的基因组参考文献列表1234567890。

```
aws omics list-references \  
  --reference-store-id 1234567890
```

输出：

```
{  
  "references": [  
    {  
      "arn": "arn:aws:omics:us-west-2:123456789012:referenceStore/1234567890/  
reference/1234567890",  
      "creationTime": "2022-11-22T22:27:09.033Z",  
      "id": "1234567890",  
      "md5": "7ff134953dcca8c8997453bbb80b6b5e",  
      "name": "assembly-38",  
      "referenceStoreId": "1234567890",  
      "status": "ACTIVE",  
      "updateTime": "2022-11-22T22:27:09.033Z"  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

• 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListReferences](#)中的。

## list-run-groups

以下代码示例演示了如何使用 `list-run-groups`。

### AWS CLI

#### 获取跑步组列表

以下`list-run-groups`示例获取了运行组列表。

```
aws omics list-run-groups
```

输出：

```
{
  "items": [
    {
      "arn": "arn:aws:omics:us-west-2:123456789012:runGroup/1234567",
      "creationTime": "2022-12-01T00:58:42.915219Z",
      "id": "1234567",
      "maxCpus": 20,
      "maxDuration": 600,
      "name": "cram-convert"
    }
  ]
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRunGroups](#)中的。

## list-run-tasks

以下代码示例演示了如何使用 list-run-tasks。

### AWS CLI

获取任务清单

以下list-run-tasks示例获取工作流程运行的任务列表。

```
aws omics list-run-tasks \
  --id 1234567
```

输出：

```
{
  "items": [
    {
      "cpus": 1,
      "creationTime": "2022-11-30T23:13:00.718651Z",
      "memory": 15,
      "name": "CramToBamTask",
      "startTime": "2022-11-30T23:17:47.016Z",
      "status": "COMPLETED",
    }
  ]
}
```

```
    "stopTime": "2022-11-30T23:18:21.503Z",
    "taskId": "1234567"
  },
  {
    "cpus": 1,
    "creationTime": "2022-11-30T23:18:32.315606Z",
    "memory": 4,
    "name": "ValidateSamFile",
    "startTime": "2022-11-30T23:23:40.165Z",
    "status": "COMPLETED",
    "stopTime": "2022-11-30T23:24:14.766Z",
    "taskId": "1234567"
  }
]
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListRunTasks](#) 中的。

## list-runs

以下代码示例演示了如何使用 `list-runs`。

### AWS CLI

获取工作流程运行列表

以下 `list-runs` 示例获取工作流程运行列表。

```
aws omics list-runs
```

输出：

```
{
  "items": [
    {
      "arn": "arn:aws:omics:us-west-2:123456789012:run/1234567",
      "creationTime": "2022-12-02T23:20:01.202074Z",
      "id": "1234567",
      "name": "cram-to-bam",
      "priority": 1,
      "startTime": "2022-12-02T23:29:18.115Z",
```

```
    "status": "COMPLETED",
    "stopTime": "2022-12-02T23:57:54.428812Z",
    "storageCapacity": 10,
    "workflowId": "1234567"
  },
  {
    "arn": "arn:aws:omics:us-west-2:123456789012:run/1234567",
    "creationTime": "2022-12-03T00:16:57.180066Z",
    "id": "1234567",
    "name": "cram-to-bam",
    "priority": 1,
    "startTime": "2022-12-03T00:26:50.233Z",
    "status": "FAILED",
    "stopTime": "2022-12-03T00:37:21.451340Z",
    "storageCapacity": 10,
    "workflowId": "1234567"
  },
  {
    "arn": "arn:aws:omics:us-west-2:123456789012:run/1234567",
    "creationTime": "2022-12-05T17:57:08.444817Z",
    "id": "1234567",
    "name": "cram-to-bam",
    "status": "STARTING",
    "workflowId": "1234567"
  }
]
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListRuns](#) 中的。

## list-sequence-stores

以下代码示例演示了如何使用 `list-sequence-stores`。

### AWS CLI

获取序列存储列表

以下 `list-sequence-stores` 示例获取序列存储列表。

```
aws omics list-sequence-stores
```

输出：

```
{
  "sequenceStores": [
    {
      "arn": "arn:aws:omics:us-west-2:123456789012:sequenceStore/1234567890",
      "creationTime": "2022-11-23T01:24:33.629Z",
      "id": "1234567890",
      "name": "my-seq-store"
    }
  ]
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSequenceStores](#)中的。

## list-shares

以下代码示例演示了如何使用 list-shares。

### AWS CLI

列出 HealthOmics 分析数据的可用份额

以下list-shares示例列出了为资源所有者创建的所有共享。

```
aws omics list-shares \
  --resource-owner SELF
```

输出：

```
{
  "shares": [
    {
      "shareId": "595c1cbd-a008-4eca-a887-954d30c91c6e",
      "name": "myShare",
      "resourceArn": "arn:aws:omics:us-west-2:555555555555:variantStore/
store_1",
      "principalSubscriber": "123456789012",
      "ownerId": "555555555555",
      "status": "PENDING"
    }
  ]
}
```



```

    }
    {
      "shareId": "39b65d0d-4368-4a19-9814-b0e31d73c10a",
      "name": "myShare3456",
      "resourceArn": "arn:aws:omics:us-west-2:555555555555:variantStore/
store_2",
      "principalSubscriber": "123456789012",
      "ownerId": "555555555555",
      "status": "ACTIVE"
    },
    {
      "shareId": "203152f5-eef9-459d-a4e0-a691668d44ef",
      "name": "myShare4",
      "resourceArn": "arn:aws:omics:us-west-2:555555555555:variantStore/
store_3",
      "principalSubscriber": "123456789012",
      "ownerId": "555555555555",
      "status": "ACTIVE"
    }
  ]
}

```

有关更多信息，请参阅AWS HealthOmics 用户指南中的[跨账户共享](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListShares](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

#### 获取标签列表

以下`list-tags-for-resource`示例获取带有 id 的工作流的标签列表1234567。

```
aws omics list-tags-for-resource \
  --resource-arn arn:aws:omics:us-west-2:123456789012:workflow/1234567
```

#### 输出：

```
{
```

```
"tags": {
  "department": "analytics"
}
}
```

有关更多信息，请参阅《[亚马逊 Omics 开发者指南](#)》中的[在 Amazon Omics 中为资源添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## list-variant-import-jobs

以下代码示例演示了如何使用 `list-variant-import-jobs`。

### AWS CLI

获取变体导入任务列表

以下 `list-variant-import-jobs` 示例获取变体导入任务列表。

```
aws omics list-variant-import-jobs
```

输出：

```
{
  "variantImportJobs": [
    {
      "creationTime": "2022-11-23T22:47:02.514002Z",
      "destinationName": "my_var_store",
      "id": "69cb65d6-xmpl-4a4a-9025-4565794b684e",
      "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-
serviceRole-W801XMPL7QZ",
      "runLeftNormalization": false,
      "status": "COMPLETED",
      "updateTime": "2022-11-23T22:49:17.976597Z"
    },
    {
      "creationTime": "2022-11-23T22:42:50.037812Z",
      "destinationName": "my_var_store",
      "id": "edd7b8ce-xmpl-47e2-bc99-258cac95a508",
      "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-
serviceRole-W801XMPL7QZ",
      "runLeftNormalization": false,
      "status": "COMPLETED",

```

```

        "updateTime": "2022-11-23T22:45:26.009880Z"
      }
    ]
  }

```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analytics](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListVariantImportJobs](#) 中的。

## list-variant-stores

以下代码示例演示了如何使用 `list-variant-stores`。

### AWS CLI

获取变体商店列表

以下 `list-variant-stores` 示例获取变体商店列表。

```
aws omics list-variant-stores
```

输出：

```

{
  "variantStores": [
    {
      "creationTime": "2022-11-23T22:09:07.534499Z",
      "id": "02dexmplcfdd",
      "name": "my_var_store",
      "reference": {
        "referenceArn": "arn:aws:omics:us-west-2:123456789012:referenceStore/1234567890/reference/1234567890"
      },
      "status": "CREATING",
      "storeArn": "arn:aws:omics:us-west-2:123456789012:variantStore/my_var_store",
      "storeSizeBytes": 0,
      "updateTime": "2022-11-23T22:09:24.931711Z"
    },
    {
      "creationTime": "2022-09-23T23:00:09.140265Z",
      "id": "8777xmpl1a24",
      "name": "myvstore0",

```

```

        "status": "ACTIVE",
        "storeArn": "arn:aws:omics:us-west-2:123456789012:variantStore/
myvstore0",
        "storeSizeBytes": 0,
        "updateTime": "2022-09-23T23:03:26.013220Z"
    }
]
}

```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analyt ics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListVariantStores](#)中的。

## list-workflows

以下代码示例演示了如何使用 list-workflows。

### AWS CLI

获取工作流程列表

以下list-workflows示例获取工作流程列表。

```
aws omics list-workflows
```

输出：

```

{
  "items": [
    {
      "arn": "arn:aws:omics:us-west-2:123456789012:workflow/1234567",
      "creationTime": "2022-09-23T23:08:22.041227Z",
      "digest": "nSCNo/qMWFxmplXpUdokXJnwgne0axyyc2Y0xVxrJTE=",
      "id": "1234567",
      "name": "my-wkflow-0",
      "status": "ACTIVE",
      "type": "PRIVATE"
    },
    {
      "arn": "arn:aws:omics:us-west-2:123456789012:workflow/1234567",
      "creationTime": "2022-11-30T22:33:16.225368Z",
      "digest":
"sha256:c54bxmpl1742dcc26f7fa1f10e37550ddd8f251f418277c0a58e895b801ed28cf",

```

```
        "id": "1234567",
        "name": "cram-converter",
        "status": "ACTIVE",
        "type": "PRIVATE"
    }
]
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListWorkflows](#)中的。

## start-annotation-import-job

以下代码示例演示了如何使用 start-annotation-import-job。

### AWS CLI

#### 导入注释

以下start-annotation-import-job示例从 Amazon S3 导入注释。

```
aws omics start-annotation-import-job \
  --destination-name tsv_ann_store \
  --no-run-left-normalization \
  --role-arn arn:aws:iam::123456789012:role/omics-service-role-serviceRole-
W801XMPL7QZ \
  --items source=s3://omics-artifacts-01d6xmpl4e72dd32/targetedregions.bed.gz
```

#### 输出：

```
{
  "jobId": "984162c7-xmpl-4d23-ab47-286f7950bfbf"
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analyt ics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartAnnotationImportJob](#)中的。

## start-read-set-activation-job

以下代码示例演示了如何使用 start-read-set-activation-job。

## AWS CLI

### 激活已存档的读取集

以下 `start-read-set-activation-job` 示例激活了两个读取集。

```
aws omics start-read-set-activation-job \  
  --sequence-store-id 1234567890 \  
  --sources readSetId=1234567890 readSetId=1234567890
```

输出：

```
{  
  "creationTime": "2022-12-06T22:35:10.100Z",  
  "id": "1234567890",  
  "sequenceStoreId": "1234567890",  
  "status": "SUBMITTED"  
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartReadSetActivationJob](#) 中的。

## start-read-set-export-job

以下代码示例演示了如何使用 `start-read-set-export-job`。

## AWS CLI

### 导出读取集

以下 `start-read-set-export-job` 示例将两个读取集导出到 Amazon S3。

```
aws omics start-read-set-export-job \  
  --sequence-store-id 1234567890 \  
  --sources readSetId=1234567890 readSetId=1234567890 \  
  --role-arn arn:aws:iam::123456789012:role/omics-service-role-serviceRole-  
W801XMPL7QZ  
 \  
  --destination s3://omics-artifacts-01d6xmpl4e72dd32/read-set-export/
```

输出：

```
{
  "creationTime": "2022-12-06T22:37:18.612Z",
  "destination": "s3://omics-artifacts-01d6xmpl4e72dd32/read-set-export/",
  "id": "1234567890",
  "sequenceStoreId": "1234567890",
  "status": "SUBMITTED"
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartReadSetExportJob](#) 中的。

## start-read-set-import-job

以下代码示例演示了如何使用 start-read-set-import-job。

### AWS CLI

#### 导入读取集

以下 start-read-set-import-job 示例导入读取集。

```
aws omics start-read-set-import-job \
  --sequence-store-id 1234567890 \
  --role-arn arn:aws:iam::123456789012:role/omics-service-role-serviceRole-
W801XMPL7QZ \
  --sources file://readset-sources.json
```

readset-sources.json 是一个 JSON 文档，内容如下。

```
[
  {
    "sourceFiles":
    {
      "source1": "s3://omics-artifacts-01d6xmpl4e72dd32/
HG00100.chrom20.ILLUMINA.bwa.GBR.low_coverage.20101123.bam"
    },
    "sourceFileType": "BAM",
    "subjectId": "bam-subject",
    "sampleId": "bam-sample",
    "referenceArn": "arn:aws:omics:us-
west-2:123456789012:referenceStore/1234567890/reference/1234567890",
```

```
    "name": "HG00100"
  }
]
```

输出：

```
{
  "creationTime": "2022-11-23T01:36:38.158Z",
  "id": "1234567890",
  "roleArn": "arn:aws:iam::123456789012:role/omics-service-role-serviceRole-
W801XMPL7QZ",
  "sequenceStoreId": "1234567890",
  "status": "SUBMITTED"
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartReadSetImportJob](#) 中的。

## start-reference-import-job

以下代码示例演示了如何使用 start-reference-import-job。

### AWS CLI

#### 导入参考基因组

以下 start-reference-import-job 示例从 Amazon S3 导入参考基因组。

```
aws omics start-reference-import-job \
  --reference-store-id 1234567890 \
  --role-arn arn:aws:iam::123456789012:role/omics-service-role-serviceRole-
W801XMPL7QZ \
  --sources sourceFile=s3://omics-artifacts-01d6xmpl4e72dd32/
Homo_sapiens_assembly38.fasta,name=assembly-38
```

输出：

```
{
  "creationTime": "2022-11-22T22:25:41.124Z",
  "id": "1234567890",
  "referenceStoreId": "1234567890",
```



```
"roleArn": "arn:aws:iam::123456789012:role/omics-service-role-serviceRole-
W801XMPL7QZ",
  "status": "SUBMITTED"
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartReferenceImportJob](#)中的。

## start-run

以下代码示例演示了如何使用 start-run。

### AWS CLI

#### 运行工作流程

以下start-run示例运行一个带有 ID 的工作流程1234567。

```
aws omics start-run \
  --workflow-id 1234567 \
  --role-arn arn:aws:iam::123456789012:role/omics-service-role-serviceRole-
W801XMPL7QZ \
  --name 'cram-to-bam' \
  --output-uri s3://omics-artifacts-01d6xmpl4e72dd32/workflow-output/ \
  --run-group-id 1234567 \
  --priority 1 \
  --storage-capacity 10 \
  --log-level ALL \
  --parameters file://workflow-inputs.json
```

workflow-inputs.json 是一个 JSON 文档，内容如下。

```
{
  "sample_name": "NA12878",
  "input_cram": "s3://omics-artifacts-01d6xmpl4e72dd32/NA12878.cram",
  "ref_dict": "s3://omics-artifacts-01d6xmpl4e72dd32/
Homo_sapiens_assembly38.dict",
  "ref_fasta": "s3://omics-artifacts-01d6xmpl4e72dd32/
Homo_sapiens_assembly38.fasta",
  "ref_fasta_index": "omics-artifacts-01d6xmpl4e72dd32/
Homo_sapiens_assembly38.fasta.fai"
```

```
}
```

输出：

```
{
  "arn": "arn:aws:omics:us-west-2:123456789012:run/1234567",
  "id": "1234567",
  "status": "PENDING",
  "tags": {}
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

从 Amazon Omics 加载源文件

您还可以使用特定于服务的 URI 从 Amazon Omics 存储空间加载源文件。以下示例 workflow-inputs.json 文件使用 Amazon Omics URI 作为读取集和参考基因组源。

```
{
  "sample_name": "NA12878",
  "input_cram": "omics://123456789012.storage.us-west-2.amazonaws.com/1234567890/readSet/1234567890/source1",
  "ref_dict": "s3://omics-artifacts-01d6xmpl4e72dd32/Homo_sapiens_assembly38.dict",
  "ref_fasta": "omics://123456789012.storage.us-west-2.amazonaws.com/1234567890/reference/1234567890",
  "ref_fasta_index": "omics://123456789012.storage.us-west-2.amazonaws.com/1234567890/reference/1234567890/index"
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartRun](#) 中的。

## start-variant-import-job

以下代码示例演示了如何使用 start-variant-import-job。

### AWS CLI

导入变体文件

以下 `start-variant-import-job` 示例导入 VCF 格式的变体文件。

```
aws omics start-variant-import-job \  
  --destination-name my_var_store \  
  --no-run-left-normalization \  
  --role-arn arn:aws:iam::123456789012:role/omics-service-role-serviceRole-  
W801XMPL7QZ \  
  --items source=s3://omics-artifacts-01d6xmpl4e72dd32/  
Homo_sapiens_assembly38.known_indels.vcf.gz
```

输出：

```
{  
  "jobId": "edd7b8ce-xmpl-47e2-bc99-258cac95a508"  
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analytics](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartVariantImportJob](#) 中的。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

### AWS CLI

为资源添加标签

以下 `tag-resource` 示例向工作流程添加 `department` 标记，标识为 1234567。

```
aws omics tag-resource \  
  --resource-arn arn:aws:omics:us-west-2:123456789012:workflow/1234567 \  
  --tags department=analytics
```

有关更多信息，请参阅《[亚马逊 Omics 开发者指南](#)》中的 [在 Amazon Omics 中为资源添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 `untag-resource`。

## AWS CLI

### 从资源中移除标签

以下`untag-resource`示例将`department`标签从工作流程中移除。

```
aws omics untag-resource \  
  --resource-arn arn:aws:omics:us-west-2:123456789012:workflow/1234567 \  
  --tag-keys department
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-annotation-store

以下代码示例演示了如何使用 `update-annotation-store`。

### AWS CLI

#### 更新注释存储库

以下`update-annotation-store`示例更新了名为的注释存储库的描述`my_vcf_store`。

```
aws omics update-annotation-store \  
  --name my_vcf_store \  
  --description "VCF annotation store"
```

输出：

```
{  
  "creationTime": "2022-12-05T18:00:56.101860Z",  
  "description": "VCF annotation store",  
  "id": "bd6axmpl2444",  
  "name": "my_vcf_store",  
  "reference": {  
    "referenceArn": "arn:aws:omics:us-west-2:123456789012:referenceStore/1234567890/reference/1234567890"  
  },  
  "status": "ACTIVE",  
  "storeFormat": "VCF",  
  "updateTime": "2022-12-05T18:13:16.100051Z"
```

```
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analytics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAnnotationStore](#)中的。

## update-run-group

以下代码示例演示了如何使用 update-run-group。

### AWS CLI

#### 更新跑步组

以下update-run-group示例更新了带有 id 的跑步组的设置1234567。

```
aws omics update-run-group \  
  --id 1234567 \  
  --max-cpus 10
```

输出：

```
{  
  "arn": "arn:aws:omics:us-west-2:123456789012:runGroup/1234567",  
  "creationTime": "2022-12-01T00:58:42.915219Z",  
  "id": "1234567",  
  "maxCpus": 10,  
  "maxDuration": 600,  
  "name": "cram-convert",  
  "tags": {}  
}
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [工作流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRunGroup](#)中的。

## update-variant-store

以下代码示例演示了如何使用 update-variant-store。

### AWS CLI

#### 更新多属性商店

以下update-variant-store示例更新了名为的变体商店的描述my\_var\_store。

```
aws omics update-variant-store \  
  --name my_var_store \  
  --description "variant store"
```

输出：

```
{  
  "creationTime": "2022-11-23T22:09:07.534499Z",  
  "description": "variant store",  
  "id": "02dexmplcfdd",  
  "name": "my_var_store",  
  "reference": {  
    "referenceArn": "arn:aws:omics:us-  
west-2:123456789012:referenceStore/1234567890/reference/1234567890"  
  },  
  "status": "ACTIVE",  
  "updateTime": "2022-12-05T18:23:37.686402Z"  
}
```

有关更多信息，请参阅《Amazon [Omics 开发者指南](#)》中的 [Omics Analyt ics](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateVariantStore](#)中的。

## update-workflow

以下代码示例演示了如何使用 update-workflow。

AWS CLI

### 更新工作流程

以下update-workflow示例更新了带有 ID 的工作流程的描述1234567。

```
aws omics update-workflow \  
  --id 1234567 \  
  --description "copy workflow"
```

有关更多信息，请参阅《Amazon Omics 开发者指南》中的 Omics [存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateWorkflow](#)中的。

## upload-read-set-part

以下代码示例演示了如何使用 `upload-read-set-part`。

### AWS CLI

上传读取集分段。

以下 `upload-read-set-part` 示例上传读取集的指定部分。

```
aws omics upload-read-set-part \  
  --sequence-store-id 0123456789 \  
  --upload-id 1122334455 \  
  --part-source SOURCE1 \  
  --part-number 1 \  
  --payload /path/to/file/read_1_part_1.fastq.gz
```

输出：

```
{  
  "checksum": "984979b9928ae8d8622286c4a9cd8e99d964a22d59ed0f5722e1733eb280e635"  
}
```

有关更多信息，请参阅《AWS HealthOmics 用户指南》中的[直接上传到序列存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UploadReadSetPart](#) 中的。

## 使用的 IAM 示例 AWS CLI

以下代码示例向您展示了如何使用 `with IAM` 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### **add-client-id-to-open-id-connect-provider**

以下代码示例演示了如何使用 `add-client-id-to-open-id-connect-provider`。

#### AWS CLI

将客户端 ID ( 受众 ) 添加到 Open-ID Connect ( OIDC ) 提供者

以下 `add-client-id-to-open-id-connect-provider` 命令将客户端 ID `my-application-ID` 添加到名为 `server.example.com` 的 OIDC 提供者。

```
aws iam add-client-id-to-open-id-connect-provider \  
  --client-id my-application-ID \  
  --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/  
server.example.com
```

此命令不生成任何输出。

要创建 OIDC 提供者，请使用 `create-open-id-connect-provider` 命令。

有关更多信息，请参阅《AWS IAM 用户指南》中的 [创建 OpenID Connect \( OIDC \) 身份提供者](#)。

• 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AddClientIdToOpenIdConnectProvider](#) 中的。

### **add-role-to-instance-profile**

以下代码示例演示了如何使用 `add-role-to-instance-profile`。

#### AWS CLI

向实例配置文件添加角色

以下 `add-role-to-instance-profile` 命令可将名为 `S3Access` 的角色添加到名为 `Webserver` 的实例配置文件。

```
aws iam add-role-to-instance-profile \  
  --role-name S3Access \  
  --instance-profile-name Webserver
```

此命令不生成任何输出。



要创建实例配置文件，请使用 `create-instance-profile` 命令。

有关更多信息，请参阅《IAM 用户指南》中的 [使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限](#) AWS。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AddRoleToInstanceProfile](#) 中的。

## add-user-to-group

以下代码示例演示了如何使用 `add-user-to-group`。

### AWS CLI

要将用户添加到 IAM 组

以下 `add-user-to-group` 命令可将名为 Bob 的 IAM 用户添加到名为 Admins 的 IAM 组。

```
aws iam add-user-to-group \  
  --user-name Bob \  
  --group-name Admins
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的 [在 IAM 用户组中添加和删除用户](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AddUserToGroup](#) 中的。

## attach-group-policy

以下代码示例演示了如何使用 `attach-group-policy`。

### AWS CLI

将托管策略附加到 IAM 组

以下 `attach-group-policy` 命令将名为的 AWS 托管策略附加 `ReadOnlyAccess` 到名为的 IAM 组 `Finance`。

```
aws iam attach-group-policy \  
  --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \  
  --group-name Finance
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[托管策略和内联策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachGroupPolicy](#)中的。

## attach-role-policy

以下代码示例演示了如何使用 attach-role-policy。

### AWS CLI

要将托管策略附加到 IAM 角色

以下attach-role-policy命令将名为的 AWS 托管策略附加ReadOnlyAccess到名为的 IAM 角色ReadOnlyRole。

```
aws iam attach-role-policy \  
  --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \  
  --role-name ReadOnlyRole
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[托管策略和内联策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachRolePolicy](#)中的。

## attach-user-policy

以下代码示例演示了如何使用 attach-user-policy。

### AWS CLI

要将托管策略附加到 IAM 用户

以下attach-user-policy命令将名为的 AWS 托管策略附加AdministratorAccess到名为的 IAM 用户Alice。

```
aws iam attach-user-policy \  
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \  
  --user-name Alice
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[托管策略和内联策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachUserPolicy](#)中的。

## change-password

以下代码示例演示了如何使用 change-password。

### AWS CLI

#### 更改 IAM 用户的密码

要更改 IAM 用户的密码，建议使用 `--cli-input-json` 参数传递包含新旧密码的 JSON 文件。通过此方法，您可以使用含有非字母数字字符的强密码。将密码作为命令行参数传递时，可能难以使用含有非字母数字字符的密码。要使用 `--cli-input-json` 参数，请先使用带 `--generate-cli-skeleton` 参数的 `change-password` 命令，如以下示例所示。

```
aws iam change-password \  
  --generate-cli-skeleton > change-password.json
```

前述命令创建一个名为 `change-password.json` 的 JSON 文件，您可以用该文件来填写旧密码和新密码。例如，该文件可能如下所示。

```
{  
  "OldPassword": "3s0K_;xh4~8XXI",  
  "NewPassword": "]35d/{pB9Fo9wJ"  
}
```

接下来，要更改密码，请再次使用 `change-password` 命令，这次是传递 `--cli-input-json` 参数以指定您的 JSON 文件。以下 `change-password` 命令将 `--cli-input-json` 参数与名为 `change-password.json` 的 JSON 文件一起使用。

```
aws iam change-password \  
  --cli-input-json file://change-password.json
```

此命令不生成任何输出。

此命令只能由 IAM 用户调用。如果使用 AWS 账户（根）凭据调用此命令，则该命令将返回 `InvalidUserType` 错误。

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 用户如何更改自己的密码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ChangePassword](#)中的。

## create-access-key

以下代码示例演示了如何使用 create-access-key。

### AWS CLI

要为 IAM 用户创建访问密钥

以下 create-access-key 命令将创建名为 Bob IAM 用户的访问密钥 ( 访问密钥 ID 和秘密访问密钥 )。

```
aws iam create-access-key \  
  --user-name Bob
```

输出：

```
{  
  "AccessKey": {  
    "UserName": "Bob",  
    "Status": "Active",  
    "CreateDate": "2015-03-09T18:39:23.411Z",  
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxrFiCYzEXAMPLEKEY",  
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"  
  }  
}
```

将秘密访问密钥存储在安全位置。如果访问密钥丢失，将无法恢复，必须创建一个新的访问密钥。

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM 用户的访问密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateAccessKey](#) 中的。

## create-account-alias

以下代码示例演示了如何使用 create-account-alias。

### AWS CLI

要创建账户别名

以下 create-account-alias 命令 examplecorp 为您的 AWS 账户创建别名。

```
aws iam create-account-alias \  
  --account-alias examplecorp
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IAM 用户指南中的[您的 AWS 账户 ID 及其别名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateAccountAlias](#)中的。

## create-group

以下代码示例演示了如何使用 create-group。

### AWS CLI

创建 IAM 组

以下 create-group 命令将创建名为 Admins 的 IAM 组。

```
aws iam create-group \  
  --group-name Admins
```

输出：

```
{  
  "Group": {  
    "Path": "/",  
    "CreateDate": "2015-03-09T20:30:24.940Z",  
    "GroupId": "AIDGPM59R04H3FEXAMPLE",  
    "Arn": "arn:aws:iam::123456789012:group/Admins",  
    "GroupName": "Admins"  
  }  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 IAM 用户组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateGroup](#)中的。

## create-instance-profile

以下代码示例演示了如何使用 create-instance-profile。

## AWS CLI

### 要创建实例配置文件

以下 `create-instance-profile` 命令将创建名为 `Webserver` 的实例配置文件。

```
aws iam create-instance-profile \  
  --instance-profile-name Webserver
```

输出：

```
{  
  "InstanceProfile": {  
    "InstanceId": "AIPAJMBC7DLSPEXAMPLE",  
    "Roles": [],  
    "CreateDate": "2015-03-09T20:33:19.626Z",  
    "InstanceProfileName": "Webserver",  
    "Path": "/",  
    "Arn": "arn:aws:iam::123456789012:instance-profile/Webserver"  
  }  
}
```

要将角色添加到实例配置文件，请使用 `add-role-to-instance-profile` 命令。

有关更多信息，请参阅《AWS IAM 用户指南》中的[使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateInstanceProfile](#) 中的。

## create-login-profile

以下代码示例演示了如何使用 `create-login-profile`。

### AWS CLI

#### 为 IAM 用户创建密码

要为 IAM 用户创建密码，建议使用 `--cli-input-json` 参数传递包含密码的 JSON 文件。通过此方法，您可以创建含有非字母数字字符的强密码。将密码作为命令行参数传递时，可能难以创建含有非字母数字字符的密码。

要使用 `--cli-input-json` 参数，请先使用带 `--generate-cli-skeleton` 参数的 `create-login-profile` 命令，如以下示例所示。

```
aws iam create-login-profile \  
  --generate-cli-skeleton > create-login-profile.json
```

前面的命令创建了一个名为 create-login-profile.json 的 JSON 文件，您可以使用该文件为后续 create-login-profile 命令填写信息。例如：

```
{  
  "UserName": "Bob",  
  "Password": "&1-3a6u:RA0djs",  
  "PasswordResetRequired": true  
}
```

接下来，要为 IAM 用户创建密码，请再次使用 create-login-profile 命令，这次是传递 --cli-input-json 参数以指定您的 JSON 文件。以下 create-login-profile 命令将 --cli-input-json 参数与名为 create-login-profile.json 的 JSON 文件一起使用。

```
aws iam create-login-profile \  
  --cli-input-json file://create-login-profile.json
```

输出：

```
{  
  "LoginProfile": {  
    "UserName": "Bob",  
    "CreateDate": "2015-03-10T20:55:40.274Z",  
    "PasswordResetRequired": true  
  }  
}
```

如果新密码违反了账户密码策略，则该命令将返回 PasswordPolicyViolation 错误。

要更改已有密码用户的密码，请使用 update-login-profile。要为账户设置密码策略，请使用 update-account-password-policy 命令。

如果账户密码策略允许，则 IAM 用户可以使用 change-password 命令更改自己的密码。

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM 用户的密码](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateLoginProfile](#) 中的。

## create-open-id-connect-provider

以下代码示例演示了如何使用 `create-open-id-connect-provider`。

### AWS CLI

#### 创建 OpenID Connect ( OIDC ) 提供者

要创建 OpenID Connect ( OIDC ) 提供者，建议使用 `--cli-input-json` 参数来传递包含所需参数的 JSON 文件。创建 OIDC 提供者时，必须传递提供者的 URL，而且 URL 必须以 `https://` 开头。将 URL 作为命令行参数传递可能很困难，因为冒号 ( `:` ) 和正斜杠 ( `/` ) 字符在某些命令行环境中具有特殊含义。使用 `--cli-input-json` 参数可以绕过这一限制。

要使用 `--cli-input-json` 参数，请先使用带 `--generate-cli-skeleton` 参数的 `create-open-id-connect-provider` 命令，如以下示例所示。

```
aws iam create-open-id-connect-provider \  
  --generate-cli-skeleton > create-open-id-connect-provider.json
```

前面的命令创建了一个名为 `create-open-id-connect-provider.json` 的 JSON 文件，你可以用它来填写后续命令的信息。`create-open-id-connect-provider` 例如：

```
{  
  "Url": "https://server.example.com",  
  "ClientIDList": [  
    "example-application-ID"  
  ],  
  "ThumbprintList": [  
    "c3768084dfb3d2b68b7897bf5f565da8eEXAMPLE"  
  ]  
}
```

接下来，要创建 OpenID Connect ( OIDC ) 提供者，请再次使用 `create-open-id-connect-provider` 命令，这次是传递 `--cli-input-json` 参数以指定您的 JSON 文件。以下 `create-open-id-connect-provider` 命令将 `--cli-input-json` 参数与名为 `create-open-id-connect-provider.json` 的 JSON 文件一起使用。

```
aws iam create-open-id-connect-provider \  
  --cli-input-json file://create-open-id-connect-provider.json
```

输出：



```
{
  "OpenIDConnectProviderArn": "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
}
```

有关 OIDC 提供者的更多信息，请参阅《AWS IAM 用户指南》中的[创建 OpenID Connect \( OIDC \) 身份提供者](#)。

有关获取 OIDC 提供者指纹的更多信息，请参阅《AWS IAM 用户指南》中的[获取 OpenID Connect 身份提供者的指纹](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateOpenIdConnectProvider](#)中的。

## create-policy-version

以下代码示例演示了如何使用 create-policy-version。

### AWS CLI

要创建新版本的托管策略

此示例创建了一个新的 v2 版本的 IAM policy，其 ARN 为 arn:aws:iam::123456789012:policy/MyPolicy，并设为默认版本。

```
aws iam create-policy-version \
  --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
  --policy-document file://NewPolicyVersion.json \
  --set-as-default
```

输出：

```
{
  "PolicyVersion": {
    "CreateDate": "2015-06-16T18:56:03.721Z",
    "VersionId": "v2",
    "IsDefaultVersion": true
  }
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[IAM policy 版本控制](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreatePolicyVersion](#)中的。

## create-policy

以下代码示例演示了如何使用 create-policy。

### AWS CLI

#### 示例 1：创建客户管理型策略

以下命令将创建名为 my-policy 的客户管理型策略。

```
aws iam create-policy \  
  --policy-name my-policy \  
  --policy-document file://policy
```

policy 文件是当前文件夹中的一个 JSON 文档，它授予对名为 my-bucket 的 Amazon S3 存储桶中 shared 文件夹的只读权限。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:Get*",  
        "s3:List*"  
      ],  
      "Resource": [  
        "arn:aws:s3:::my-bucket/shared/*"  
      ]  
    }  
  ]  
}
```

输出：

```
{  
  "Policy": {  
    "PolicyName": "my-policy",  
    "CreateDate": "2015-06-01T19:31:18.620Z",  
    "AttachmentCount": 0,  
    "IsAttachable": true,  
    "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",  
    "DefaultVersionId": "v1",
```

```
    "Path": "/",
    "Arn": "arn:aws:iam::0123456789012:policy/my-policy",
    "UpdateDate": "2015-06-01T19:31:18.620Z"
  }
}
```

有关使用文件作为字符串参数输入的更多信息，请参阅 [AWS CLI 用户指南中的为 CL AWS I 指定参数值](#)。

## 示例 2：创建带有描述的客户管理型策略

以下命令将创建名为 my-policy 且带有不可变描述的客户管理型策略：

```
aws iam create-policy \
  --policy-name my-policy \
  --policy-document file://policy.json \
  --description "This policy grants access to all Put, Get, and List actions for
my-bucket"
```

policy.json 文件是当前文件夹中的一个 JSON 文档，它授予对名为 my-bucket 的 Amazon S3 存储桶的所有 Put、List 和 Get 操作的访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket*",
        "s3:PutBucket*",
        "s3:GetBucket*"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket"
      ]
    }
  ]
}
```

输出：

```
{
```

```
"Policy": {
  "PolicyName": "my-policy",
  "PolicyId": "ANPAWGSUGIDPEXAMPLE",
  "Arn": "arn:aws:iam::123456789012:policy/my-policy",
  "Path": "/",
  "DefaultVersionId": "v1",
  "AttachmentCount": 0,
  "PermissionsBoundaryUsageCount": 0,
  "IsAttachable": true,
  "CreateDate": "2023-05-24T22:38:47+00:00",
  "UpdateDate": "2023-05-24T22:38:47+00:00"
}
```

有关基于身份的策略的更多信息，请参阅《AWS IAM 用户指南》中的[基于身份的策略和基于资源的策略](#)。

### 示例 3：创建带有标签的客户管理型策略

以下命令将创建名为 my-policy 且带有标签的客户管理型策略。此示例使用带有以下 JSON 格式标签的 --tags 参数标志：'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'。或者，--tags 标志可与简写格式的标签一起使用：'Key=Department,Value=Accounting Key=Location,Value=Seattle'。

```
aws iam create-policy \
  --policy-name my-policy \
  --policy-document file://policy.json \
  --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
"Value": "Seattle"}'
```

policy.json 文件是当前文件夹中的一个 JSON 文档，它授予对名为 my-bucket 的 Amazon S3 存储桶的所有 Put、List 和 Get 操作的访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket*",
        "s3:PutBucket*",

```

```
        "s3:GetBucket*"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket"
      ]
    }
  ]
}
```

输出：

```
{
  "Policy": {
    "PolicyName": "my-policy",
    "PolicyId": "ANPAWGSUGIDPEXAMPLE",
    "Arn": "arn:aws:iam::12345678012:policy/my-policy",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2023-05-24T23:16:39+00:00",
    "UpdateDate": "2023-05-24T23:16:39+00:00",
    "Tags": [
      {
        "Key": "Department",
        "Value": "Accounting"
      },
      {
        "Key": "Location",
        "Value": "Seattle"
      }
    ]
  }
}
```

有关标记策略的更多信息，请参阅《AWS IAM 用户指南》中的[标记客户管理型策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePolicy](#)中的。

## create-role

以下代码示例演示了如何使用 create-role。

## AWS CLI

### 示例 1：创建 IAM 角色

以下 `create-role` 命令将创建一个名为 `Test-Role` 的角色并对其附加信任策略。

```
aws iam create-role \  
  --role-name Test-Role \  
  --assume-role-policy-document file://Test-Role-Trust-Policy.json
```

输出：

```
{  
  "Role": {  
    "AssumeRolePolicyDocument": "<URL-encoded-JSON>",  
    "RoleId": "AKIAIOSFODNN7EXAMPLE",  
    "CreateDate": "2013-06-07T20:43:32.821Z",  
    "RoleName": "Test-Role",  
    "Path": "/",  
    "Arn": "arn:aws:iam::123456789012:role/Test-Role"  
  }  
}
```

信任策略在 `Test-Role-Trust-Policy.json` 文件中定义为 JSON 文档。（文件名和扩展名没有意义。）信任策略必须指定主体。

要将权限策略附加到角色，请使用 `put-role-policy` 命令。

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 IAM 角色](#)。

### 示例 2：创建具有指定最长会话持续时间的 IAM 角色

以下 `create-role` 命令将创建一个名为 `Test-Role` 的角色，并将最长会话持续时间设置为 7200 秒（2 小时）。

```
aws iam create-role \  
  --role-name Test-Role \  
  --assume-role-policy-document file://Test-Role-Trust-Policy.json \  
  --max-session-duration 7200
```

输出：

```
{
  "Role": {
    "Path": "/",
    "RoleName": "Test-Role",
    "RoleId": "AKIAIOSFODNN7EXAMPLE",
    "Arn": "arn:aws:iam::12345678012:role/Test-Role",
    "CreateDate": "2023-05-24T23:50:25+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "Statement1",
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::12345678012:root"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
}
```

有关更多信息，请参阅 [AWS IAM 用户指南中的修改角色最长会话持续时间 \(AWS API\)](#)。

### 示例 3：创建带有标签的 IAM 角色

以下命令将创建带有标签的 IAM 角色 Test-Role。此示例使用带有以下 JSON 格式标签的 `--tags` 参数标志：'`{"Key": "Department", "Value": "Accounting"}`' '`{"Key": "Location", "Value": "Seattle"}`'。或者，`--tags` 标志可与简写格式的标签一起使用：`'Key=Department,Value=Accounting Key=Location,Value=Seattle'`。

```
aws iam create-role \
  --role-name Test-Role \
  --assume-role-policy-document file://Test-Role-Trust-Policy.json \
  --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
  "Value": "Seattle"}'
```

输出：

```
{
```

```
"Role": {
  "Path": "/",
  "RoleName": "Test-Role",
  "RoleId": "AKIAIOSFODNN7EXAMPLE",
  "Arn": "arn:aws:iam::123456789012:role/Test-Role",
  "CreateDate": "2023-05-25T23:29:41+00:00",
  "AssumeRolePolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "Statement1",
        "Effect": "Allow",
        "Principal": {
          "AWS": "arn:aws:iam::123456789012:root"
        },
        "Action": "sts:AssumeRole"
      }
    ]
  },
  "Tags": [
    {
      "Key": "Department",
      "Value": "Accounting"
    },
    {
      "Key": "Location",
      "Value": "Seattle"
    }
  ]
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 角色](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRole](#)中的。

## create-saml-provider

以下代码示例演示了如何使用 create-saml-provider。

### AWS CLI

要创建 SAML 提供者



此示例在 IAM 中创建了一个名为 MySAMLProvider 的新 SAML 提供者。SAMLMetaData.xml 文件中的 SAML 元数据文档对其进行了描述。

```
aws iam create-saml-provider \  
  --saml-metadata-document file://SAMLMetaData.xml \  
  --name MySAMLProvider
```

输出：

```
{  
  "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/MySAMLProvider"  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 IAM SAML 身份提供者](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的[CreateSAMLProvider](#)。

## create-service-linked-role

以下代码示例演示了如何使用 create-service-linked-role。

### AWS CLI

要创建服务相关角色

以下create-service-linked-role示例为指定 AWS 服务创建服务相关角色并附加指定的描述。

```
aws iam create-service-linked-role \  
  --aws-service-name lex.amazonaws.com \  
  --description "My service-linked role to support Lex"
```

输出：

```
{  
  "Role": {  
    "Path": "/aws-service-role/lex.amazonaws.com/",  
    "RoleName": "AWSServiceRoleForLexBots",  
    "RoleId": "ARO1234567890EXAMPLE",  
    "Arn": "arn:aws:iam::1234567890:role/aws-service-role/lex.amazonaws.com/  
AWSServiceRoleForLexBots",
```

```
"CreateDate": "2019-04-17T20:34:14+00:00",
"AssumeRolePolicyDocument": {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "sts:AssumeRole"
      ],
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "lex.amazonaws.com"
        ]
      }
    }
  ]
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[使用服务相关角色](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateServiceLinkedRole](#)中的。

## create-service-specific-credential

以下代码示例演示了如何使用 create-service-specific-credential。

### AWS CLI

为用户创建一组特定于服务的凭证

以下create-service-specific-credential示例创建的用户名和密码只能用于访问已配置的服务。

```
aws iam create-service-specific-credential \
  --user-name sofia \
  --service-name codecommit.amazonaws.com
```

输出：

```
{
```

```
"ServiceSpecificCredential": {
  "CreateDate": "2019-04-18T20:45:36+00:00",
  "ServiceName": "codecommit.amazonaws.com",
  "ServiceUserName": "sofia-at-123456789012",
  "ServicePassword": "k1zPZM6uVxMQ3oxqgoY1NuJPyRTZ1vREs76zTQE3eJk=",
  "ServiceSpecificCredentialId": "ACCAEXAMPLE123EXAMPLE",
  "UserName": "sofia",
  "Status": "Active"
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》CodeCommit中的[创建 HTTPS 连接的 Git 凭证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateServiceSpecificCredential](#)中的。

## create-user

以下代码示例演示了如何使用 create-user。

### AWS CLI

#### 示例 1：创建 IAM 用户

以下 create-user 命令可在当前账户中创建一个名为 Bob 的 IAM 用户。

```
aws iam create-user \
  --user-name Bob
```

输出：

```
{
  "User": {
    "UserName": "Bob",
    "Path": "/",
    "CreateDate": "2023-06-08T03:20:41.270Z",
    "UserId": "AIDAIOSFODNN7EXAMPLE",
    "Arn": "arn:aws:iam::123456789012:user/Bob"
  }
}
```

有关更多信息，请参阅 [IAM 用户指南中的在您的 AWS 账户中创建AWS IAM 用户](#)。

## 示例 2：在指定路径创建 IAM 用户

以下 `create-user` 命令可在指定路径创建一个名为 Bob 的 IAM 用户。

```
aws iam create-user \  
  --user-name Bob \  
  --path /division_abc/subdivision_xyz/
```

输出：

```
{  
  "User": {  
    "Path": "/division_abc/subdivision_xyz/",  
    "UserName": "Bob",  
    "UserId": "AIDAIOSFODNN7EXAMPLE",  
    "Arn": "arn:aws:iam::12345678012:user/division_abc/subdivision_xyz/Bob",  
    "CreateDate": "2023-05-24T18:20:17+00:00"  
  }  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 标识符](#)。

## 示例 3：创建带有标签的 IAM 用户

以下 `create-user` 命令将创建一个名为 Bob 且带有标签的 IAM 用户。此示例使用带有以下 JSON 格式标签的 `--tags` 参数标志：`'{"Key": "Department", "Value": "Accounting"}'` `'{"Key": "Location", "Value": "Seattle"}'`。或者，`--tags` 标志可与简写格式的标签一起使用：`'Key=Department,Value=Accounting Key=Location,Value=Seattle'`。

```
aws iam create-user \  
  --user-name Bob \  
  --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",  
  "Value": "Seattle"}'
```

输出：

```
{  
  "User": {  
    "Path": "/",  
    "UserName": "Bob",
```

```
"UserId": "AIDAIOSFODNN7EXAMPLE",
"Arn": "arn:aws:iam::12345678012:user/Bob",
"CreateDate": "2023-05-25T17:14:21+00:00",
"Tags": [
  {
    "Key": "Department",
    "Value": "Accounting"
  },
  {
    "Key": "Location",
    "Value": "Seattle"
  }
]
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 用户](#)。

### 示例 3：创建具有设定权限边界的 IAM 用户

以下 `create-user` 命令创建一个名为的 IAM 用户，Bob 其权限范围为 AmazonS3 FullAccess。

```
aws iam create-user \
  --user-name Bob \
  --permissions-boundary arn:aws:iam::aws:policy/AmazonS3FullAccess
```

输出：

```
{
  "User": {
    "Path": "/",
    "UserName": "Bob",
    "UserId": "AIDAIOSFODNN7EXAMPLE",
    "Arn": "arn:aws:iam::12345678012:user/Bob",
    "CreateDate": "2023-05-24T17:50:53+00:00",
    "PermissionsBoundary": {
      "PermissionsBoundaryType": "Policy",
      "PermissionsBoundaryArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
    }
  }
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 实体的权限边界](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUser](#)中的。

## create-virtual-mfa-device

以下代码示例演示了如何使用 `create-virtual-mfa-device`。

### AWS CLI

#### 创建虚拟 MFA 设备

此示例创建名为 `BobsMFADevice` 的新虚拟 MFA 设备。它会创建一个包含引导信息的、名为 `QRCode.png` 的文件，并将其放在 `C:/` 目录中。本示例中使用的引导方法为 `QRCodePNG`。

```
aws iam create-virtual-mfa-device \  
  --virtual-mfa-device-name BobsMFADevice \  
  --outfile C:/QRCode.png \  
  --bootstrap-method QRCodePNG
```

输出：

```
{  
  "VirtualMFADevice": {  
    "SerialNumber": "arn:aws:iam::210987654321:mfa/BobsMFADevice"  
  }  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[在 AWS 中使用多重身份验证 \(MFA\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVirtualMfaDevice](#)中的。

## deactivate-mfa-device

以下代码示例演示了如何使用 `deactivate-mfa-device`。

### AWS CLI

#### 停用 MFA 设备

此命令使用与用户 `Bob` 关联的 ARN `arn:aws:iam::210987654321:mfa/BobsMFADevice` 停用虚拟 MFA 设备。

```
aws iam deactivate-mfa-device \  
  --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice
```

```
--user-name Bob \  
--serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[在 AWS 中使用多重身份验证 \(MFA\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeactivateMfaDevice](#) 中的。

## decode-authorization-message

以下代码示例演示了如何使用 decode-authorization-message。

### AWS CLI

#### 解码授权失败消息

以下 decode-authorization-message 示例解码 EC2 控制台在没有所需权限的情况下尝试启动实例时返回的消息。

```
aws sts decode-authorization-message \  
  --encoded-message lxzA8VEjEvu-s0TTt3PgYCXik9Yak0qsrFJGRZR98xNcyWAxwRq14xIvd-  
npzbgTevuufCTbjeBAaDARg9cbTK1rJbg3awM33o-Vy3ebPErE2-  
mWR9hVYdvX-0zKgV0WF9pWjZaJSMqxB-aLXo-I_8TTvBq88x8IFPbMArNdpu0IjxDjzf22PF3S0E3XvIQ-  
_PE00aUqHCCcsSrFtvxm6yQD1nbm6VTIVrfa0Bzy8lsoMo7SjIaJ2r5vph6SY5vCCwg6o2JKe3hIHTa8zRrDbZSFMkcX  
Xx9AYAAIr6bhcis7C__bZh4d1AAWooHFGKgoJcWGwgdzgbu9hWyVvKTpeot5hsb8qANYjJRCPXTKpi6PZfdijIkwb6g
```

输出格式为 JSON 文本的单行 JSON 文本字符串，您可以使用任何 JSON 文本处理器对其进行解析。

```
{  
  "DecodedMessage": "{\"allowed\":false,\"explicitDeny\":false,\"matchedStatements\":"  
  \":{\\"items\":[],\\"failures\":{\\"items\":[],\\"context\":{\\"principal\":"  
  \":{\\"id\":"AIDAV3ZUEFP6J7GY706L0\\",\\"name\":"chain-user\\",\\"arn\":"  
  \":{\\"arn:aws:iam::403299380220:user/chain-user\\"},\\"action\":"ec2:RunInstances\\",  
  \":{\\"resource\":"arn:aws:ec2:us-east-2:403299380220:instance/*\\"},\\"conditions\":"  
  \":{\\"items\":[{\\"key\":"ec2:InstanceMarketType\\",\\"values\":{\\"items\":[{\\"value\":"  
  \":{\\"on-demand\\"}]}]}],{\\"key\":"aws:Resource\\",\\"values\":{\\"items\":[{\\"value\":"  
  \":{\\"instance/*\\"}]}]}],{\\"key\":"aws:Account\\",\\"values\":{\\"items\":[{\\"value\":"  
  \":{\\"403299380220\\"}]}]}],{\\"key\":"ec2:AvailabilityZone\\",\\"values\":{\\"items\":"  
  \":{\\"value\":"us-east-2b\\"}]}]}],{\\"key\":"ec2:ebsoptimized\\",\\"values\":{\\"items\":"  
  \":{\\"value\":"false\\"}]}]}],{\\"key\":"ec2:IsLaunchTemplateResource\\",\\"values
```

```

\":"items\":[{"value\":"false\"}]},{"key\":"ec2:InstanceType\","\values
\":"items\":[{"value\":"t2.micro\"}]},{"key\":"ec2:RootDeviceType\","
\values\":"items\":[{"value\":"ebs\"}]},{"key\":"aws:Region\","\values
\":"items\":[{"value\":"us-east-2\"}]},{"key\":"aws:Service\","\values
\":"items\":[{"value\":"ec2\"}]},{"key\":"ec2:InstanceID\","\values\":"
items\":[{"value\":"*\"}]},{"key\":"aws:Type\","\values\":"items\":"
[{"value\":"instance\"}]},{"key\":"ec2:Tenancy\","\values\":"items\":"
[{"value\":"default\"}]},{"key\":"ec2:Region\","\values\":"items\":"[{"value
\":"us-east-2\"}]},{"key\":"aws:ARN\","\values\":"items\":"[{"value\":"
arn:aws:ec2:us-east-2:403299380220:instance/*\"}]}]}"}"
}

```

有关更多信息，请参阅在 [EC2 实例启动期间收到“UnauthorizedOperation”错误后如何解码授权失败消息](#)？在 re AWS : post 中。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DecodeAuthorizationMessage](#)中的。

## delete-access-key

以下代码示例演示了如何使用 delete-access-key。

### AWS CLI

要删除 IAM 用户的访问密钥

以下 delete-access-key 命令将删除名为 Bob IAM 用户的指定访问密钥（访问密钥 ID 和秘密访问密钥）。

```

aws iam delete-access-key \
  --access-key-id AKIDPMS9R04H3FEXAMPLE \
  --user-name Bob

```

此命令不生成任何输出。

要列出为 IAM 用户定义的访问密钥，请使用 list-access-keys 命令。

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM 用户的访问密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAccessKey](#)中的。

## delete-account-alias

以下代码示例演示了如何使用 delete-account-alias。



## AWS CLI

### 要删除账户别名

以下 `delete-account-alias` 命令将删除当前账户的别名 `mycompany`。

```
aws iam delete-account-alias \  
  --account-alias mycompany
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IAM 用户指南中的[您的 AWS 账户 ID 及其别名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteAccountAlias](#)中的。

## `delete-account-password-policy`

以下代码示例演示了如何使用 `delete-account-password-policy`。

### AWS CLI

#### 删除当前账户密码策略

以下 `delete-account-password-policy` 命令将删除当前账户的密码策略。

```
aws iam delete-account-password-policy
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[为 IAM 用户设置账户密码策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteAccountPasswordPolicy](#)中的。

## `delete-group-policy`

以下代码示例演示了如何使用 `delete-group-policy`。

### AWS CLI

#### 要从 IAM 组中删除策略

以下 `delete-group-policy` 命令可将名为 `ExamplePolicy` 的策略从名为 `Admins` 的组中删除。

```
aws iam delete-group-policy \  
  --group-name Admins \  
  --policy-name ExamplePolicy
```

此命令不生成任何输出。

要查看附加到组的策略，请使用 `list-group-policies` 命令。

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM policy](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteGroupPolicy](#)中的。

## delete-group

以下代码示例演示了如何使用 `delete-group`。

### AWS CLI

要删除 IAM 组

以下 `delete-group` 命令将删除名为 `MyTestGroup` 的 IAM 组。

```
aws iam delete-group \  
  --group-name MyTestGroup
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[删除 IAM 用户组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteGroup](#)中的。

## delete-instance-profile

以下代码示例演示了如何使用 `delete-instance-profile`。

### AWS CLI

要删除实例配置文件

以下 `delete-instance-profile` 命令将删除名为 `ExampleInstanceProfile` 的实例配置文件。

```
aws iam delete-instance-profile \  
  --instance-profile-name ExampleInstanceProfile
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[使用实例配置文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInstanceProfile](#)中的。

## delete-login-profile

以下代码示例演示了如何使用 delete-login-profile。

### AWS CLI

删除 IAM 用户的密码

以下 delete-login-profile 命令删除名为 Bob 的 IAM 用户的密码。

```
aws iam delete-login-profile \  
  --user-name Bob
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM 用户的密码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLoginProfile](#)中的。

## delete-open-id-connect-provider

以下代码示例演示了如何使用 delete-open-id-connect-provider。

### AWS CLI

删除 IAM OpenID Connect 身份提供者

此示例删除了连接到提供者 example.oidcprovider.com 的 IAM OIDC 提供者。

```
aws iam delete-open-id-connect-provider \  
  --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/  
example.oidcprovider.com
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的 [创建 OpenID Connect \( OIDC \) 身份提供者](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteOpenIdConnectProvider](#)中的。

## delete-policy-version

以下代码示例演示了如何使用 delete-policy-version。

### AWS CLI

删除托管策略的某个版本

此示例从 ARN 为 `arn:aws:iam::123456789012:policy/MySamplePolicy` 的策略中删除标识为 `v2` 的版本。

```
aws iam delete-policy-version \  
  --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \  
  --version-id v2
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePolicyVersion](#)中的。

## delete-policy

以下代码示例演示了如何使用 delete-policy。

### AWS CLI

要删除 IAM policy

此示例删除了 ARN 为 `arn:aws:iam::123456789012:policy/MySamplePolicy` 的策略。

```
aws iam delete-policy \  
  --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePolicy](#)中的。

## delete-role-permissions-boundary

以下代码示例演示了如何使用 delete-role-permissions-boundary。

### AWS CLI

删除 IAM 角色的权限边界

以下 delete-role-permissions-boundary 示例删除指定 IAM 角色的权限边界。要对角色应用权限边界，请使用 put-role-permissions-boundary 命令。

```
aws iam delete-role-permissions-boundary \  
  --role-name lambda-application-role
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRolePermissionsBoundary](#)中的。

## delete-role-policy

以下代码示例演示了如何使用 delete-role-policy。

### AWS CLI

要从 IAM 角色中删除策略

以下 delete-role-policy 命令可将名为 ExamplePolicy 的策略从名为 Test-Role 的角色中删除。

```
aws iam delete-role-policy \  
  --role-name Test-Role \  
  --policy-name ExamplePolicy
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的 [修改角色](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRolePolicy](#)中的。

## delete-role

以下代码示例演示了如何使用 delete-role。

### AWS CLI

要删除 IAM 角色

以下 delete-role 命令将删除名为 Test-Role 的角色。

```
aws iam delete-role \  
  --role-name Test-Role
```

此命令不生成任何输出。

在删除角色之前，必须从所有实例配置文件中删除该角色 ( remove-role-from-instance-profile )，分离所有托管策略 ( detach-role-policy )，删除附加到该角色的所有内联策略 ( delete-role-policy )。

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 IAM 角色](#)和[使用实例配置文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRole](#)中的。

## delete-saml-provider

以下代码示例演示了如何使用 delete-saml-provider。

### AWS CLI

要删除 SAML 提供者

此示例删除了 ARN 为 arn:aws:iam::123456789012:saml-provider/SAMLADFSPROVIDER 的 IAM SAML 2.0 提供者。

```
aws iam delete-saml-provider \  
  --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFSPROVIDER
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 IAM SAML 身份提供者](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteSAMLProvider](#)。

## delete-server-certificate

以下代码示例演示了如何使用 delete-server-certificate。

### AWS CLI

从您的 AWS 账户中删除服务器证书

以下delete-server-certificate命令从您的 AWS 账户中删除指定的服务器证书。

```
aws iam delete-server-certificate \  
  --server-certificate-name myUpdatedServerCertificate
```

此命令不生成任何输出。

要列出您 AWS 账户中可用的服务器证书，请使用list-server-certificates命令。

有关更多信息，请参阅《AWS IAM 用户指南》中的[在 IAM 中管理服务器证书](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteServerCertificate](#)中的。

## delete-service-linked-role

以下代码示例演示了如何使用 delete-service-linked-role。

### AWS CLI

要删除服务相关角色

以下 delete-service-linked-role 示例将删除您不再需要的指定服务相关角色。删除操作异步进行。您也可以使用 get-service-linked-role-deletion-status 命令查看删除状态并确认何时删除。

```
aws iam delete-service-linked-role \  
  --role-name AWSServiceRoleForLexBots
```

输出：

```
{
  "DeletionTaskId": "task/aws-service-role/lex.amazonaws.com/
  AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE"
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[使用服务相关角色](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteServiceLinkedRole](#)中的。

## delete-service-specific-credential

以下代码示例演示了如何使用 delete-service-specific-credential。

### AWS CLI

示例 1：删除请求用户的特定服务凭证

以下 delete-service-specific-credential 示例删除了发出请求的用户的指定服务特定凭证。service-specific-credential-id 是在您创建凭据时提供的，您可以使用 list-service-specific-credentials 命令进行检索。

```
aws iam delete-service-specific-credential \
  --service-specific-credential-id ACCAEXAMPLE123EXAMPLE
```

此命令不生成任何输出。

示例 2：删除指定用户的服务专用凭证

以下 delete-service-specific-credential 示例删除指定用户的指定服务专用凭证。service-specific-credential-id 是在您创建凭据时提供的，您可以使用 list-service-specific-credentials 命令进行检索。

```
aws iam delete-service-specific-credential \
  --user-name sofia \
  --service-specific-credential-id ACCAEXAMPLE123EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodeCommit 用户指南》CodeCommit 中的[创建 HTTPS 连接的 Git 凭证](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteServiceSpecificCredential](#)中的。

## delete-signing-certificate

以下代码示例演示了如何使用 delete-signing-certificate。

### AWS CLI

删除 IAM 用户的签名证书

以下 delete-signing-certificate 命令删除名为 Bob 的 IAM 用户的指定签名证书。

```
aws iam delete-signing-certificate \  
  --user-name Bob \  
  --certificate-id TA7SMP42TDN5Z260BPJE7EXAMPLE
```

此命令不生成任何输出。

要获取签名证书的 ID，请使用 list-signing-certificates 命令。

有关更多信息，请参阅《Amazon EC2 用户指南》中的[管理签名证书](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSigningCertificate](#)中的。

## delete-ssh-public-key

以下代码示例演示了如何使用 delete-ssh-public-key。

### AWS CLI

删除附加到 IAM 用户的 SSH 公钥

以下delete-ssh-public-key命令删除附加到 IAM 用户的指定 SSH 公钥sofia。

```
aws iam delete-ssh-public-key \  
  --user-name sofia \  
  --ssh-public-key-id APKA123456789EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IAM 用户指南 CodeCommit中的[使用 SSH 密钥和 SSH](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSshPublicKey](#)中的。

## delete-user-permissions-boundary

以下代码示例演示了如何使用 delete-user-permissions-boundary。

### AWS CLI

删除 IAM 用户的权限边界

以下 delete-user-permissions-boundary 示例删除了附加到名为 intern 的 IAM 用户的权限边界。要对用户应用权限边界，请使用 put-user-permissions-boundary 命令。

```
aws iam delete-user-permissions-boundary \  
  --user-name intern
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUserPermissionsBoundary](#)中的。

## delete-user-policy

以下代码示例演示了如何使用 delete-user-policy。

### AWS CLI

要从 IAM 用户中删除策略

以下 delete-user-policy 命令可将指定策略从名为 Bob 的 IAM 用户中删除。

```
aws iam delete-user-policy \  
  --user-name Bob \  
  --policy-name ExamplePolicy
```

此命令不生成任何输出。

要获取 IAM 用户的策略列表，请使用 list-user-policies 命令。

有关更多信息，请参阅 [IAM 用户指南中的在您的 AWS 账户中创建AWS IAM 用户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUserPolicy](#)中的。

## delete-user

以下代码示例演示了如何使用 delete-user。

### AWS CLI

要删除 IAM 用户

以下 delete-user 命令可将名为 Bob 的 IAM 用户从当前账户中删除。

```
aws iam delete-user \  
  --user-name Bob
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[删除 IAM 用户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUser](#)中的。

## delete-virtual-mfa-device

以下代码示例演示了如何使用 delete-virtual-mfa-device。

### AWS CLI

删除虚拟 MFA 设备

以下 delete-virtual-mfa-device 命令从当前账户中删除指定的 MFA 设备。

```
aws iam delete-virtual-mfa-device \  
  --serial-number arn:aws:iam::123456789012:mfa/MFATest
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[停用 MFA 设备](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVirtualMfaDevice](#)中的。

## detach-group-policy

以下代码示例演示了如何使用 detach-group-policy。

## AWS CLI

### 从组中分离策略

此示例将从名为 Testers 的组重删除 ARN 为 `arn:aws:iam::123456789012:policy/TesterAccessPolicy` 的托管策略。

```
aws iam detach-group-policy \  
  --group-name Testers \  
  --policy-arn arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM 用户组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachGroupPolicy](#)中的。

## **detach-role-policy**

以下代码示例演示了如何使用 `detach-role-policy`。

## AWS CLI

### 要从角色分离策略

此示例将从名为 FedTesterRole 的角色删除具有 ARN `arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy` 的托管策略。

```
aws iam detach-role-policy \  
  --role-name FedTesterRole \  
  --policy-arn arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[修改角色](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachRolePolicy](#)中的。

## **detach-user-policy**

以下代码示例演示了如何使用 `detach-user-policy`。

## AWS CLI

### 要从用户分离策略

此示例将从用户 Bob 删除具有 ARN `arn:aws:iam::123456789012:policy/TesterPolicy` 的托管策略。

```
aws iam detach-user-policy \  
  --user-name Bob \  
  --policy-arn arn:aws:iam::123456789012:policy/TesterPolicy
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[更改 IAM 用户的权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachUserPolicy](#)中的。

## enable-mfa-device

以下代码示例演示了如何使用 `enable-mfa-device`。

### AWS CLI

#### 启用 MFA 设备

使用 `create-virtual-mfa-device` 命令创建新的虚拟 MFA 设备后，您可以将 MFA 设备分配给用户。以下 `enable-mfa-device` 示例将序列号为 `arn:aws:iam::210987654321:mfa/BobsMFADevice` 的 MFA 设备分配给用户 Bob。该命令还 AWS 通过按顺序包含虚拟 MFA 设备中的前两个代码来与设备同步。

```
aws iam enable-mfa-device \  
  --user-name Bob \  
  --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \  
  --authentication-code1 123456 \  
  --authentication-code2 789012
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[启用虚拟多重身份验证 \(MFA\) 设备](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableMfaDevice](#)中的。

## generate-credential-report

以下代码示例演示了如何使用 generate-credential-report。

### AWS CLI

要生成凭证报告

以下示例尝试为该 AWS 账户生成凭证报告。

```
aws iam generate-credential-report
```

输出：

```
{
  "State": "STARTED",
  "Description": "No report exists. Starting a new report generation task"
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[获取 AWS 账户的凭证报告](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GenerateCredentialReport](#)中的。

## generate-organizations-access-report

以下代码示例演示了如何使用 generate-organizations-access-report。

### AWS CLI

示例 1：为组织中的根用户生成访问报告

以下 generate-organizations-access-report 示例启动后台作业，为组织中的指定根目录创建访问报告。创建报告后，您可以通过运行 get-organizations-access-report 命令来显示报告。

```
aws iam generate-organizations-access-report \
  --entity-path o-4fxmplt198/r-c3xb
```

输出：

```
{
```

```
"JobId": "a8b6c06f-aaa4-8xmp-28bc-81da71836359"
}
```

### 示例 2：为组织中的账户生成访问报告

以下generate-organizations-access-report示例启动后台作业，为组织123456789012中的账户 ID 创建访问报告o-4fxmplt198。创建报告后，您可以通过运行get-organizations-access-report命令来显示报告。

```
aws iam generate-organizations-access-report \
  --entity-path o-4fxmplt198/r-c3xb/123456789012
```

输出：

```
{
  "JobId": "14b6c071-75f6-2xmp-fb77-faf6fb4201d2"
}
```

### 示例 3：为组织中组织单位的账户生成访问报告

以下generate-organizations-access-report示例启动后台作业，为组织234567890123中的组织单位ou-c3xb-lmu7j2yg中的账户 ID 创建访问报告o-4fxmplt198。创建报告后，您可以通过运行get-organizations-access-report命令来显示报告。

```
aws iam generate-organizations-access-report \
  --entity-path o-4fxmplt198/r-c3xb/ou-c3xb-lmu7j2yg/234567890123
```

输出：

```
{
  "JobId": "2eb6c2e6-0xmp-ec04-1425-c937916a64af"
}
```

要获取有关组织中根和组织单位的详细信息，请使用organizations list-roots和organizations list-organizational-units-for-parent命令。

有关更多信息，请参阅 [AWS IAM 用户指南中的在 AWS 使用上次访问的信息时细化权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GenerateOrganizationsAccessReport](#)中的。

## generate-service-last-accessed-details

以下代码示例演示了如何使用 generate-service-last-accessed-details。

### AWS CLI

#### 示例 1：为自定义策略生成服务访问报告

以下 generate-service-last-accessed-details 示例启动后台作业以生成一份报告，其中列出使用名为 intern-boundary 的自定义策略的 IAM 用户和其他实体所访问的服务。创建报告后，您可以通过运行 get-service-last-accessed-details 命令来显示该报告。

```
aws iam generate-service-last-accessed-details \  
  --arn arn:aws:iam::123456789012:policy/intern-boundary
```

输出：

```
{  
  "JobId": "2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc"  
}
```

#### 示例 2：为 AWS 托管 AdministratorAccess 策略生成服务访问报告

以下 generate-service-last-accessed-details 示例启动后台作业以生成一份报告，其中列出了 IAM 用户和其他实体使用 AWS 托管 AdministratorAccess 策略访问的服务。创建报告后，您可以通过运行 get-service-last-accessed-details 命令来显示该报告。

```
aws iam generate-service-last-accessed-details \  
  --arn arn:aws:iam::aws:policy/AdministratorAccess
```

输出：

```
{  
  "JobId": "78b6c2ba-d09e-6xmp-7039-ecde30b26916"  
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[在 AWS 使用上次访问的信息时细化权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GenerateServiceLastAccessedDetails](#)中的。



## get-access-key-last-used

以下代码示例演示了如何使用 `get-access-key-last-used`。

### AWS CLI

要检索上次使用指定访问密钥的时间信息

以下示例将检索上次使用访问密钥 ABCDEXAMPLE 的时间信息。

```
aws iam get-access-key-last-used \
  --access-key-id ABCDEXAMPLE
```

输出：

```
{
  "UserName": "Bob",
  "AccessKeyLastUsed": {
    "Region": "us-east-1",
    "ServiceName": "iam",
    "LastUsedDate": "2015-06-16T22:45:00Z"
  }
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM 用户的访问密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetAccessKeyLastUsed](#) 中的。

## get-account-authorization-details

以下代码示例演示了如何使用 `get-account-authorization-details`。

### AWS CLI

列出 AWS 账户 IAM 用户、群组、角色和策略

以下 `get-account-authorization-details` 命令返回有关 AWS 账户中所有 IAM 用户、群组、角色和策略的信息。

```
aws iam get-account-authorization-details
```

输出：

```
{
  "RoleDetailList": [
    {
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
              "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "RoleId": "AROA1234567890EXAMPLE",
      "CreateDate": "2014-07-30T17:09:20Z",
      "InstanceProfileList": [
        {
          "InstanceProfileId": "AIPA1234567890EXAMPLE",
          "Roles": [
            {
              "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                  {
                    "Sid": "",
                    "Effect": "Allow",
                    "Principal": {
                      "Service": "ec2.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                  }
                ]
              },
              "RoleId": "AROA1234567890EXAMPLE",
              "CreateDate": "2014-07-30T17:09:20Z",
              "RoleName": "EC2role",
              "Path": "/",
              "Arn": "arn:aws:iam::123456789012:role/EC2role"
            }
          ]
        }
      ]
    }
  ]
}
```

```
    ],
    "CreateDate": "2014-07-30T17:09:20Z",
    "InstanceProfileName": "EC2role",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:instance-profile/EC2role"
  }
],
"RoleName": "EC2role",
"Path": "/",
"AttachedManagedPolicies": [
  {
    "PolicyName": "AmazonS3FullAccess",
    "PolicyArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
  },
  {
    "PolicyName": "AmazonDynamoDBFullAccess",
    "PolicyArn": "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess"
  }
],
"RoleLastUsed": {
  "Region": "us-west-2",
  "LastUsedDate": "2019-11-13T17:30:00Z"
},
"RolePolicyList": [],
"Arn": "arn:aws:iam::123456789012:role/EC2role"
}
],
"GroupDetailList": [
  {
    "GroupId": "AIDA1234567890EXAMPLE",
    "AttachedManagedPolicies": {
      "PolicyName": "AdministratorAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    },
    "GroupName": "Admins",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:group/Admins",
    "CreateDate": "2013-10-14T18:32:24Z",
    "GroupPolicyList": []
  },
  {
    "GroupId": "AIDA1234567890EXAMPLE",
    "AttachedManagedPolicies": {
      "PolicyName": "PowerUserAccess",
```

```

        "PolicyArn": "arn:aws:iam::aws:policy/PowerUserAccess"
    },
    "GroupName": "Dev",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:group/Dev",
    "CreateDate": "2013-10-14T18:33:55Z",
    "GroupPolicyList": []
},
{
    "GroupId": "AIDA1234567890EXAMPLE",
    "AttachedManagedPolicies": [],
    "GroupName": "Finance",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:group/Finance",
    "CreateDate": "2013-10-14T18:57:48Z",
    "GroupPolicyList": [
        {
            "PolicyName": "policygen-201310141157",
            "PolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Action": "aws-portal:*",
                        "Sid": "Stmnt1381777017000",
                        "Resource": "*",
                        "Effect": "Allow"
                    }
                ]
            }
        }
    ]
}
],
"UserDetailList": [
    {
        "UserName": "Alice",
        "GroupList": [
            "Admins"
        ],
        "CreateDate": "2013-10-14T18:32:24Z",
        "UserId": "AIDA1234567890EXAMPLE",
        "UserPolicyList": [],
        "Path": "/",
        "AttachedManagedPolicies": [],

```

```
    "Arn": "arn:aws:iam::123456789012:user/Alice"
  },
  {
    "UserName": "Bob",
    "GroupList": [
      "Admins"
    ],
    "CreateDate": "2013-10-14T18:32:25Z",
    "UserId": "AIDA1234567890EXAMPLE",
    "UserPolicyList": [
      {
        "PolicyName": "DenyBillingAndIAMPolicy",
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": {
            "Effect": "Deny",
            "Action": [
              "aws-portal:*",
              "iam:*"
            ],
            "Resource": "*"
          }
        }
      }
    ],
    "Path": "/",
    "AttachedManagedPolicies": [],
    "Arn": "arn:aws:iam::123456789012:user/Bob"
  },
  {
    "UserName": "Charlie",
    "GroupList": [
      "Dev"
    ],
    "CreateDate": "2013-10-14T18:33:56Z",
    "UserId": "AIDA1234567890EXAMPLE",
    "UserPolicyList": [],
    "Path": "/",
    "AttachedManagedPolicies": [],
    "Arn": "arn:aws:iam::123456789012:user/Charlie"
  }
],
"Policies": [
  {
```

```
"PolicyName": "create-update-delete-set-managed-policies",
"CreateDate": "2015-02-06T19:58:34Z",
"AttachmentCount": 1,
"IsAttachable": true,
"PolicyId": "ANPA1234567890EXAMPLE",
"DefaultVersionId": "v1",
"PolicyVersionList": [
  {
    "CreateDate": "2015-02-06T19:58:34Z",
    "VersionId": "v1",
    "Document": {
      "Version": "2012-10-17",
      "Statement": {
        "Effect": "Allow",
        "Action": [
          "iam:CreatePolicy",
          "iam:CreatePolicyVersion",
          "iam>DeletePolicy",
          "iam>DeletePolicyVersion",
          "iam:GetPolicy",
          "iam:GetPolicyVersion",
          "iam>ListPolicies",
          "iam>ListPolicyVersions",
          "iam:SetDefaultPolicyVersion"
        ],
        "Resource": "*"
      }
    },
    "IsDefaultVersion": true
  }
],
"Path": "/",
"Arn": "arn:aws:iam::123456789012:policy/create-update-delete-set-
managed-policies",
"UpdateDate": "2015-02-06T19:58:34Z"
},
{
  "PolicyName": "S3-read-only-specific-bucket",
  "CreateDate": "2015-01-21T21:39:41Z",
  "AttachmentCount": 1,
  "IsAttachable": true,
  "PolicyId": "ANPA1234567890EXAMPLE",
  "DefaultVersionId": "v1",
  "PolicyVersionList": [
```

```
    {
      "CreateDate": "2015-01-21T21:39:41Z",
      "VersionId": "v1",
      "Document": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "s3:Get*",
              "s3:List*"
            ],
            "Resource": [
              "arn:aws:s3:::example-bucket",
              "arn:aws:s3:::example-bucket/*"
            ]
          }
        ]
      },
      "IsDefaultVersion": true
    }
  ],
  "Path": "/",
  "Arn": "arn:aws:iam::123456789012:policy/S3-read-only-specific-bucket",
  "UpdateDate": "2015-01-21T23:39:41Z"
},
{
  "PolicyName": "AmazonEC2FullAccess",
  "CreateDate": "2015-02-06T18:40:15Z",
  "AttachmentCount": 1,
  "IsAttachable": true,
  "PolicyId": "ANPA1234567890EXAMPLE",
  "DefaultVersionId": "v1",
  "PolicyVersionList": [
    {
      "CreateDate": "2014-10-30T20:59:46Z",
      "VersionId": "v1",
      "Document": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Action": "ec2:*",
            "Effect": "Allow",
            "Resource": "*"
          }
        ]
      }
    }
  ]
}
```

```

    },
    {
      "Effect": "Allow",
      "Action": "elasticloadbalancing:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "cloudwatch:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "autoscaling:*",
      "Resource": "*"
    }
  ]
},
  "IsDefaultVersion": true
}
],
  "Path": "/",
  "Arn": "arn:aws:iam::aws:policy/AmazonEC2FullAccess",
  "UpdateDate": "2015-02-06T18:40:15Z"
}
],
  "Marker": "EXAMPLEkakov9BCuUNFDtxWSyFzetYwEx2ADc8dnzfvERF5S6YMvXKx41t6gCl/
eeaCX3Jo94/bKqezEAg8TEVS99EKFLxm3jtbpl25FDWEXAMPLE",
  "IsTruncated": true
}

```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [AWS 安全审核指南](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetAccountAuthorizationDetails](#) 中的。

## get-account-password-policy

以下代码示例演示了如何使用 `get-account-password-policy`。

### AWS CLI

要查看当前账户密码策略

以下 `get-account-password-policy` 命令将显示有关当前账户密码策略的详细信息。



```
aws iam get-account-password-policy
```

输出：

```
{
  "PasswordPolicy": {
    "AllowUsersToChangePassword": false,
    "RequireLowercaseCharacters": false,
    "RequireUppercaseCharacters": false,
    "MinimumPasswordLength": 8,
    "RequireNumbers": true,
    "RequireSymbols": true
  }
}
```

如果没有为账户定义密码策略，命令将返回 `NoSuchEntity` 错误。

有关更多信息，请参阅《AWS IAM 用户指南》中的[为 IAM 用户设置账户密码策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetAccountPasswordPolicy](#)中的。

## get-account-summary

以下代码示例演示了如何使用 `get-account-summary`。

### AWS CLI

要获取当前账户中 IAM 实体使用情况和 IAM 配额的信息

以下 `get-account-summary` 命令将返回账户中当前 IAM 实体使用情况和当前 IAM 实体配额的信息。

```
aws iam get-account-summary
```

输出：

```
{
  "SummaryMap": {
    "UsersQuota": 5000,
    "GroupsQuota": 100,
  }
}
```

```
"InstanceProfiles": 6,
"SigningCertificatesPerUserQuota": 2,
"AccountAccessKeysPresent": 0,
"RolesQuota": 250,
"RolePolicySizeQuota": 10240,
"AccountSigningCertificatesPresent": 0,
"Users": 27,
"ServerCertificatesQuota": 20,
"ServerCertificates": 0,
"AssumeRolePolicySizeQuota": 2048,
"Groups": 7,
"MFADevicesInUse": 1,
"Roles": 3,
"AccountMFAEnabled": 1,
"MFADevices": 3,
"GroupsPerUserQuota": 10,
"GroupPolicySizeQuota": 5120,
"InstanceProfilesQuota": 100,
"AccessKeysPerUserQuota": 2,
"Providers": 0,
"UserPolicySizeQuota": 2048
}
}
```

有关实体限制的更多信息，请参阅 [IAM 用户指南中的 | AWS IAM 和 AWS STS 配额](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetAccountSummary](#) 中的。

## get-context-keys-for-custom-policy

以下代码示例演示了如何使用 `get-context-keys-for-custom-policy`。

### AWS CLI

示例 1：列出作为命令行参数提供的一个或多个自定义 JSON 策略所引用的上下文键

以下 `get-context-keys-for-custom-policy` 命令解析每个提供的策略，并列出这些策略使用的上下文键。使用此命令来确定必须提供哪些上下文键值才能成功使用策略模拟器命令 `simulate-custom-policy` 和 `simulate-custom-policy`。您还可以使用 `get-context-keys-for-custom-policy` 命令检索与 IAM 用户或角色关联的所有策略使用的上下文键列表。以 `file://` 开头的参数值指示命令读取文件的内容，然后使用内容而不是文件名本身作为参数的值。

```
aws iam get-context-keys-for-custom-policy \
  --policy-input-list '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/${aws:username}","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
```

输出：

```
{
  "ContextKeyNames": [
    "aws:username",
    "aws:CurrentTime"
  ]
}
```

示例 2：列出作为文件输入提供的一个或多个自定义 JSON 策略所引用的上下文键

以下 `get-context-keys-for-custom-policy` 命令与前面的示例相同，只是策略是在文件中提供而不是作为参数提供。由于该命令需要 JSON 字符串列表而不是 JSON 结构列表，因此尽管您可以将其折叠成一个，但该文件的结构必须如下所示。

```
[
  "Policy1",
  "Policy2"
]
```

例如，包含上一个示例中策略的文件必须如下所示。您必须在策略字符串中每个嵌入的双引号前面加上反斜杠 `"` 来对其进行转义。

```
[ "{\"Version\": \"2012-10-17\", \"Statement\": {\"Effect\": \"Allow\", \"Action
\": \"dynamodb:*\", \"Resource\": \"arn:aws:dynamodb:us-west-2:128716708097:table/
${aws:username}\", \"Condition\": {\"DateGreaterThan\": {\"aws:CurrentTime\":
\"2015-08-16T12:00:00Z\"}}}}" ]
```

然后，可以将此文件提交给以下命令。

```
aws iam get-context-keys-for-custom-policy \
  --policy-input-list file://policyfile.json
```

输出：

```
{
  "ContextKeyNames": [
    "aws:username",
    "aws:CurrentTime"
  ]
}
```

有关更多信息，请参阅 [IAM 用户指南中的使用 IAM 策略模拟器 \( AWS CLI 和 AWS API \)](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetContextKeysForCustomPolicy](#) 中的。

## get-context-keys-for-principal-policy

以下代码示例演示了如何使用 `get-context-keys-for-principal-policy`。

### AWS CLI

列出与 IAM 主体关联的所有策略引用的上下文键

以下 `get-context-keys-for-principal-policy` 命令检索附加到用户 `saanvi` 及其所属任何组的所有策略。然后，该命令会解析每个策略并列出这些策略使用的上下文键。使用此命令来确定必须提供哪些上下文键值才能成功使用 `simulate-custom-policy` 和 `simulate-principal-policy` 命令。您还可以使用 `get-context-keys-for-custom-policy` 命令检索任意 JSON 策略使用的上下文键列表。

```
aws iam get-context-keys-for-principal-policy \
  --policy-source-arn arn:aws:iam::123456789012:user/saanvi
```

输出：

```
{
  "ContextKeyNames": [
    "aws:username",
    "aws:CurrentTime"
  ]
}
```

有关更多信息，请参阅 [IAM 用户指南中的使用 IAM 策略模拟器 \( AWS CLI 和 AWS API \)](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetContextKeysForPrincipalPolicy](#) 中的。

## get-credential-report

以下代码示例演示了如何使用 `get-credential-report`。

### AWS CLI

要获取凭证报告

此示例打开返回的报告，并将其作为文本行数组输出到管道。

```
aws iam get-credential-report
```

输出：

```
{
  "GeneratedTime": "2015-06-17T19:11:50Z",
  "ReportFormat": "text/csv"
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[获取 AWS 账户的凭证报告](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetCredentialReport](#)中的。

## get-group-policy

以下代码示例演示了如何使用 `get-group-policy`。

### AWS CLI

获取有关附加到 IAM 组的策略的信息

以下 `get-group-policy` 命令获取有关附加到名为 `Test-Group` 的组的指定策略的信息。

```
aws iam get-group-policy \
  --group-name Test-Group \
  --policy-name S3-ReadOnly-Policy
```

输出：

```
{
```

```
"GroupName": "Test-Group",
"PolicyDocument": {
  "Statement": [
    {
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
},
"PolicyName": "S3-ReadOnly-Policy"
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM policy](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetGroupPolicy](#) 中的。

## get-group

以下代码示例演示了如何使用 get-group。

### AWS CLI

#### 获取 IAM 组

此示例返回有关 IAM 组 Admins 的详细信息。

```
aws iam get-group \
  --group-name Admins
```

输出：

```
{
  "Group": {
    "Path": "/",
    "CreateDate": "2015-06-16T19:41:48Z",
    "GroupId": "AIDGPM9R04H3FEXAMPLE",
    "Arn": "arn:aws:iam::123456789012:group/Admins",
    "GroupName": "Admins"
  }
}
```

```
  },
  "Users": []
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 身份 \(用户、用户组和角色\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetGroup](#) 中的。

## get-instance-profile

以下代码示例演示了如何使用 `get-instance-profile`。

### AWS CLI

获取有关实例配置文件的信息

以下 `get-instance-profile` 命令可获取名为 `ExampleInstanceProfile` 的实例配置文件的信息。

```
aws iam get-instance-profile \
  --instance-profile-name ExampleInstanceProfile
```

输出：

```
{
  "InstanceProfile": {
    "InstanceProfileId": "AID2MAB8DPLSRHEXAMPLE",
    "Roles": [
      {
        "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
        "RoleId": "AIDGPMS9R04H3FEXAMPLE",
        "CreateDate": "2013-01-09T06:33:26Z",
        "RoleName": "Test-Role",
        "Path": "/",
        "Arn": "arn:aws:iam::336924118301:role/Test-Role"
      }
    ],
    "CreateDate": "2013-06-12T23:52:02Z",
    "InstanceProfileName": "ExampleInstanceProfile",
    "Path": "/",
    "Arn": "arn:aws:iam::336924118301:instance-profile/ExampleInstanceProfile"
  }
}
```

```
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[使用实例配置文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInstanceProfile](#)中的。

## get-login-profile

以下代码示例演示了如何使用 get-login-profile。

### AWS CLI

获取 IAM 用户的密码信息

以下 get-login-profile 命令可获取名为 Bob 的 IAM 用户的密码相关信息。

```
aws iam get-login-profile \  
  --user-name Bob
```

输出：

```
{  
  "LoginProfile": {  
    "UserName": "Bob",  
    "CreateDate": "2012-09-21T23:03:39Z"  
  }  
}
```

get-login-profile 命令可用于验证 IAM 用户是否有密码。如果没有为用户定义密码，则该命令将返回 NoSuchEntity 错误。

您无法使用此命令查看密码。如果密码丢失，则可以为用户重置密码 ( update-login-profile )。或者，您可以删除用户的登录配置文件 ( delete-login-profile )，然后创建新的登录配置文件 ( create-login-profile )。

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM 用户的密码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLoginProfile](#)中的。

## get-mfa-device

以下代码示例演示了如何使用 get-mfa-device。



## AWS CLI

检索有关 FIDO 安全密钥的信息

以下 `get-mfa-device` 命令示例检索有关指定 FIDO 安全密钥的信息。

```
aws iam get-mfa-device \
  --serial-number arn:aws:iam::123456789012:u2f/user/alice/fidokeyname-
  EXAMPLEBN5FHTECLFG7EXAMPLE
```

输出：

```
{
  "UserName": "alice",
  "SerialNumber": "arn:aws:iam::123456789012:u2f/user/alice/fidokeyname-
  EXAMPLEBN5FHTECLFG7EXAMPLE",
  "EnableDate": "2023-09-19T01:49:18+00:00",
  "Certifications": {
    "FIDO": "L1"
  }
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[在 AWS 中使用多重身份验证 \(MFA\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetMfaDevice](#) 中的。

## get-open-id-connect-provider

以下代码示例演示了如何使用 `get-open-id-connect-provider`。

## AWS CLI

返回有关指定 OpenID Connect 提供者的信息

此示例返回有关 ARN 为 `arn:aws:iam::123456789012:oidc-provider/server.example.com` 的 OpenID Connect 提供者的详细信息。

```
aws iam get-open-id-connect-provider \
  --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
  server.example.com
```

输出：

```
{
  "Url": "server.example.com"
  "CreateDate": "2015-06-16T19:41:48Z",
  "ThumbprintList": [
    "12345abcdefghijkl67890lmnopqrst987example"
  ],
  "ClientIDList": [
    "example-application-ID"
  ]
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [创建 OpenID Connect \( OIDC \) 身份提供者](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetOpenIdConnectProvider](#) 中的。

## get-organizations-access-report

以下代码示例演示了如何使用 get-organizations-access-report。

### AWS CLI

#### 检索访问报告

以下 get-organizations-access-report 示例显示了之前为 Organ AWS izations 实体生成的访问报告。要生成报告，请使用 generate-organizations-access-report 命令。

```
aws iam get-organizations-access-report \
  --job-id a8b6c06f-aaa4-8xmp-28bc-81da71836359
```

输出：

```
{
  "JobStatus": "COMPLETED",
  "JobCreationDate": "2019-09-30T06:53:36.187Z",
  "JobCompletionDate": "2019-09-30T06:53:37.547Z",
  "NumberOfServicesAccessible": 188,
  "NumberOfServicesNotAccessed": 171,
  "AccessDetails": [
    {
```

```
        "ServiceName": "Alexa for Business",
        "ServiceNamespace": "a4b",
        "TotalAuthenticatedEntities": 0
    },
    ...
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[在 AWS 使用上次访问的信息时细化权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetOrganizationsAccessReport](#)中的。

## get-policy-version

以下代码示例演示了如何使用 get-policy-version。

### AWS CLI

要检索有关指定托管策略的指定版本的信息

此示例将返回 ARN 为 `arn:aws:iam::123456789012:policy/MyManagedPolicy` 的策略 v2 版本的策略文档。

```
aws iam get-policy-version \
  --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
  --version-id v2
```

输出：

```
{
  "PolicyVersion": {
    "Document": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": "iam:*",
          "Resource": "*"
        }
      ]
    },
    "VersionId": "v2",
    "IsDefaultVersion": true,
  }
}
```

```
    "CreateDate": "2023-04-11T00:22:54+00:00"  
  }  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPolicyVersion](#)中的。

## get-policy

以下代码示例演示了如何使用 get-policy。

### AWS CLI

要检索有关指定托管策略的信息

此示例将返回 ARN 为 `arn:aws:iam::123456789012:policy/MySamplePolicy` 的托管策略的详细信息。

```
aws iam get-policy \  
  --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

输出：

```
{  
  "Policy": {  
    "PolicyName": "MySamplePolicy",  
    "CreateDate": "2015-06-17T19:23:32Z",  
    "AttachmentCount": 0,  
    "IsAttachable": true,  
    "PolicyId": "Z27SI6FQMGNQ2EXAMPLE1",  
    "DefaultVersionId": "v1",  
    "Path": "/",  
    "Arn": "arn:aws:iam::123456789012:policy/MySamplePolicy",  
    "UpdateDate": "2015-06-17T19:23:32Z"  
  }  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPolicy](#)中的。

## get-role-policy

以下代码示例演示了如何使用 `get-role-policy`。

### AWS CLI

获取有关附加到 IAM 角色的策略的信息

以下 `get-role-policy` 命令获取有关附加到名为 `Test-Role` 的角色的指定策略的信息。

```
aws iam get-role-policy \
  --role-name Test-Role \
  --policy-name ExamplePolicy
```

输出：

```
{
  "RoleName": "Test-Role",
  "PolicyDocument": {
    "Statement": [
      {
        "Action": [
          "s3:ListBucket",
          "s3:Put*",
          "s3:Get*",
          "s3:*MultipartUpload*"
        ],
        "Resource": "*",
        "Effect": "Allow",
        "Sid": "1"
      }
    ]
  }
  "PolicyName": "ExamplePolicy"
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 IAM 角色](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetRolePolicy](#)中的。

## get-role

以下代码示例演示了如何使用 `get-role`。

## AWS CLI

要获取有关 IAM 角色的信息

以下 `get-role` 命令可获取名为 `Test-Role` 的角色的信息。

```
aws iam get-role \  
  --role-name Test-Role
```

输出：

```
{  
  "Role": {  
    "Description": "Test Role",  
    "AssumeRolePolicyDocument": "<URL-encoded-JSON>",  
    "MaxSessionDuration": 3600,  
    "RoleId": "ARO1234567890EXAMPLE",  
    "CreateDate": "2019-11-13T16:45:56Z",  
    "RoleName": "Test-Role",  
    "Path": "/",  
    "RoleLastUsed": {  
      "Region": "us-east-1",  
      "LastUsedDate": "2019-11-13T17:14:00Z"  
    },  
    "Arn": "arn:aws:iam::123456789012:role/Test-Role"  
  }  
}
```

该命令会显示附加到角色的信任策略。要列出附加到角色的权限策略，请使用 `list-role-policies` 命令。

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 IAM 角色](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetRole](#)中的。

## get-saml-provider

以下代码示例演示了如何使用 `get-saml-provider`。

## AWS CLI

检索 SAML 提供者元文档

此示例检索有关 ARN 为 `arn:aws:iam::123456789012:saml-provider/SAMLADFS` 的 SAML 2.0 提供者的详细信息。响应包括您从身份提供商那里获得的用于创建 AWS SAML 提供商实体的元数据文档以及创建日期和到期日期。

```
aws iam get-saml-provider \  
  --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

输出：

```
{  
  "SAMLMetadataDocument": "...SAMLMetadataDocument-XML...",  
  "CreateDate": "2017-03-06T22:29:46+00:00",  
  "ValidUntil": "2117-03-06T22:29:46.433000+00:00",  
  "Tags": [  
    {  
      "Key": "DeptID",  
      "Value": "123456"  
    },  
    {  
      "Key": "Department",  
      "Value": "Accounting"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 IAM SAML 身份提供者](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetSamlProvider](#)中的。

## get-server-certificate

以下代码示例演示了如何使用 `get-server-certificate`。

### AWS CLI

获取有关您 AWS 账户中服务器证书的详细信息

以下 `get-server-certificate` 命令检索有关您 AWS 账户中指定服务器证书的所有详细信息。

```
aws iam get-server-certificate \  
  --server-certificate-name myUpdatedServerCertificate
```

输出：

```
{
  "ServerCertificate": {
    "ServerCertificateMetadata": {
      "Path": "/",
      "ServerCertificateName": "myUpdatedServerCertificate",
      "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:server-certificate/myUpdatedServerCertificate",
      "UploadDate": "2019-04-22T21:13:44+00:00",
      "Expiration": "2019-10-15T22:23:16+00:00"
    },
    "CertificateBody": "-----BEGIN CERTIFICATE-----
MIICiTCcAFICcQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAKGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBA5TC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWx1eHAd
BgkqhkiG9w0BCQEWEG5vb251QGFTYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAKGA1UEBhMCMVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC0lBTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWx1eHAdBgkqhkiG9w0BCQEWEG5vb251QGFT
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySwTc2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUHVxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvjx79LjStb
NYiytVbZPQUQ5Yaxu2jXnimvrszlaEXAMPLE=-----END CERTIFICATE-----",
    "CertificateChain": "-----BEGIN CERTIFICATE-----\nMIICiTCcAFICcQD6md
7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAKGA1UEBhMCMVVMxCzAJBgNVBAGT
AlldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5
TC0lBTSBDb25zb2x1MRIwEAYDVQsQQDEw1UZXR0Q21sYWx1eHAdBgkqhkiG9w0BCQ
jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAKGA1UEBh
MCMVVMxCzAJBgNVBAGTAldBMRAwDgsYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBb
WF6b24xFDASBgNVBA5TC0lBTSBDb2d5zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWx1
eHAdBgkqhkiG9w0BCQEWEG5vb251QGFTYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIgWJ21uUSfwfEvySwTc2XADZ4nB+BLyGVI
k60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
ITx0USQv7c7ugFFDzQGBzZswY6786m86gjpEIbb30hjZnzcvcQAaRHhd1QWIMm2nr
AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCku4nUHVxYUntneD9+h8Mg9q6q+auN
KyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0F1kbFFBjvSfpJI1J00zbhNYS5f6Guo
EDmFJl0ZxBHjJnyp3780D8uTs7fLvjx79LjS;TbNYiytVbZPQUQ5Yaxu2jXnimvw
3rrszlaEWEG5vb251QGFTsYXpvbiEXAMPLE=\n-----END CERTIFICATE-----"
  }
}
```



```
}
```

要列出您 AWS 账户中可用的服务器证书，请使用 `list-server-certificates` 命令。

有关更多信息，请参阅《AWS IAM 用户指南》中的[在 IAM 中管理服务器证书](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetServerCertificate](#) 中的。

## `get-service-last-accessed-details-with-entities`

以下代码示例演示了如何使用 `get-service-last-accessed-details-with-entities`。

### AWS CLI

检索包含服务详细信息的服务器访问报告

以下 `get-service-last-accessed-details-with-entities` 示例检索一份报告，其中包含有关访问指定服务的 IAM 用户和其他实体的详细信息。要生成报告，请使用 `generate-service-last-accessed-details` 命令。要获取使用命名空间访问的服务列表，请使用 `get-service-last-accessed-details`。

```
aws iam get-service-last-accessed-details-with-entities \  
  --job-id 78b6c2ba-d09e-6xmp-7039-ecde30b26916 \  
  --service-namespace lambda
```

输出：

```
{  
  "JobStatus": "COMPLETED",  
  "JobCreationDate": "2019-10-01T03:55:41.756Z",  
  "JobCompletionDate": "2019-10-01T03:55:42.533Z",  
  "EntityDetailsList": [  
    {  
      "EntityInfo": {  
        "Arn": "arn:aws:iam::123456789012:user/admin",  
        "Name": "admin",  
        "Type": "USER",  
        "Id": "AIDAI02XMPLENQEXAMPLE",  
        "Path": "/"  
      },  
      "LastAuthenticated": "2019-09-30T23:02:00Z"  
    }  
  ]  
}
```

```
    },
    {
      "EntityInfo": {
        "Arn": "arn:aws:iam::123456789012:user/developer",
        "Name": "developer",
        "Type": "USER",
        "Id": "AIDAIBEYXMPL2YEXAMPLE",
        "Path": "/"
      },
      "LastAuthenticated": "2019-09-16T19:34:00Z"
    }
  ]
}
```

有关更多信息，请参阅 [AWS IAM 用户指南中的在 AWS 使用上次访问的信息时细化权限](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `GetServiceLastAccessedDetailsWithEntities` 中的](#)。

## get-service-last-accessed-details

以下代码示例演示了如何使用 `get-service-last-accessed-details`。

### AWS CLI

#### 检索服务访问报告

以下 `get-service-last-accessed-details` 示例检索之前生成的报告，其中列出了 IAM 实体所访问的服务。要生成报告，请使用 `generate-service-last-accessed-details` 命令。

```
aws iam get-service-last-accessed-details \
  --job-id 2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc
```

输出：

```
{
  "JobStatus": "COMPLETED",
  "JobCreationDate": "2019-10-01T03:50:35.929Z",
  "ServicesLastAccessed": [
    ...
    {
      "ServiceName": "AWS Lambda",
```

```
    "LastAuthenticated": "2019-09-30T23:02:00Z",
    "ServiceNamespace": "lambda",
    "LastAuthenticatedEntity": "arn:aws:iam::123456789012:user/admin",
    "TotalAuthenticatedEntities": 6
  },
]
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[在 AWS 使用上次访问的信息时细化权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetServiceLastAccessedDetails](#)中的。

## get-service-linked-role-deletion-status

以下代码示例演示了如何使用 `get-service-linked-role-deletion-status`。

### AWS CLI

要查看删除服务相关角色的请求状态

以下 `get-service-linked-role-deletion-status` 示例演示了先前请求删除服务相关角色的状态。删除操作异步进行。当您发出请求时，您将得到一个 `DeletionTaskId` 值，该值将作为此命令的参数提供。

```
aws iam get-service-linked-role-deletion-status \
  --deletion-task-id task/aws-service-role/lex.amazonaws.com/
  AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE
```

输出：

```
{
  "Status": "SUCCEEDED"
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[使用服务相关角色](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetServiceLinkedRoleDeletionStatus](#)中的。

## get-ssh-public-key

以下代码示例演示了如何使用 `get-ssh-public-key`。

## AWS CLI

示例 1：检索以 SSH 编码形式附加到 IAM 用户的 SSH 公钥

以下`get-ssh-public-key`命令从 IAM 用户`sofia`那里检索指定的 SSH 公钥。输出采用 SSH 编码。

```
aws iam get-ssh-public-key \  
  --user-name sofia \  
  --ssh-public-key-id APKA123456789EXAMPLE \  
  --encoding SSH
```

输出：

```
{  
  "SSHPublicKey": {  
    "UserName": "sofia",  
    "SSHPublicKeyId": "APKA123456789EXAMPLE",  
    "Fingerprint": "12:34:56:78:90:ab:cd:ef:12:34:56:78:90:ab:cd:ef",  
    "SSHPublicKeyBody": "ssh-rsa <<long encoded SSH string>>",  
    "Status": "Inactive",  
    "UploadDate": "2019-04-18T17:04:49+00:00"  
  }  
}
```

示例 2：检索以 PEM 编码形式附加到 IAM 用户的 SSH 公钥

以下`get-ssh-public-key`命令从 IAM 用户`sofia`那里检索指定的 SSH 公钥。输出采用 PEM 编码。

```
aws iam get-ssh-public-key \  
  --user-name sofia \  
  --ssh-public-key-id APKA123456789EXAMPLE \  
  --encoding PEM
```

输出：

```
{  
  "SSHPublicKey": {  
    "UserName": "sofia",
```

```
    "SSHPublicKeyId": "APKA123456789EXAMPLE",
    "Fingerprint": "12:34:56:78:90:ab:cd:ef:12:34:56:78:90:ab:cd:ef",
    "SSHPublicKeyBody": ""-----BEGIN PUBLIC KEY-----\n<<long encoded PEM
string>>\n-----END PUBLIC KEY-----\n"",
    "Status": "Inactive",
    "UploadDate": "2019-04-18T17:04:49+00:00"
  }
}
```

有关更多信息，请参阅 AWS IAM 用户指南 CodeCommit 中的 [使用 SSH 密钥和 SSH](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetSshPublicKey](#) 中的。

## get-user-policy

以下代码示例演示了如何使用 get-user-policy。

### AWS CLI

列出 IAM 用户的策略详细信息

以下 get-user-policy 命令将列出附加到名为 Bob 的 IAM 用户的指定策略的详细信息。

```
aws iam get-user-policy \
  --user-name Bob \
  --policy-name ExamplePolicy
```

输出：

```
{
  "UserName": "Bob",
  "PolicyName": "ExamplePolicy",
  "PolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "*",
        "Resource": "*",
        "Effect": "Allow"
      }
    ]
  }
}
```

```
}
```

要获取 IAM 用户的策略列表，请使用 `list-user-policies` 命令。

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetUserPolicy](#) 中的。

## get-user

以下代码示例演示了如何使用 `get-user`。

### AWS CLI

要获取有关 IAM 用户的信息

以下 `get-user` 命令可获取名为 Paulo 的 IAM 用户的信息。

```
aws iam get-user \  
  --user-name Paulo
```

输出：

```
{  
  "User": {  
    "UserName": "Paulo",  
    "Path": "/",  
    "CreateDate": "2019-09-21T23:03:13Z",  
    "UserId": "AIDA123456789EXAMPLE",  
    "Arn": "arn:aws:iam::123456789012:user/Paulo"  
  }  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [管理 IAM 用户](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetUser](#) 中的。

## list-access-keys

以下代码示例演示了如何使用 `list-access-keys`。

## AWS CLI

要列出 IAM 用户的访问密钥 ID

以下 `list-access-keys` 命令将列出名为 Bob 的 IAM 用户的访问密钥 ID。

```
aws iam list-access-keys \  
  --user-name Bob
```

输出：

```
{  
  "AccessKeyMetadata": [  
    {  
      "UserName": "Bob",  
      "Status": "Active",  
      "CreateDate": "2013-06-04T18:17:34Z",  
      "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"  
    },  
    {  
      "UserName": "Bob",  
      "Status": "Inactive",  
      "CreateDate": "2013-06-06T20:42:26Z",  
      "AccessKeyId": "AKIAI44QH8DHBEXAMPLE"  
    }  
  ]  
}
```

无法列出 IAM 用户的秘密访问密钥。如果秘密访问密钥丢失，则必须使用 `create-access-keys` 命令创建新的访问密钥。

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM 用户的访问密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAccessKeys](#) 中的。

## **list-account-aliases**

以下代码示例演示了如何使用 `list-account-aliases`。

## AWS CLI

要列出账户别名

以下 `list-account-aliases` 命令将列出当前账户的别名。

```
aws iam list-account-aliases
```

输出：

```
{
  "AccountAliases": [
    "mycompany"
  ]
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[您的 AWS 账户 ID 及其别名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListAccountAliases](#)中的。

## list-attached-group-policies

以下代码示例演示了如何使用 `list-attached-group-policies`。

AWS CLI

列出附加到指定组的所有托管策略

此示例返回挂载到 AWS 账户中名为 Admins 的 IAM 组的托管策略的名称和 ARN。

```
aws iam list-attached-group-policies \
  --group-name Admins
```

输出：

```
{
  "AttachedPolicies": [
    {
      "PolicyName": "AdministratorAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    },
    {
      "PolicyName": "SecurityAudit",
      "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
    }
  ]
}
```



```
  ],
  "IsTruncated": false
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAttachedGroupPolicies](#)中的。

## list-attached-role-policies

以下代码示例演示了如何使用 list-attached-role-policies。

### AWS CLI

要列出附加到指定角色的所有托管策略

此命令返回挂载到 AWS 账户中命名的 IAM 角色的托管策略SecurityAuditRole的名称和ARN。

```
aws iam list-attached-role-policies \
  --role-name SecurityAuditRole
```

输出：

```
{
  "AttachedPolicies": [
    {
      "PolicyName": "SecurityAudit",
      "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
    }
  ],
  "IsTruncated": false
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAttachedRolePolicies](#)中的。

## list-attached-user-policies

以下代码示例演示了如何使用 list-attached-user-policies。

## AWS CLI

列出附加到指定用户的所有托管策略

此命令返回 AWS 账户中指定Bob的 IAM 用户的托管策略的名称和 ARN。

```
aws iam list-attached-user-policies \  
  --user-name Bob
```

输出：

```
{  
  "AttachedPolicies": [  
    {  
      "PolicyName": "AdministratorAccess",  
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"  
    },  
    {  
      "PolicyName": "SecurityAudit",  
      "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"  
    }  
  ],  
  "IsTruncated": false  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAttachedUserPolicies](#)中的。

## list-entities-for-policy

以下代码示例演示了如何使用 list-entities-for-policy。

## AWS CLI

列出指定托管策略所附加到的所有用户、组和角色

此示例返回附加了策略 arn:aws:iam::123456789012:policy/TestPolicy 的 IAM 组、角色和用户的列表。

```
aws iam list-entities-for-policy \  
  --policy-arn arn:aws:iam::123456789012:policy/TestPolicy
```

```
--policy-arn arn:aws:iam::123456789012:policy/TestPolicy
```

输出：

```
{
  "PolicyGroups": [
    {
      "GroupName": "Admins",
      "GroupId": "AGPACKCEVSQ6C2EXAMPLE"
    }
  ],
  "PolicyUsers": [
    {
      "UserName": "Alice",
      "UserId": "AIDACKCEVSQ6C2EXAMPLE"
    }
  ],
  "PolicyRoles": [
    {
      "RoleName": "DevRole",
      "RoleId": "AR0ADBQP57FF2AEXAMPLE"
    }
  ],
  "IsTruncated": false
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListEntitiesForPolicy](#) 中的。

## list-group-policies

以下代码示例演示了如何使用 list-group-policies。

AWS CLI

列出附加到指定组的所有内联策略

以下 list-group-policies 命令列出附加到当前文档中名为 Admins 的 IAM 组的内联策略名称。

```
aws iam list-group-policies \
```

```
--group-name Admins
```

输出：

```
{
  "PolicyNames": [
    "AdminRoot",
    "ExamplePolicy"
  ]
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM policy](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListGroupPolicies](#)中的。

## list-groups-for-user

以下代码示例演示了如何使用 `list-groups-for-user`。

### AWS CLI

列出 IAM 用户所属的组

以下 `list-groups-for-user` 命令显示名为 Bob 的 IAM 用户所属的组。

```
aws iam list-groups-for-user \
  --user-name Bob
```

输出：

```
{
  "Groups": [
    {
      "Path": "/",
      "CreateDate": "2013-05-06T01:18:08Z",
      "GroupId": "AKIAIOSFODNN7EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:group/Admin",
      "GroupName": "Admin"
    },
    {
      "Path": "/",
      "CreateDate": "2013-05-06T01:37:28Z",
```

```
    "GroupId": "AKIAI44QH8DHBEXAMPLE",
    "Arn": "arn:aws:iam::123456789012:group/s3-Users",
    "GroupName": "s3-Users"
  }
]
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM 用户组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListGroupsForUser](#)中的。

## list-groups

以下代码示例演示了如何使用 list-groups。

### AWS CLI

要列出当前账户的 IAM 组

以下 list-groups 命令将列出当前账户中的 IAM 组。

```
aws iam list-groups
```

输出：

```
{
  "Groups": [
    {
      "Path": "/",
      "CreateDate": "2013-06-04T20:27:27.972Z",
      "GroupId": "AIDACKCEVSQ6C2EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:group/Admins",
      "GroupName": "Admins"
    },
    {
      "Path": "/",
      "CreateDate": "2013-04-16T20:30:42Z",
      "GroupId": "AIDGPMS9R04H3FEXAMPLE",
      "Arn": "arn:aws:iam::123456789012:group/S3-Admins",
      "GroupName": "S3-Admins"
    }
  ]
}
```

```
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM 用户组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListGroups](#)中的。

## list-instance-profile-tags

以下代码示例演示了如何使用 `list-instance-profile-tags`。

### AWS CLI

列出附加到实例配置文件的标签

以下`list-instance-profile-tags`命令检索与指定实例配置文件关联的标签列表。

```
aws iam list-instance-profile-tags \  
  --instance-profile-name deployment-role
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "DeptID",  
      "Value": "123456"  
    },  
    {  
      "Key": "Department",  
      "Value": "Accounting"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListInstanceProfileTags](#)中的。

## list-instance-profiles-for-role

以下代码示例演示了如何使用 `list-instance-profiles-for-role`。

## AWS CLI

### 列出 IAM 角色的实例配置文件

以下 `list-instance-profiles-for-role` 命令列出了中与角色 `Test-Role` 关联的实例配置文件。

```
aws iam list-instance-profiles-for-role \  
  --role-name Test-Role
```

输出：

```
{  
  "InstanceProfiles": [  
    {  
      "InstanceId": "AIDGPM59R04H3FEXAMPLE",  
      "Roles": [  
        {  
          "AssumeRolePolicyDocument": "<URL-encoded-JSON>",  
          "RoleId": "AIDACKCEVSQ6C2EXAMPLE",  
          "CreateDate": "2013-06-07T20:42:15Z",  
          "RoleName": "Test-Role",  
          "Path": "/",  
          "Arn": "arn:aws:iam::123456789012:role/Test-Role"  
        }  
      ],  
      "CreateDate": "2013-06-07T21:05:24Z",  
      "InstanceProfileName": "ExampleInstanceProfile",  
      "Path": "/",  
      "Arn": "arn:aws:iam::123456789012:instance-profile/  
ExampleInstanceProfile"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[使用实例配置文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListInstanceProfilesForRole](#)中的。

## list-instance-profiles

以下代码示例演示了如何使用 `list-instance-profiles`。

## AWS CLI

列出账户的实例配置文件

以下 `list-instance-profiles` 命令列出与当前账户关联的实例配置文件。

```
aws iam list-instance-profiles
```

输出：

```
{
  "InstanceProfiles": [
    {
      "Path": "/",
      "InstanceProfileName": "example-dev-role",
      "InstanceProfileId": "AIPAIXEU4NUHUPEXAMPLE",
      "Arn": "arn:aws:iam::123456789012:instance-profile/example-dev-role",
      "CreateDate": "2023-09-21T18:17:41+00:00",
      "Roles": [
        {
          "Path": "/",
          "RoleName": "example-dev-role",
          "RoleId": "AROAJ520TH4H7LEXAMPLE",
          "Arn": "arn:aws:iam::123456789012:role/example-dev-role",
          "CreateDate": "2023-09-21T18:17:40+00:00",
          "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
              {
                "Effect": "Allow",
                "Principal": {
                  "Service": "ec2.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
              }
            ]
          }
        }
      ]
    },
    {
      "Path": "/",
      "InstanceProfileName": "example-s3-role",
```



```

    "InstanceProfileId": "AIPAJVJVNRIQFREXAMPLE",
    "Arn": "arn:aws:iam::123456789012:instance-profile/example-s3-role",
    "CreateDate": "2023-09-21T18:18:50+00:00",
    "Roles": [
      {
        "Path": "/",
        "RoleName": "example-s3-role",
        "RoleId": "AROAINUBC507XLEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:role/example-s3-role",
        "CreateDate": "2023-09-21T18:18:49+00:00",
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "ec2.amazonaws.com"
              },
              "Action": "sts:AssumeRole"
            }
          ]
        }
      }
    ]
  }
]
}

```

有关更多信息，请参阅《AWS IAM 用户指南》中的[使用实例配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListInstanceProfiles](#)中的。

## list-mfa-device-tags

以下代码示例演示了如何使用 `list-mfa-device-tags`。

### AWS CLI

列出附加到 MFA 设备的标签

以下 `list-mfa-device-tags` 命令检索与指定 MFA 设备关联的标签列表。

```
aws iam list-mfa-device-tags \
```

```
--serial-number arn:aws:iam::123456789012:mfa/alice
```

输出：

```
{
  "Tags": [
    {
      "Key": "DeptID",
      "Value": "123456"
    },
    {
      "Key": "Department",
      "Value": "Accounting"
    }
  ]
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListMfaDeviceTags](#)中的。

## list-mfa-devices

以下代码示例演示了如何使用 list-mfa-devices。

### AWS CLI

列出指定用户的所有 MFA 设备

此示例返回有关分配给 IAM 用户 Bob 的 MFA 设备的详细信息。

```
aws iam list-mfa-devices \
  --user-name Bob
```

输出：

```
{
  "MFADevices": [
    {
      "UserName": "Bob",
      "SerialNumber": "arn:aws:iam::123456789012:mfa/Bob",
      "EnableDate": "2019-10-28T20:37:09+00:00"
    }
  ]
}
```

```

    },
    {
      "UserName": "Bob",
      "SerialNumber": "GAKT12345678",
      "EnableDate": "2023-02-18T21:44:42+00:00"
    },
    {
      "UserName": "Bob",
      "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey1-7XNL7NFNLZ123456789EXAMPLE",
      "EnableDate": "2023-09-19T02:25:35+00:00"
    },
    {
      "UserName": "Bob",
      "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey2-VDRQTDBBN5123456789EXAMPLE",
      "EnableDate": "2023-09-19T01:49:18+00:00"
    }
  ]
}

```

有关更多信息，请参阅《AWS IAM 用户指南》中的[在 AWS 中使用多重身份验证 \(MFA\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListMfaDevices](#) 中的。

## list-open-id-connect-provider-tags

以下代码示例演示了如何使用 `list-open-id-connect-provider-tags`。

### AWS CLI

列出附加到兼容 OpenID Connect (OIDC) 的身份提供商的标签

以下 `list-open-id-connect-provider-tags` 命令检索与指定 OIDC 身份提供商关联的标签列表。

```

aws iam list-open-id-connect-provider-tags \
  --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
server.example.com

```

输出：

```
{
```

```
    "Tags": [
      {
        "Key": "DeptID",
        "Value": "123456"
      },
      {
        "Key": "Department",
        "Value": "Accounting"
      }
    ]
  }
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListOpenIdConnectProviderTags](#)中的。

## list-open-id-connect-providers

以下代码示例演示了如何使用 list-open-id-connect-providers。

### AWS CLI

列出账户中 OpenID Connect 提供商的相关信息 AWS

此示例返回当前账户中定义的所有 OpenID Connect 提供商的 ARN 列表。 AWS

```
aws iam list-open-id-connect-providers
```

输出：

```
{
  "OpenIDConnectProviderList": [
    {
      "Arn": "arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com"
    }
  ]
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [创建 OpenID Connect \( OIDC \) 身份提供者](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListOpenIdConnectProviders](#)中的。

## list-policies-granting-service-access

以下代码示例演示了如何使用 `list-policies-granting-service-access`。

### AWS CLI

列出授予委托人访问指定服务的权限的策略

以下 `list-policies-granting-service-access` 示例检索授予 IAM 用户 `sofia` 访问 AWS CodeCommit 服务的策略列表。

```
aws iam list-policies-granting-service-access \
  --arn arn:aws:iam::123456789012:user/sofia \
  --service-namespaces codecommit
```

输出：

```
{
  "PoliciesGrantingServiceAccess": [
    {
      "ServiceNamespace": "codecommit",
      "Policies": [
        {
          "PolicyName": "Grant-Sofia-Access-To-CodeCommit",
          "PolicyType": "INLINE",
          "EntityType": "USER",
          "EntityName": "sofia"
        }
      ]
    }
  ],
  "IsTruncated": false
}
```

有关更多信息，请参阅 [IAM 用户指南中的将 IAM 与 CodeCommit Git 证书、SSH 密钥和 AWS 访问密钥配合使用](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListPoliciesGrantingServiceAccess](#) 中的。

## list-policies

以下代码示例演示了如何使用 `list-policies`。

## AWS CLI

列出您的 AWS 账户可用的托管政策

此示例返回当前 AWS 账户中可用的前两个托管策略的集合。

```
aws iam list-policies \  
  --max-items 3
```

输出：

```
{  
  "Policies": [  
    {  
      "PolicyName": "AWSCloudTrailAccessPolicy",  
      "PolicyId": "ANPAXQE2B5PJ7YEXAMPLE",  
      "Arn": "arn:aws:iam::123456789012:policy/AWSCloudTrailAccessPolicy",  
      "Path": "/",  
      "DefaultVersionId": "v1",  
      "AttachmentCount": 0,  
      "PermissionsBoundaryUsageCount": 0,  
      "IsAttachable": true,  
      "CreateDate": "2019-09-04T17:43:42+00:00",  
      "UpdateDate": "2019-09-04T17:43:42+00:00"  
    },  
    {  
      "PolicyName": "AdministratorAccess",  
      "PolicyId": "ANPAIWMBCKSKIEE64ZLYK",  
      "Arn": "arn:aws:iam::aws:policy/AdministratorAccess",  
      "Path": "/",  
      "DefaultVersionId": "v1",  
      "AttachmentCount": 6,  
      "PermissionsBoundaryUsageCount": 0,  
      "IsAttachable": true,  
      "CreateDate": "2015-02-06T18:39:46+00:00",  
      "UpdateDate": "2015-02-06T18:39:46+00:00"  
    },  
    {  
      "PolicyName": "PowerUserAccess",  
      "PolicyId": "ANPAJYRXTHIB4FOVS3ZXS",  
      "Arn": "arn:aws:iam::aws:policy/PowerUserAccess",  
      "Path": "/",  
      "DefaultVersionId": "v5",  
      "AttachmentCount": 0,  
      "PermissionsBoundaryUsageCount": 0,  
      "IsAttachable": true,  
      "CreateDate": "2015-02-06T18:39:46+00:00",  
      "UpdateDate": "2015-02-06T18:39:46+00:00"  
    }  
  ]  
}
```

```

        "AttachmentCount": 1,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2015-02-06T18:39:47+00:00",
        "UpdateDate": "2023-07-06T22:04:00+00:00"
    }
],
    "NextToken": "EXAMPLErZXIi0iBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQi0iA4fQ=="
}

```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListPolicies](#) 中的。

## list-policy-tags

以下代码示例演示了如何使用 list-policy-tags。

### AWS CLI

列出附加到托管策略的标签

以下 list-policy-tags 命令检索与指定托管策略关联的标签列表。

```

aws iam list-policy-tags \
    --policy-arn arn:aws:iam::123456789012:policy/billing-access

```

输出：

```

{
  "Tags": [
    {
      "Key": "DeptID",
      "Value": "123456"
    },
    {
      "Key": "Department",
      "Value": "Accounting"
    }
  ]
}

```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPolicyTags](#)中的。

## list-policy-versions

以下代码示例演示了如何使用 list-policy-versions。

### AWS CLI

列出有关指定托管策略版本的信息

此示例返回 ARN 为 arn:aws:iam::123456789012:policy/MySamplePolicy 的策略的可用版本列表。

```
aws iam list-policy-versions \  
  --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

输出：

```
{  
  "IsTruncated": false,  
  "Versions": [  
    {  
      "VersionId": "v2",  
      "IsDefaultVersion": true,  
      "CreateDate": "2015-06-02T23:19:44Z"  
    },  
    {  
      "VersionId": "v1",  
      "IsDefaultVersion": false,  
      "CreateDate": "2015-06-02T22:30:47Z"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPolicyVersions](#)中的。

## list-role-policies

以下代码示例演示了如何使用 list-role-policies。



## AWS CLI

要列出附加到 IAM 角色的策略

以下 `list-role-policies` 命令将列出指定 IAM 角色的权限策略名称。

```
aws iam list-role-policies \  
  --role-name Test-Role
```

输出：

```
{  
  "PolicyNames": [  
    "ExamplePolicy"  
  ]  
}
```

要查看附加到角色的信任策略，请使用 `get-role` 命令。要查看权限策略的详细信息，请使用 `get-role-policy` 命令。

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 IAM 角色](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListRolePolicies](#) 中的。

## list-role-tags

以下代码示例演示了如何使用 `list-role-tags`。

### AWS CLI

列出附加到角色的标签

以下 `list-role-tags` 命令检索与指定角色关联的标签列表。

```
aws iam list-role-tags \  
  --role-name production-role
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "Department",
```

```
        "Value": "Accounting"
      },
      {
        "Key": "DeptID",
        "Value": "12345"
      }
    ],
    "IsTruncated": false
  }
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRoleTags](#)中的。

## list-roles

以下代码示例演示了如何使用 list-roles。

### AWS CLI

要列出当前账户的 IAM 角色

以下 list-roles 命令将列出当前账户中的 IAM 角色。

```
aws iam list-roles
```

输出：

```
{
  "Roles": [
    {
      "Path": "/",
      "RoleName": "ExampleRole",
      "RoleId": "AR0AJ520TH4H7LEXAMPLE",
      "Arn": "arn:aws:iam::123456789012:role/ExampleRole",
      "CreateDate": "2017-09-12T19:23:36+00:00",
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
```

```

        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
},
"MaxSessionDuration": 3600
},
{
  "Path": "/example_path/",
  "RoleName": "ExampleRoleWithPath",
  "RoleId": "AROAI4QRP7UFT7EXAMPLE",
  "Arn": "arn:aws:iam::123456789012:role/example_path/
ExampleRoleWithPath",
  "CreateDate": "2023-09-21T20:29:38+00:00",
  "AssumeRolePolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Allow",
        "Principal": {
          "Service": "ec2.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
      }
    ]
  },
  "MaxSessionDuration": 3600
}
]
}
}

```

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 IAM 角色](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListRoles](#)中的。

## list-saml-provider-tags

以下代码示例演示了如何使用 list-saml-provider-tags。

### AWS CLI

列出附加到 SAML 提供商的标签

以下 `list-saml-provider-tags` 命令检索与指定 SAML 提供程序关联的标签列表。

```
aws iam list-saml-provider-tags \  
  --saml-provider-arn arn:aws:iam::123456789012:saml-provider/ADFS
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "DeptID",  
      "Value": "123456"  
    },  
    {  
      "Key": "Department",  
      "Value": "Accounting"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListSamlProviderTags](#) 中的。

## list-saml-providers

以下代码示例演示了如何使用 `list-saml-providers`。

AWS CLI

列出账户中的 SAML 提供商 AWS

此示例检索在当前 AWS 账户中创建的 SAML 2.0 提供商列表。

```
aws iam list-saml-providers
```

输出：

```
{  
  "SAMLProviderList": [  
    {  
      "Name": "ADFS",  
      "CreateDate": "2017-01-01T00:00:00Z",  
      "LastModifiedDate": "2017-01-01T00:00:00Z",  
      "Type": "SAML_2_0"  
    }  
  ]  
}
```

```
{
  "Arn": "arn:aws:iam::123456789012:saml-provider/SAML-ADFS",
  "ValidUntil": "2015-06-05T22:45:14Z",
  "CreateDate": "2015-06-05T22:45:14Z"
}
]
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 IAM SAML 身份提供者](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的[ListSAMLProviders](#)。

## list-server-certificate-tags

以下代码示例演示了如何使用 `list-server-certificate-tags`。

### AWS CLI

列出附加到服务器证书的标签

以下 `list-server-certificate-tags` 命令检索与指定服务器证书关联的标签列表。

```
aws iam list-server-certificate-tags \
  --server-certificate-name ExampleCertificate
```

输出：

```
{
  "Tags": [
    {
      "Key": "DeptID",
      "Value": "123456"
    },
    {
      "Key": "Department",
      "Value": "Accounting"
    }
  ]
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListServerCertificateTags](#)中的。

## list-server-certificates

以下代码示例演示了如何使用 list-server-certificates。

### AWS CLI

列出您 AWS 账户中的服务器证书

以下list-server-certificates命令列出了您的 AWS 账户中存储并可供使用的所有服务器证书。

```
aws iam list-server-certificates
```

输出：

```
{
  "ServerCertificateMetadataList": [
    {
      "Path": "/",
      "ServerCertificateName": "myUpdatedServerCertificate",
      "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:server-certificate/myUpdatedServerCertificate",
      "UploadDate": "2019-04-22T21:13:44+00:00",
      "Expiration": "2019-10-15T22:23:16+00:00"
    },
    {
      "Path": "/cloudfront/",
      "ServerCertificateName": "MyTestCert",
      "ServerCertificateId": "ASCAEXAMPLE456EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyTestCert",
      "UploadDate": "2015-04-21T18:14:16+00:00",
      "Expiration": "2018-01-14T17:52:36+00:00"
    }
  ]
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[在 IAM 中管理服务器证书](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListServerCertificates](#)中的。

## list-service-specific-credential

以下代码示例演示了如何使用 `list-service-specific-credential`。

### AWS CLI

示例 1：列出用户的服务专用证书

以下`list-service-specific-credentials`示例显示分配给指定用户的所有服务专用凭证。密码未包含在响应中。

```
aws iam list-service-specific-credentials \  
  --user-name sofia
```

输出：

```
{  
  "ServiceSpecificCredential": {  
    "CreateDate": "2019-04-18T20:45:36+00:00",  
    "ServiceName": "codecommit.amazonaws.com",  
    "ServiceUserName": "sofia-at-123456789012",  
    "ServiceSpecificCredentialId": "ACCAEXAMPLE123EXAMPLE",  
    "UserName": "sofia",  
    "Status": "Active"  
  }  
}
```

示例 2：列出筛选到指定服务的用户的服务特定凭证

以下`list-service-specific-credentials`示例显示了分配给发出请求的用户的特定服务凭证。列表经过筛选，仅包含指定服务的凭证。密码未包含在响应中。

```
aws iam list-service-specific-credentials \  
  --service-name codecommit.amazonaws.com
```

输出：

```
{  
  "ServiceSpecificCredential": {
```

```

    "CreateDate": "2019-04-18T20:45:36+00:00",
    "ServiceName": "codecommit.amazonaws.com",
    "ServiceUserName": "sofia-at-123456789012",
    "ServiceSpecificCredentialId": "ACCAEXAMPLE123EXAMPLE",
    "UserName": "sofia",
    "Status": "Active"
  }
}

```

有关更多信息，请参阅《AWS CodeCommit 用户指南》CodeCommit中的[创建 HTTPS 连接的 Git 凭证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListServiceSpecificCredential](#)中的。

## list-service-specific-credentials

以下代码示例演示了如何使用 list-service-specific-credentials。

### AWS CLI

#### 检索凭证列表

以下list-service-specific-credentials示例列出了名为的用户为 HTTPS 访问 AWS CodeCommit 存储库而生成的凭证developer。

```

aws iam list-service-specific-credentials \
  --user-name developer \
  --service-name codecommit.amazonaws.com

```

输出：

```

{
  "ServiceSpecificCredentials": [
    {
      "UserName": "developer",
      "Status": "Inactive",
      "ServiceUserName": "developer-at-123456789012",
      "CreateDate": "2019-10-01T04:31:41Z",
      "ServiceSpecificCredentialId": "ACCAQFODXMPL4YFHP7DZE",
      "ServiceName": "codecommit.amazonaws.com"
    },
    {

```



```
    "UserName": "developer",
    "Status": "Active",
    "ServiceUserName": "developer+1-at-123456789012",
    "CreateDate": "2019-10-01T04:31:45Z",
    "ServiceSpecificCredentialId": "ACCAQFOXMPL6VW57M7AJP",
    "ServiceName": "codecommit.amazonaws.com"
  }
]
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》CodeCommit中的[创建 HTTPS 连接的 Git 凭证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListServiceSpecificCredentials](#)中的。

## list-signing-certificates

以下代码示例演示了如何使用 list-signing-certificates。

### AWS CLI

列出 IAM 用户的签名证书

以下 list-signing-certificates 命令列出名为 Bob 的 IAM 用户的签名证书。

```
aws iam list-signing-certificates \
  --user-name Bob
```

输出：

```
{
  "Certificates": [
    {
      "UserName": "Bob",
      "Status": "Inactive",
      "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-body>-----
END CERTIFICATE-----",
      "CertificateId": "TA7SMP42TDN5Z260BPJE7EXAMPLE",
      "UploadDate": "2013-06-06T21:40:08Z"
    }
  ]
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的[管理签名证书](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSigningCertificates](#)中的。

## list-ssh-public-keys

以下代码示例演示了如何使用 list-ssh-public-keys。

### AWS CLI

列出附加到 IAM 用户的 SSH 公钥

以下list-ssh-public-keys示例列出了附加到 IAM 用户的 SSH 公钥sofia。

```
aws iam list-ssh-public-keys \
  --user-name sofia
```

输出：

```
{
  "SSHPublicKeys": [
    {
      "UserName": "sofia",
      "SSHPublicKeyId": "APKA1234567890EXAMPLE",
      "Status": "Inactive",
      "UploadDate": "2019-04-18T17:04:49+00:00"
    }
  ]
}
```

有关更多信息，请参阅 AWS IAM 用户指南 CodeCommit 中的 [SSH 密钥和 SSH 与](#)

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSshPublicKeys](#)中的。

## list-user-policies

以下代码示例演示了如何使用 list-user-policies。

### AWS CLI

要列出 IAM 用户的策略

以下 list-user-policies 命令将列出附加到名为 Bob 的 IAM 用户的策略。

```
aws iam list-user-policies \  
  --user-name Bob
```

输出：

```
{  
  "PolicyNames": [  
    "ExamplePolicy",  
    "TestPolicy"  
  ]  
}
```

有关更多信息，请参阅 [IAM 用户指南中的在您的 AWS 账户中创建AWS IAM 用户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUserPolicies](#)中的。

## list-user-tags

以下代码示例演示了如何使用 list-user-tags。

### AWS CLI

列出附加到用户的标签

以下 list-user-tags 命令检索与指定 IAM 用户关联的标签列表。

```
aws iam list-user-tags \  
  --user-name alice
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "Department",  
      "Value": "Accounting"  
    },  
    {  
      "Key": "DeptID",  
      "Value": "12345"  
    }  
  ]  
}
```

```
    ],  
    "IsTruncated": false  
  }  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListUserTags](#) 中的。

## list-users

以下代码示例演示了如何使用 `list-users`。

### AWS CLI

要列出 IAM 用户

以下 `list-users` 命令将列出当前账户中的 IAM 用户。

```
aws iam list-users
```

输出：

```
{  
  "Users": [  
    {  
      "UserName": "Adele",  
      "Path": "/",  
      "CreateDate": "2013-03-07T05:14:48Z",  
      "UserId": "AKIAI44QH8DHBEXAMPLE",  
      "Arn": "arn:aws:iam::123456789012:user/Adele"  
    },  
    {  
      "UserName": "Bob",  
      "Path": "/",  
      "CreateDate": "2012-09-21T23:03:13Z",  
      "UserId": "AKIAIOSFODNN7EXAMPLE",  
      "Arn": "arn:aws:iam::123456789012:user/Bob"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[列出 IAM 用户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUsers](#)中的。

## list-virtual-mfa-devices

以下代码示例演示了如何使用 list-virtual-mfa-devices。

### AWS CLI

列出虚拟 MFA 设备

以下 list-virtual-mfa-devices 命令列出已为当前账户配置的虚拟 MFA 设备。

```
aws iam list-virtual-mfa-devices
```

输出：

```
{
  "VirtualMFADevices": [
    {
      "SerialNumber": "arn:aws:iam::123456789012:mfa/ExampleMFADevice"
    },
    {
      "SerialNumber": "arn:aws:iam::123456789012:mfa/Fred"
    }
  ]
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[启用虚拟多重身份验证 \(MFA\) 设备](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListVirtualMfaDevices](#)中的。

## put-group-policy

以下代码示例演示了如何使用 put-group-policy。

### AWS CLI

要向组中添加策略

以下 put-group-policy 命令可将策略添加到名为 Admins 的 IAM 组。

```
aws iam put-group-policy \
```

```
--group-name Admins \  
--policy-document file://AdminPolicy.json \  
--policy-name AdminRoot
```

此命令不生成任何输出。

该策略在 AdminPolicy.json 文件中定义为一个 JSON 文档。（文件名和扩展名没有意义。）

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM policy](#)。

• 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutGroupPolicy](#) 中的。

## put-role-permissions-boundary

以下代码示例演示了如何使用 put-role-permissions-boundary。

### AWS CLI

示例 1：将基于自定义策略的权限边界应用于 IAM 角色

以下 put-role-permissions-boundary 示例应用名为 intern-boundary 的自定义策略作为指定 IAM 角色的权限边界。

```
aws iam put-role-permissions-boundary \  
  --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \  
  --role-name lambda-application-role
```

此命令不生成任何输出。

示例 2：将基于 AWS 托管策略的权限边界应用于 IAM 角色

以下 put-role-permissions-boundary 示例应用 AWS 托管 PowerUserAccess 策略作为指定 IAM 角色的权限边界。

```
aws iam put-role-permissions-boundary \  
  --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \  
  --role-name x-account-admin
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[修改角色](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutRolePermissionsBoundary](#)中的。

## put-role-policy

以下代码示例演示了如何使用 put-role-policy。

### AWS CLI

要将权限策略附加到 IAM 角色

以下 put-role-policy 命令可将权限策略附加到名为 Test-Role 的角色。

```
aws iam put-role-policy \  
  --role-name Test-Role \  
  --policy-name ExamplePolicy \  
  --policy-document file://AdminPolicy.json
```

此命令不生成任何输出。

该策略在 AdminPolicy.json 文件中定义为一个 JSON 文档。（文件名和扩展名没有意义。）

要将信任策略附加到角色，请使用 update-assume-role-policy 命令。

有关更多信息，请参阅《AWS IAM 用户指南》中的[修改角色](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutRolePolicy](#)中的。

## put-user-permissions-boundary

以下代码示例演示了如何使用 put-user-permissions-boundary。

### AWS CLI

示例 1：将基于自定义策略的权限边界应用于 IAM 用户

以下 put-user-permissions-boundary 示例应用名为 intern-boundary 的自定义策略作为指定 IAM 用户的权限边界。

```
aws iam put-user-permissions-boundary \  
  --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \  
  --user-name intern
```

此命令不生成任何输出。

示例 2：根据 AWS 托管策略向 IAM 用户应用权限边界

以下 `put-user-permissions-boundary` 示例应用名 `PowerUserAccess` 为指定 IAM 用户的权限边界的 AWS 托管策略。

```
aws iam put-user-permissions-boundary \  
  --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \  
  --user-name developer
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[添加和删除 IAM 身份权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutUserPermissionsBoundary](#) 中的。

## put-user-policy

以下代码示例演示了如何使用 `put-user-policy`。

### AWS CLI

要将策略附加到 IAM 用户

以下 `put-user-policy` 命令可将策略附加到名为 Bob 的 IAM 用户。

```
aws iam put-user-policy \  
  --user-name Bob \  
  --policy-name ExamplePolicy \  
  --policy-document file://AdminPolicy.json
```

此命令不生成任何输出。

该策略在 `AdminPolicy.json` 文件中定义为一个 JSON 文档。（文件名和扩展名没有意义。）

有关更多信息，请参阅《AWS IAM 用户指南》中的[添加和删除 IAM 身份权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutUserPolicy](#) 中的。

## remove-client-id-from-open-id-connect-provider

以下代码示例演示了如何使用 `remove-client-id-from-open-id-connect-provider`。



## AWS CLI

从为指定的 IAM OpenID Connect 提供者注册的客户端 ID 列表中删除指定的客户端 ID

此示例从与 ARN 为 `arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com` 的 IAM OIDC 提供者关联的客户端 ID 列表中删除客户端 ID `My-TestApp-3`。

```
aws iam remove-client-id-from-open-id-connect-provider
  --client-id My-TestApp-3 \
  --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的 [创建 OpenID Connect \( OIDC \) 身份提供者](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RemoveClientIdFromOpenIdConnectProvider](#) 中的。

## remove-role-from-instance-profile

以下代码示例演示了如何使用 `remove-role-from-instance-profile`。

### AWS CLI

从实例配置文件中删除角色

以下 `remove-role-from-instance-profile` 命令可将名为 `Test-Role` 的角色从名为 `ExampleInstanceProfile` 的实例配置文件中删除。

```
aws iam remove-role-from-instance-profile \
  --instance-profile-name ExampleInstanceProfile \
  --role-name Test-Role
```

有关更多信息，请参阅《AWS IAM 用户指南》中的 [使用实例配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RemoveRoleFromInstanceProfile](#) 中的。

## remove-user-from-group

以下代码示例演示了如何使用 `remove-user-from-group`。

## AWS CLI

要从 IAM 组中删除用户

以下 `remove-user-from-group` 命令可将名为 Bob 的用户从名为 Admins 的 IAM 组中删除。

```
aws iam remove-user-from-group \  
  --user-name Bob \  
  --group-name Admins
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[在 IAM 用户组中添加和删除用户](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RemoveUserFromGroup](#)中的。

## reset-service-specific-credential

以下代码示例演示了如何使用 `reset-service-specific-credential`。

### AWS CLI

示例 1：重置附加到提出请求的用户的特定服务凭证的密码

以下 `reset-service-specific-credential` 示例为附加到发出请求的用户的特定服务凭证生成一个新的加密强密码。

```
aws iam reset-service-specific-credential \  
  --service-specific-credential-id ACCAEXAMPLE123EXAMPLE
```

输出：

```
{  
  "ServiceSpecificCredential": {  
    "CreateDate": "2019-04-18T20:45:36+00:00",  
    "ServiceName": "codecommit.amazonaws.com",  
    "ServiceUserName": "sofia-at-123456789012",  
    "ServicePassword": "+oaFsNk7tLco+C/obP9Ghhc0zGcK0ayTmE3LnAmAmH4=",  
    "ServiceSpecificCredentialId": "ACCAEXAMPLE123EXAMPLE",  
    "UserName": "sofia",  
    "Status": "Active"  
  }  
}
```

```
}
```

## 示例 2：重置附加到指定用户的服务特定凭证的密码

以下 `reset-service-specific-credential` 示例为附加到指定用户的服务专用凭证生成新的加密强密码。

```
aws iam reset-service-specific-credential \  
  --user-name sofia \  
  --service-specific-credential-id ACCAEXAMPLE123EXAMPLE
```

输出：

```
{  
  "ServiceSpecificCredential": {  
    "CreateDate": "2019-04-18T20:45:36+00:00",  
    "ServiceName": "codecommit.amazonaws.com",  
    "ServiceUserName": "sofia-at-123456789012",  
    "ServicePassword": "+oaFsNk7tLco+C/obP9Ghhc0zGcK0ayTmE3LnAmAmH4=",  
    "ServiceSpecificCredentialId": "ACCAEXAMPLE123EXAMPLE",  
    "UserName": "sofia",  
    "Status": "Active"  
  }  
}
```

有关更多信息，请参阅《AWS CodeCommit 用户指南》CodeCommit 中的 [创建 HTTPS 连接的 Git 凭证](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ResetServiceSpecificCredential](#) 中的。

## resync-mfa-device

以下代码示例演示了如何使用 `resync-mfa-device`。

### AWS CLI

#### 重新同步 MFA 设备

以下 `resync-mfa-device` 示例将与 IAM 用户 Bob 关联且 ARN 为 `arn:aws:iam::123456789012:mfa/BobsMFADevice` 的 MFA 设备与提供这两个身份验证码的身份验证器程序同步。

```
aws iam resync-mfa-device \  
  --user-name Bob \  
  --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \  
  --authentication-code1 123456 \  
  --authentication-code2 987654
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[在 AWS 中使用多重身份验证 \(MFA\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ResyncMfaDevice](#)中的。

## set-default-policy-version

以下代码示例演示了如何使用 set-default-policy-version。

### AWS CLI

将指定策略的指定版本设置为策略的默认版本。

此示例将 ARN 为 arn:aws:iam::123456789012:policy/MyPolicy 的策略的 v2 版本设置为默认活动版本。

```
aws iam set-default-policy-version \  
  --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \  
  --version-id v2
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[IAM 中的策略和权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[SetDefaultPolicyVersion](#)中的。

## set-security-token-service-preferences

以下代码示例演示了如何使用 set-security-token-service-preferences。

### AWS CLI

设置全局终端节点令牌版本

以下 set-security-token-service-preferences 示例将 Amazon STS 配置为在您针对全局终端节点进行身份验证时使用版本 2 令牌。

```
aws iam set-security-token-service-preferences \  
  --global-endpoint-token-version v2Token
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IAM 用户指南中的在[AWS 区域中管理 AWS STS](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[SetSecurityTokenServicePreferences](#)中的。

## simulate-custom-policy

以下代码示例演示了如何使用 simulate-custom-policy。

### AWS CLI

示例 1：模拟与 IAM 用户或角色关联的所有 IAM 策略的影响

以下内容 simulate-custom-policy 展示了如何同时提供策略和定义变量值，以及如何模拟 API 调用以查看是允许还是被拒绝。以下示例显示了仅在指定日期和时间之后才允许访问数据库的策略。模拟之所以成功，是因为模拟的操作和指定的 aws:CurrentTime 变量都符合策略的要求。

```
aws iam simulate-custom-policy \  
  --policy-input-list '{"Version":"2012-10-17","Statement":  
{ "Effect":"Allow","Action":"dynamodb:*","Resource":"*","Condition":  
{ "DateGreaterThan":{"aws:CurrentTime":"2018-08-16T12:00:00Z"}}}' \  
  --action-names dynamodb:CreateBackup \  
  --context-entries  
  "ContextKeyName='aws:CurrentTime',ContextKeyValues='2019-04-25T11:00:00Z',ContextKeyType=da
```

输出：

```
{  
  "EvaluationResults": [  
    {  
      "EvalActionName": "dynamodb:CreateBackup",  
      "EvalResourceName": "*",  
      "EvalDecision": "allowed",  
      "MatchedStatements": [  
        {  
          "SourcePolicyId": "PolicyInputList.1",  
          "StartPosition": {
```

```

        "Line": 1,
        "Column": 38
      },
      "EndPosition": {
        "Line": 1,
        "Column": 167
      }
    }
  ],
  "MissingContextValues": []
}
]
}

```

## 示例 2：模拟策略禁止的命令

以下simulate-custom-policy示例显示了模拟策略禁止的命令的结果。在此示例中，提供的日期早于保单条件所要求的日期。

```

aws iam simulate-custom-policy \
  --policy-input-list '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"*","Condition":
{"DateGreaterThan":{"aws:CurrentTime":"2018-08-16T12:00:00Z"}}}' \
  --action-names dynamodb>CreateBackup \
  --context-entries
"ContextKeyName='aws:CurrentTime',ContextKeyValues='2014-04-25T11:00:00Z',ContextKeyType=da

```

输出：

```

{
  "EvaluationResults": [
    {
      "EvalActionName": "dynamodb>CreateBackup",
      "EvalResourceName": "*",
      "EvalDecision": "implicitDeny",
      "MatchedStatements": [],
      "MissingContextValues": []
    }
  ]
}

```

有关更多信息，请参阅 [IAM 用户指南中的使用 IAM 策略模拟器测试AWS IAM 策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SimulateCustomPolicy](#)中的。

## simulate-principal-policy

以下代码示例演示了如何使用 simulate-principal-policy。

### AWS CLI

#### 示例 1：模拟任意 IAM 策略的影响

以下内容simulate-principal-policy显示如何模拟用户调用 API 操作并确定与该用户关联的策略是允许还是拒绝该操作。在以下示例中，用户的策略仅允许该codecommit:ListRepositories操作。

```
aws iam simulate-principal-policy \  
  --policy-source-arn arn:aws:iam::123456789012:user/alejandro \  
  --action-names codecommit:ListRepositories
```

输出：

```
{  
  "EvaluationResults": [  
    {  
      "EvalActionName": "codecommit:ListRepositories",  
      "EvalResourceName": "*",  
      "EvalDecision": "allowed",  
      "MatchedStatements": [  
        {  
          "SourcePolicyId": "Grant-Access-To-CodeCommit-ListRepo",  
          "StartPosition": {  
            "Line": 3,  
            "Column": 19  
          },  
          "EndPosition": {  
            "Line": 9,  
            "Column": 10  
          }  
        }  
      ],  
      "MissingContextValues": []  
    }  
  ]  
}
```

```
}
```

## 示例 2：模拟禁止命令的效果

以下 `simulate-custom-policy` 示例显示了模拟用户策略所禁止的命令的结果。在以下示例中，用户的策略仅允许在特定日期和时间之后访问 DynamoDB 数据库。在模拟中，用户尝试使用早于策略条件允许的 `aws:CurrentTime` 值访问数据库。

```
aws iam simulate-principal-policy \  
  --policy-source-arn arn:aws:iam::123456789012:user/alejandro \  
  --action-names dynamodb:CreateBackup \  
  --context-entries  
  "ContextKeyName='aws:CurrentTime',ContextKeyValues='2018-04-25T11:00:00Z',ContextKeyType=da
```

输出：

```
{  
  "EvaluationResults": [  
    {  
      "EvalActionName": "dynamodb:CreateBackup",  
      "EvalResourceName": "*",  
      "EvalDecision": "implicitDeny",  
      "MatchedStatements": [],  
      "MissingContextValues": []  
    }  
  ]  
}
```

有关更多信息，请参阅 [IAM 用户指南中的使用 IAM 策略模拟器测试 AWS IAM 策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SimulatePrincipalPolicy](#) 中的。

## tag-instance-profile

以下代码示例演示了如何使用 `tag-instance-profile`。

### AWS CLI

向实例配置文件添加标签

以下 `tag-instance-profile` 命令将带有部门名称的标签添加到指定的实例配置文件中。



```
aws iam tag-instance-profile \  
  --instance-profile-name deployment-role \  
  --tags '[{"Key": "Department", "Value": "Accounting"}]'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagInstanceProfile](#)中的。

## tag-mfa-device

以下代码示例演示了如何使用 tag-mfa-device。

### AWS CLI

向 MFA 设备添加标签

以下tag-mfa-device命令将带有部门名称的标签添加到指定的 MFA 设备。

```
aws iam tag-mfa-device \  
  --serial-number arn:aws:iam::123456789012:mfa/alice \  
  --tags '[{"Key": "Department", "Value": "Accounting"}]'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagMfaDevice](#)中的。

## tag-open-id-connect-provider

以下代码示例演示了如何使用 tag-open-id-connect-provider。

### AWS CLI

向兼容 OpenID Connect (OIDC) 的身份提供商添加标签

以下tag-open-id-connect-provider命令将带有部门名称的标签添加到指定的 OIDC 身份提供商。

```
aws iam tag-open-id-connect-provider \  
  --serial-number arn:aws:iam::123456789012:mfa/alice \  
  --tags '[{"Key": "Department", "Value": "Accounting"}]'
```

```
--open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
server.example.com \
--tags '[{"Key": "Department", "Value": "Accounting"}]'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagOpenIdConnectProvider](#)中的。

## tag-policy

以下代码示例演示了如何使用 tag-policy。

### AWS CLI

向客户托管策略添加标签

以下tag-policy命令将带有部门名称的标签添加到指定的客户托管策略中。

```
aws iam tag-policy \
--policy-arn arn:aws:iam::123456789012:policy/billing-access \
--tags '[{"Key": "Department", "Value": "Accounting"}]'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagPolicy](#)中的。

## tag-role

以下代码示例演示了如何使用 tag-role。

### AWS CLI

为角色添加标签

以下 tag-role 命令为指定角色添加带有部门名称的标签。

```
aws iam tag-role --role-name my-role \
--tags '{"Key": "Department", "Value": "Accounting"}'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagRole](#)中的。

## tag-saml-provider

以下代码示例演示了如何使用 tag-saml-provider。

### AWS CLI

向 SAML 提供商添加标签

以下tag-saml-provider命令将带有部门名称的标签添加到指定的 SAML 提供商。

```
aws iam tag-saml-provider \  
  --saml-provider-arn arn:aws:iam::123456789012:saml-provider/ADFS \  
  --tags '[{"Key": "Department", "Value": "Accounting"}]'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagSamlProvider](#)中的。

## tag-server-certificate

以下代码示例演示了如何使用 tag-server-certificate。

### AWS CLI

向服务器证书添加标签

以下tag-saml-provider命令将带有部门名称的标签添加到指定的服务器证书。

```
aws iam tag-server-certificate \  
  --server-certificate-name ExampleCertificate \  
  --tags '[{"Key": "Department", "Value": "Accounting"}]'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagServerCertificate](#)中的。

## tag-user

以下代码示例演示了如何使用 tag-user。

### AWS CLI

为用户添加标签

以下 tag-user 命令为指定用户添加与部门关联的标签。

```
aws iam tag-user \  
  --user-name alice \  
  --tags '{"Key": "Department", "Value": "Accounting"}'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagUser](#)中的。

## untag-instance-profile

以下代码示例演示了如何使用 untag-instance-profile。

### AWS CLI

从实例配置文件中删除标签

以下 untag-instance-profile 命令从指定的实例配置文件中删除密钥名称为“Department”的所有标签。

```
aws iam untag-instance-profile \  
  --instance-profile-name deployment-role \  
  --tag-keys Department
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagInstanceProfile](#)中的。

## untag-mfa-device

以下代码示例演示了如何使用 `untag-mfa-device`。

### AWS CLI

从 MFA 设备中移除标签

以下`untag-mfa-device`命令从指定的 MFA 设备中删除所有密钥名称为“部门”的标签。

```
aws iam untag-mfa-device \  
  --serial-number arn:aws:iam::123456789012:mfa/alice \  
  --tag-keys Department
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagMfaDevice](#)中的。

## untag-open-id-connect-provider

以下代码示例演示了如何使用 `untag-open-id-connect-provider`。

### AWS CLI

从 OIDC 身份提供商处移除标签

以下`untag-open-id-connect-provider`命令从指定的 OIDC 身份提供商中删除密钥名称为“部门”的所有标签。

```
aws iam untag-open-id-connect-provider \  
  --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/  
server.example.com \  
  --tag-keys Department
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagOpenIdConnectProvider](#)中的。

## untag-policy

以下代码示例演示了如何使用 `untag-policy`。

### AWS CLI

从客户托管策略中移除标签

以下 `untag-policy` 命令从指定的客户托管策略中删除密钥名称为 “Department” 的所有标签。

```
aws iam untag-policy \  
  --policy-arn arn:aws:iam::452925170507:policy/billing-access \  
  --tag-keys Department
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagPolicy](#) 中的。

## untag-role

以下代码示例演示了如何使用 `untag-role`。

### AWS CLI

删除角色的标签

以下 `untag-role` 命令从指定角色中删除键名称为“部门”的所有标签。

```
aws iam untag-role \  
  --role-name my-role \  
  --tag-keys Department
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagRole](#) 中的。

## untag-saml-provider

以下代码示例演示了如何使用 `untag-saml-provider`。

## AWS CLI

### 从 SAML 提供商中移除标签

以下`untag-saml-provider`命令从指定的实例配置文件中删除密钥名称为“Department”的所有标签。

```
aws iam untag-saml-provider \  
  --saml-provider-arn arn:aws:iam::123456789012:saml-provider/ADFS \  
  --tag-keys Department
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagSamlProvider](#)中的。

## untag-server-certificate

以下代码示例演示了如何使用 `untag-server-certificate`。

## AWS CLI

### 从服务器证书中删除标签

以下`untag-server-certificate`命令从指定的服务器证书中删除所有密钥名为“部门”的标签。

```
aws iam untag-server-certificate \  
  --server-certificate-name ExampleCertificate \  
  --tag-keys Department
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagServerCertificate](#)中的。

## untag-user

以下代码示例演示了如何使用 `untag-user`。

## AWS CLI

### 删除用户的标签

以下 `untag-user` 命令从指定用户中删除键名称为“部门”的所有标签。

```
aws iam untag-user \  
  --user-name alice \  
  --tag-keys Department
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[标记 IAM 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagUser](#)中的。

## update-access-key

以下代码示例演示了如何使用 `update-access-key`。

## AWS CLI

### 要激活或停用 IAM 用户的访问密钥

以下 `update-access-key` 命令将停用名为 Bob IAM 用户的指定访问密钥（访问密钥 ID 和秘密访问密钥）。

```
aws iam update-access-key \  
  --access-key-id AKIAIOSFODNN7EXAMPLE \  
  --status Inactive \  
  --user-name Bob
```

此命令不生成任何输出。

停用密钥意味着它不能用于以编程方式访问。AWS 但密钥仍然可用，可以重新激活。

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM 用户的访问密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAccessKey](#)中的。

## update-account-password-policy

以下代码示例演示了如何使用 `update-account-password-policy`。



## AWS CLI

### 设置或更改当前账户密码策略

以下 `update-account-password-policy` 命令将密码策略设置为要求长度最少为八个字符，并要求在密码中包含一个或多个数字。

```
aws iam update-account-password-policy \  
  --minimum-password-length 8 \  
  --require-numbers
```

此命令不生成任何输出。

对账户密码策略的更改会影响为账户中的 IAM 用户创建的所有新密码。密码策略更改不会影响现有的密码。

有关更多信息，请参阅《AWS IAM 用户指南》中的[为 IAM 用户设置账户密码策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAccountPasswordPolicy](#)中的。

## update-assume-role-policy

以下代码示例演示了如何使用 `update-assume-role-policy`。

## AWS CLI

### 更新 IAM 角色的信任策略

以下 `update-assume-role-policy` 命令更新名为 `Test-Role` 的角色的信任策略。

```
aws iam update-assume-role-policy \  
  --role-name Test-Role \  
  --policy-document file://Test-Role-Trust-Policy.json
```

此命令不生成任何输出。

信任策略在 `Test-Role-Trust-Policy.json` 文件中定义为 JSON 文档。（文件名和扩展名没有意义。）信任策略必须指定主体。

要更新角色的权限策略，请使用 `put-role-policy` 命令。

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 IAM 角色](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAssumeRolePolicy](#)中的。

## update-group

以下代码示例演示了如何使用 update-group。

### AWS CLI

#### 重命名 IAM 组

以下 update-group 命令可将 IAM 组 Test 的名称更改为 Test-1。

```
aws iam update-group \  
  --group-name Test \  
  --new-group-name Test-1
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[重命名 IAM 用户组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateGroup](#) 中的。

## update-login-profile

以下代码示例演示了如何使用 update-login-profile。

### AWS CLI

#### 更新 IAM 用户的密码

以下 update-login-profile 命令为名为 Bob 的 IAM 用户创建新密码。

```
aws iam update-login-profile \  
  --user-name Bob \  
  --password <password>
```

此命令不生成任何输出。

要为账户设置密码策略，请使用 update-account-password-policy 命令。如果新密码违反了账户密码策略，则该命令将返回 PasswordPolicyViolation 错误。

如果账户密码策略允许，则 IAM 用户可以使用 change-password 命令更改自己的密码。

将密码保存在安全位置。如果密码丢失，则将无法恢复，必须使用 create-login-profile 命令创建新密码。

有关更多信息，请参阅《AWS IAM 用户指南》中的[管理 IAM 用户的密码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLoginProfile](#)中的。

## update-open-id-connect-provider-thumbprint

以下代码示例演示了如何使用 update-open-id-connect-provider-thumbprint。

### AWS CLI

将现有服务器证书指纹列表替换为新列表

此示例更新了其 ARN 为 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com 的 OIDC 提供者的证书指纹列表以使用新指纹。

```
aws iam update-open-id-connect-provider-thumbprint \  
  --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/  
example.oidcprovider.com \  
  --thumbprint-list 7359755EXAMPLEEabc3060bce3EXAMPLEec4542a3
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 OpenID Connect \( OIDC \) 身份提供者](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateOpenIdConnectProviderThumbprint](#)中的。

## update-role-description

以下代码示例演示了如何使用 update-role-description。

### AWS CLI

更改 IAM 角色的描述

以下 update-role 命令可将 IAM 角色 production-role 的描述更改为 Main production role。

```
aws iam update-role-description \  
  --role-name production-role \  
  --description 'Main production role'
```

输出：

```
{
  "Role": {
    "Path": "/",
    "RoleName": "production-role",
    "RoleId": "ARO0A1234567890EXAMPLE",
    "Arn": "arn:aws:iam::123456789012:role/production-role",
    "CreateDate": "2017-12-06T17:16:37+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::123456789012:root"
          },
          "Action": "sts:AssumeRole",
          "Condition": {}
        }
      ]
    },
    "Description": "Main production role"
  }
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[修改角色](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRoleDescription](#)中的。

## update-role

以下代码示例演示了如何使用 update-role。

### AWS CLI

更改 IAM 角色的描述或会话持续时间

以下 update-role 命令将 IAM 角色 production-role 的描述更改为 Main production role，并将最长会话持续时间设置为 12 小时。

```
aws iam update-role \
  --role-name production-role \
```

```
--description 'Main production role' \  
--max-session-duration 43200
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[修改角色](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRole](#)中的。

## update-saml-provider

以下代码示例演示了如何使用 update-saml-provider。

### AWS CLI

更新现有 SAML 提供者的元数据文档

此示例使用文件 SAMLMetaData.xml 中的新 SAML 元数据文档更新 ARN 为 arn:aws:iam::123456789012:saml-provider/SAMLADFS 的 IAM 中的 SAML 提供者。

```
aws iam update-saml-provider \  
  --saml-metadata-document file://SAMLMetaData.xml \  
  --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

输出：

```
{  
  "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/SAMLADFS"  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[创建 IAM SAML 身份提供者](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateSamlProvider](#)中的。

## update-server-certificate

以下代码示例演示了如何使用 update-server-certificate。

### AWS CLI

更改您 AWS 账户中服务器证书的路径或名称

以下 `update-server-certificate` 命令可将证书名称从 `myServerCertificate` 更改为 `myUpdatedServerCertificate`。它还会将路径更改为 `/cloudfront/`，以便 Amazon CloudFront 服务可以对其进行访问。此命令不生成任何输出。运行 `list-server-certificates` 命令即可查看更新结果。

```
aws iam update-server-certificate \
  --server-certificate-name myServerCertificate \
  --new-server-certificate-name myUpdatedServerCertificate \
  --new-path /cloudfront/
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的[在 IAM 中管理服务器证书](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateServerCertificate](#)中的。

## update-service-specific-credential

以下代码示例演示了如何使用 `update-service-specific-credential`。

### AWS CLI

示例 1：更新请求用户的服务专用证书的状态

以下 `update-service-specific-credential` 示例更改了向其发出请求的用户的指定凭据的状态。Inactive

```
aws iam update-service-specific-credential \
  --service-specific-credential-id ACCAEXAMPLE123EXAMPLE \
  --status Inactive
```

此命令不生成任何输出。

示例 2：更新指定用户的服务专用凭证的状态

以下 `update-service-specific-credential` 示例将指定用户的凭据状态更改为“非活动”。

```
aws iam update-service-specific-credential \
  --user-name sofia \
  --service-specific-credential-id ACCAEXAMPLE123EXAMPLE \
```

```
--status Inactive
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CodeCommit 用户指南》CodeCommit 中的 [“创建 HTTPS 连接的 Git 凭证”](#)

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateServiceSpecificCredential](#) 中的。

## update-signing-certificate

以下代码示例演示了如何使用 update-signing-certificate。

### AWS CLI

激活或停用 IAM 用户的签名证书

以下 update-signing-certificate 命令停用名为 Bob 的 IAM 用户的指定签名证书。

```
aws iam update-signing-certificate \  
  --certificate-id TA7SMP42TDN5Z260BPJE7EXAMPLE \  
  --status Inactive \  
  --user-name Bob
```

要获取签名证书的 ID，请使用 list-signing-certificates 命令。

有关更多信息，请参阅《Amazon EC2 用户指南》中的 [管理签名证书](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateSigningCertificate](#) 中的。

## update-ssh-public-key

以下代码示例演示了如何使用 update-ssh-public-key。

### AWS CLI

更改 SSH 公钥的状态

以下 update-ssh-public-key 命令将指定公钥的状态更改为 Inactive。

```
aws iam update-ssh-public-key \  
  --public-key-id TA7SMP42TDN5Z260BPJE7EXAMPLE \  
  --status Inactive
```

```
--user-name sofia \  
--ssh-public-key-id APKA1234567890EXAMPLE \  
--status Inactive
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IAM 用户指南 CodeCommit 中的 [使用 SSH 密钥和 SSH](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateSshPublicKey](#) 中的。

## update-user

以下代码示例演示了如何使用 update-user。

### AWS CLI

要更改 IAM 用户的名称

以下 update-user 命令可将 IAM 用户 Bob 的名称更改为 Robert。

```
aws iam update-user \  
  --user-name Bob \  
  --new-user-name Robert
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IAM 用户指南》中的 [重命名 IAM 用户组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateUser](#) 中的。

## upload-server-certificate

以下代码示例演示了如何使用 upload-server-certificate。

### AWS CLI

将服务器证书上传到您的 AWS 账户

以下 upload-server-certificate 命令将服务器证书上传到您的 AWS 账户。在此示例中，证书位于 public\_key\_cert\_file.pem 文件中，关联的私有密钥位于 my\_private\_key.pem 文件中，而证书颁发机构 (CA) 提供的证书链位于 my\_certificate\_chain\_file.pem 文件中。文件上



传完后，该文件将以该名称显示myServerCertificate。以 file:// 开头的参数让命令读取文件的内容，将其用作参数值，而不是文件名本身。

```
aws iam upload-server-certificate \  
  --server-certificate-name myServerCertificate \  
  --certificate-body file://public_key_cert_file.pem \  
  --private-key file://my_private_key.pem \  
  --certificate-chain file://my_certificate_chain_file.pem
```

输出：

```
{  
  "ServerCertificateMetadata": {  
    "Path": "/",  
    "ServerCertificateName": "myServerCertificate",  
    "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",  
    "Arn": "arn:aws:iam::1234567989012:server-certificate/myServerCertificate",  
    "UploadDate": "2019-04-22T21:13:44+00:00",  
    "Expiration": "2019-10-15T22:23:16+00:00"  
  }  
}
```

有关更多信息，请参阅《使用 IAM》中的创建、上传和删除服务器证书。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UploadServerCertificate](#)中的。

## upload-signing-certificate

以下代码示例演示了如何使用 upload-signing-certificate。

### AWS CLI

上传 IAM 用户的签名证书

以下 upload-signing-certificate 命令上传名为 Bob 的 IAM 用户的签名证书。

```
aws iam upload-signing-certificate \  
  --user-name Bob \  
  --certificate-body file://certificate.pem
```

输出：

```
{
  "Certificate": {
    "UserName": "Bob",
    "Status": "Active",
    "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-body>-----END
CERTIFICATE-----",
    "CertificateId": "TA7SMP42TDN5Z260BPJE7EXAMPLE",
    "UploadDate": "2013-06-06T21:40:08.121Z"
  }
}
```

证书位于名为 `certificate.pem` 的文件中，采用 PEM 格式。

有关更多信息，请参阅《使用 IAM》指南中的“创建和上传用户签名证书”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UploadSigningCertificate](#) 中的。

## upload-ssh-public-key

以下代码示例演示了如何使用 `upload-ssh-public-key`。

### AWS CLI

上传 SSH 公钥并将其与用户关联

以下 `upload-ssh-public-key` 命令上传在文件中找到的公钥 `sshkey.pub` 并将其附加到用户 `sofia`。

```
aws iam upload-ssh-public-key \
  --user-name sofia \
  --ssh-public-key-body file://sshkey.pub
```

输出：

```
{
  "SSHPublicKey": {
    "UserName": "sofia",
    "SSHPublicKeyId": "APKA1234567890EXAMPLE",
    "Fingerprint": "12:34:56:78:90:ab:cd:ef:12:34:56:78:90:ab:cd:ef",
    "SSHPublicKeyBody": "ssh-rsa <<long string generated by ssh-keygen
command>>",
```

```
    "Status": "Active",  
    "UploadDate": "2019-04-18T17:04:49+00:00"  
  }  
}
```

有关如何以适合此命令的格式生成密钥的更多信息，请参阅 [SSH 和 Linux、macOS 或 Unix : 为 Git 和/ CodeCommit 或 SSH 和 Windows 设置公钥和私钥 : 为 Git 设置公钥和私钥，以及 CodeCommit](#) 《AWS CodeCommit 用户指南》。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UploadSshPublicKey](#) 中的。

## 使用 IAM 访问分析器示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 IAM Access Analyzer 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **apply-archive-rule**

以下代码示例演示了如何使用 `apply-archive-rule`。

#### AWS CLI

将存档规则应用于符合存档规则标准的现有查找结果

以下 `apply-archive-rule` 示例将存档规则应用于符合存档规则条件的现有查找结果。

```
aws accessanalyzer apply-archive-rule \  

```

```
--analyzer-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/
UnusedAccess-ConsoleAnalyzer-organization \
--rule-name MyArchiveRule
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IAM 用户指南中的[存档规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ApplyArchiveRule](#)中的。

## cancel-policy-generation

以下代码示例演示了如何使用 cancel-policy-generation。

### AWS CLI

取消所请求的策略生成

以下cancel-policy-generation示例取消了请求的策略生成任务 ID。

```
aws accessanalyzer cancel-policy-generation \
--job-id 923a56b0-ebb8-4e80-8a3c-a11ccfbcd6f2
```

此命令不生成任何输出。

有关更多信息，请参阅 [IAM 用户指南中的 I AWS AM 访问分析器策略生成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CancelPolicyGeneration](#)中的。

## check-access-not-granted

以下代码示例演示了如何使用 check-access-not-granted。

### AWS CLI

检查策略是否不允许指定访问权限

以下check-access-not-granted示例检查策略是否不允许指定访问。

```
aws accessanalyzer check-access-not-granted \
--policy-document file://myfile.json \
```

```
--access actions="s3:DeleteBucket","s3:GetBucketLocation" \  
--policy-type IDENTITY_POLICY
```

myfile.json 的内容：

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetObject",  
        "s3:ListBucket"  
      ],  
      "Resource": [  
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET",  
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"  
      ]  
    }  
  ]  
}
```

输出：

```
{  
  "result": "PASS",  
  "message": "The policy document does not grant access to perform the listed  
actions."  
}
```

有关更多信息，请参阅 IAM 用户指南中的[使用 IAM Access Analyzer API 预览访问权限](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CheckAccessNotGranted](#)中的。

## check-no-new-access

以下代码示例演示了如何使用 check-no-new-access。

### AWS CLI

检查与现有策略相比，更新后的策略是否允许新的访问权限

以下check-no-new-access示例检查与现有策略相比，更新后的策略是否允许新的访问权限。

```
aws accessanalyzer check-no-new-access \  
  --existing-policy-document file://existing-policy.json \  
  --new-policy-document file://new-policy.json \  
  --policy-type IDENTITY_POLICY
```

existing-policy.json 的内容：

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetObject",  
        "s3:ListBucket"  
      ],  
      "Resource": [  
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET",  
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"  
      ]  
    }  
  ]  
}
```

new-policy.json 的内容：

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetObject",  
        "s3:GetObjectAcl",  
        "s3:ListBucket"  
      ],  
      "Resource": [  
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET",  
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"  
      ]  
    }  
  ]  
}
```

```
    }  
  ]  
}
```

输出：

```
{  
  "result": "FAIL",  
  "message": "The modified permissions grant new access compared to your existing  
policy.",  
  "reasons": [  
    {  
      "description": "New access in the statement with index: 0.",  
      "statementIndex": 0  
    }  
  ]  
}
```

有关更多信息，请参阅 IAM 用户指南中的[使用 IAM Access Analyzer API 预览访问权限](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CheckNoNewAccess](#)中的。

## create-access-preview

以下代码示例演示了如何使用 create-access-preview。

### AWS CLI

创建访问预览，允许您在部署资源权限之前预览资源的 IAM Access Analyzer 调查结果

以下 create-access-preview 示例创建了一个访问预览，允许您在 AWS 账户中部署资源权限之前预览资源的 IAM Access Analyzer 结果。

```
aws accessanalyzer create-access-preview \  
  --analyzer-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/  
ConsoleAnalyzer-account \  
  --configurations file://myfile.json
```

myfile.json 的内容：

```
{
```

```

"arn:aws:s3:::DOC-EXAMPLE-BUCKET": {
  "s3Bucket": {
    "bucketPolicy": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",\"Principal\":{\"AWS\":[\"arn:aws:iam::111122223333:root\"]},\"Action\":[\"s3:PutObject\",\"s3:PutObjectAcl\"],\"Resource\":[\"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*\"]}]}",
    "bucketPublicAccessBlock": {
      "ignorePublicAcls": true,
      "restrictPublicBuckets": true
    },
    "bucketAclGrants": [
      {
        "grantee": {
          "id":
"79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be"
        },
        "permission": "READ"
      }
    ]
  }
}

```

输出：

```

{
  "id": "3c65eb13-6ef9-4629-8919-a32043619e6b"
}

```

有关更多信息，请参阅 IAM 用户指南中的[使用 IAM Access Analyzer API 预览访问权限](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateAccessPreview](#)中的。

## create-analyzer

以下代码示例演示了如何使用 create-analyzer。

### AWS CLI

#### 创建分析器

以下 create-analyzer 示例在您的 AWS 账户中创建了一个分析器。



```
aws accessanalyzer create-analyzer \  
  --analyzer-name example \  
  --type ACCOUNT
```

输出：

```
{  
  "arn": "arn:aws:access-analyzer:us-east-2:111122223333:analyzer/example"  
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的 [Identify and Access Management Access Analyzer 发现的入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateAnalyzer](#) 中的。

## create-archive-rule

以下代码示例演示了如何使用 create-archive-rule。

### AWS CLI

为指定的分析器创建存档规则

以下 create-archive-rule 示例为您 AWS 账户中的指定分析器创建存档规则。

```
aws accessanalyzer create-archive-rule \  
  --analyzer-name UnusedAccess-ConsoleAnalyzer-organization \  
  --rule-name MyRule \  
  --filter '{"resource": {"contains": ["Cognito"]}, "resourceType": {"eq":  
  ["AWS::IAM::Role"]}]'
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IAM 用户指南中的 [存档规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateArchiveRule](#) 中的。

## delete-analyzer

以下代码示例演示了如何使用 delete-analyzer。

## AWS CLI

### 删除指定的分析器

以下delete-analyzer示例删除了您 AWS 账户中的指定分析器。

```
aws accessanalyzer delete-analyzer \  
  --analyzer-name example
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IAM 用户指南中的[存档规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAnalyzer](#)中的。

## delete-archive-rule

以下代码示例演示了如何使用 delete-archive-rule。

## AWS CLI

### 删除指定的存档规则

以下delete-archive-rule示例删除了您 AWS 账户中指定的存档规则。

```
aws accessanalyzer delete-archive-rule \  
  --analyzer-name UnusedAccess-ConsoleAnalyzer-organization \  
  --rule-name MyRule
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IAM 用户指南中的[存档规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteArchiveRule](#)中的。

## get-access-preview

以下代码示例演示了如何使用 get-access-preview。

## AWS CLI

检索有关指定分析器的访问预览的信息

以下 `get-access-preview` 示例检索有关您 AWS 账户中指定分析器的访问预览的信息。

```
aws accessanalyzer get-access-preview \
  --access-preview-id 3c65eb13-6ef9-4629-8919-a32043619e6b \
  --analyzer-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/
ConsoleAnalyzer-account
```

输出：

```
{
  "accessPreview": {
    "id": "3c65eb13-6ef9-4629-8919-a32043619e6b",
    "analyzerArn": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/
ConsoleAnalyzer-account",
    "configurations": {
      "arn:aws:s3::DOC-EXAMPLE-BUCKET": {
        "s3Bucket": {
          "bucketPolicy": "{\"Version\":\"2012-10-17\",\"Statement\":
[[{\"Effect\":\"Allow\",\"Principal\":{\"AWS\":[\"arn:aws:iam::111122223333:root\"]},
\"Action\":[\"s3:PutObject\",\"s3:PutObjectAcl\"],\"Resource\":[\"arn:aws:s3::DOC-
EXAMPLE-BUCKET/*\"]}]]",
          "bucketAclGrants": [
            {
              "permission": "READ",
              "grantee": {
                "id":
"79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be"
              }
            }
          ],
          "bucketPublicAccessBlock": {
            "ignorePublicAcls": true,
            "restrictPublicBuckets": true
          }
        }
      }
    },
    "createdAt": "2024-02-17T00:18:44+00:00",
    "status": "COMPLETED"
  }
}
```

有关更多信息，请参阅 IAM 用户指南中的 [使用 IAM Access Analyzer API 预览访问权限](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAccessPreview](#)中的。

## get-analyzed-resource

以下代码示例演示了如何使用 get-analyzed-resource。

### AWS CLI

检索有关已分析资源的信息

以下get-analyzed-resource示例检索有关在您的 AWS 账户中分析过的资源的信息。

```
aws accessanalyzer get-analyzed-resource \
  --analyzer-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/
  ConsoleAnalyzer-account \
  --resource-arn arn:aws:s3:::DOC-EXAMPLE-BUCKET
```

输出：

```
{
  "resource": {
    "analyzedAt": "2024-02-15T18:01:53.002000+00:00",
    "isPublic": false,
    "resourceArn": "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
    "resourceOwnerAccount": "111122223333",
    "resourceType": "AWS::S3::Bucket"
  }
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[使用 I AWS identity and Access Management 访问分析器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAnalyzedResource](#)中的。

## get-analyzer

以下代码示例演示了如何使用 get-analyzer。

### AWS CLI

检索有关指定分析器的信息

以下`get-analyzer`示例检索有关您 AWS 账户中指定分析器的信息。

```
aws accessanalyzer get-analyzer \  
  --analyzer-name ConsoleAnalyzer-account
```

输出：

```
{  
  "analyzer": {  
    "arn": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/  
ConsoleAnalyzer-account",  
    "createdAt": "2019-12-03T07:28:17+00:00",  
    "lastResourceAnalyzed": "arn:aws:sns:us-west-2:111122223333:config-topic",  
    "lastResourceAnalyzedAt": "2024-02-15T18:01:53.003000+00:00",  
    "name": "ConsoleAnalyzer-account",  
    "status": "ACTIVE",  
    "tags": {  
      "auto-delete": "no"  
    },  
    "type": "ACCOUNT"  
  }  
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[使用 I AWS identity and Access Management 访问分析器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetAnalyzer](#)中的。

## get-archive-rule

以下代码示例演示了如何使用 `get-archive-rule`。

### AWS CLI

检索有关存档规则的信息

以下`get-archive-rule`示例检索有关您 AWS 账户中存档规则的信息。

```
aws accessanalyzer get-archive-rule \  
  --analyzer-name UnusedAccess-ConsoleAnalyzer-organization \  
  --rule-name MyArchiveRule
```

输出：

```
{
  "archiveRule": {
    "createdAt": "2024-02-15T00:49:27+00:00",
    "filter": {
      "resource": {
        "contains": [
          "Cognito"
        ]
      },
      "resourceType": {
        "eq": [
          "AWS::IAM::Role"
        ]
      }
    },
    "ruleName": "MyArchiveRule",
    "updatedAt": "2024-02-15T00:49:27+00:00"
  }
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[存档规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetArchiveRule](#)中的。

## get-finding-v2

以下代码示例演示了如何使用 get-finding-v2。

### AWS CLI

检索有关指定结果的信息

以下 get-finding-v2 示例检索有关您 AWS 账户中指定结果的信息。

```
aws accessanalyzer get-finding-v2 \
  --analyzer-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/
  ConsoleAnalyzer-organization \
  --id 0910eedb-381e-4e95-adda-0d25c19e6e90
```

输出：

```
{
  "findingDetails": [
    {
      "externalAccessDetails": {
        "action": [
          "sts:AssumeRoleWithWebIdentity"
        ],
        "condition": {
          "cognito-identity.amazonaws.com:aud": "us-west-2:EXAMPLE0-0000-0000-0000-000000000000"
        },
        "isPublic": false,
        "principal": {
          "Federated": "cognito-identity.amazonaws.com"
        }
      }
    }
  ],
  "resource": "arn:aws:iam::111122223333:role/Cognito_testpoolAuth_Role",
  "status": "ACTIVE",
  "error": null,
  "createdAt": "2021-02-26T21:17:50.905000+00:00",
  "resourceType": "AWS::IAM::Role",
  "findingType": "ExternalAccess",
  "resourceOwnerAccount": "111122223333",
  "analyzedAt": "2024-02-16T18:17:47.888000+00:00",
  "id": "0910eedb-381e-4e95-adda-0d25c19e6e90",
  "updatedAt": "2021-02-26T21:17:50.905000+00:00"
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[查看调查结果](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的[GetFindingV2](#)。

## get-finding

以下代码示例演示了如何使用 get-finding。

### AWS CLI

检索有关指定结果的信息

以下get-finding示例检索有关您 AWS 账户中指定结果的信息。

```
aws accessanalyzer get-finding \  
  --analyzer-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/  
ConsoleAnalyzer-organization \  
  --id 0910eedb-381e-4e95-adda-0d25c19e6e90
```

输出：

```
{  
  "finding": {  
    "id": "0910eedb-381e-4e95-adda-0d25c19e6e90",  
    "principal": {  
      "Federated": "cognito-identity.amazonaws.com"  
    },  
    "action": [  
      "sts:AssumeRoleWithWebIdentity"  
    ],  
    "resource": "arn:aws:iam::111122223333:role/Cognito_testpoolAuth_Role",  
    "isPublic": false,  
    "resourceType": "AWS::IAM::Role",  
    "condition": {  
      "cognito-identity.amazonaws.com:aud": "us-  
west-2:EXAMPLE0-0000-0000-0000-000000000000"  
    },  
    "createdAt": "2021-02-26T21:17:50.905000+00:00",  
    "analyzedAt": "2024-02-16T18:17:47.888000+00:00",  
    "updatedAt": "2021-02-26T21:17:50.905000+00:00",  
    "status": "ACTIVE",  
    "resourceOwnerAccount": "111122223333"  
  }  
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[查看调查结果](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetFinding](#)中的。

## get-generated-policy

以下代码示例演示了如何使用 get-generated-policy。

### AWS CLI

检索使用 `StartPolicyGeneration` API 生成的策略



以下`get-generated-policy`示例检索使用您 AWS 账户中的 `StartPolicyGeneration` API 生成的策略。

```
aws accessanalyzer get-generated-policy \  
  --job-id c557dc4a-0338-4489-95dd-739014860ff9
```

输出：

```
{  
  "generatedPolicyResult": {  
    "generatedPolicies": [  
      {  
        "policy": "{\"Version\":\"2012-10-17\",\"Statement\":  
[{\n\"Sid\":\n\"SupportedServiceSid0\", \"Effect\":\n\"Allow\", \"Action\":  
[\n\"access-analyzer:GetAnalyzer\", \"access-analyzer:ListAnalyzers\",  
\n\"access-analyzer:ListArchiveRules\", \"access-analyzer:ListFindings  
\n\", \"cloudtrail:DescribeTrails\", \"cloudtrail:GetEventDataStore\",  
\n\"cloudtrail:GetEventSelectors\", \"cloudtrail:GetInsightSelectors  
\n\", \"cloudtrail:GetTrailStatus\", \"cloudtrail:ListChannels\",  
\n\"cloudtrail:ListEventDataStores\", \"cloudtrail:ListQueries\", \"cloudtrail:ListTags  
\n\", \"cloudtrail:LookupEvents\", \"ec2:DescribeRegions\", \"iam:GetAccountSummary  
\n\", \"iam:GetOpenIDConnectProvider\", \"iam:GetRole\", \"iam:ListAccessKeys\",  
\n\"iam:ListAccountAliases\", \"iam:ListOpenIDConnectProviders\", \"iam:ListRoles  
\n\", \"iam:ListSAMLProviders\", \"kms:ListAliases\", \"s3:GetBucketLocation\",  
\n\"s3:ListAllMyBuckets\"]\", \"Resource\": \"*\"}]"  
      }  
    ],  
    "properties": {  
      "cloudTrailProperties": {  
        "endTime": "2024-02-14T22:44:40+00:00",  
        "startTime": "2024-02-13T00:30:00+00:00",  
        "trailProperties": [  
          {  
            "allRegions": true,  
            "cloudTrailArn": "arn:aws:cloudtrail:us-  
west-2:111122223333:trail/my-trail",  
            "regions": []  
          }  
        ]  
      }  
    },  
    "isComplete": false,  
    "principalArn": "arn:aws:iam::111122223333:role/Admin"  
  }  
}
```

```
  },
  "jobDetails": {
    "completedOn": "2024-02-14T22:47:01+00:00",
    "jobId": "c557dc4a-0338-4489-95dd-739014860ff9",
    "startedOn": "2024-02-14T22:44:41+00:00",
    "status": "SUCCEEDED"
  }
}
```

有关更多信息，请参阅 [IAM 用户指南中的 I AWS AM 访问分析器策略生成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetGeneratedPolicy](#) 中的。

## list-access-preview-findings

以下代码示例演示了如何使用 `list-access-preview-findings`。

### AWS CLI

检索由指定访问预览生成的访问预览结果列表

以下 `list-access-preview-findings` 示例检索您的 AWS 账户中指定访问预览生成的访问预览结果列表。

```
aws accessanalyzer list-access-preview-findings \
  --access-preview-id 3c65eb13-6ef9-4629-8919-a32043619e6b \
  --analyzer-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/
ConsoleAnalyzer-account
```

输出：

```
{
  "findings": [
    {
      "id": "e22fc158-1c87-4c32-9464-e7f405ce8d74",
      "principal": {
        "AWS": "111122223333"
      },
      "action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
    }
  ],
}
```

```

    "condition": {},
    "resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
    "isPublic": false,
    "resourceType": "AWS::S3::Bucket",
    "createdAt": "2024-02-17T00:18:46+00:00",
    "changeType": "NEW",
    "status": "ACTIVE",
    "resourceOwnerAccount": "111122223333",
    "sources": [
      {
        "type": "POLICY"
      }
    ]
  }
]
}

```

有关更多信息，请参阅 IAM 用户指南中的[使用 IAM Access Analyzer API 预览访问权限](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListAccessPreviewFindings](#)中的。

## list-access-previews

以下代码示例演示了如何使用 list-access-previews。

### AWS CLI

检索指定分析器的访问预览列表

以下 list-access-previews 示例检索您 AWS 账户中指定分析器的访问预览列表。

```

aws accessanalyzer list-access-previews \
  --analyzer-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/
  ConsoleAnalyzer-account

```

输出：

```

{
  "accessPreviews": [
    {
      "id": "3c65eb13-6ef9-4629-8919-a32043619e6b",
      "analyzerArn": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/
      ConsoleAnalyzer-account",

```

```
        "createdAt": "2024-02-17T00:18:44+00:00",
        "status": "COMPLETED"
    }
  ]
}
```

有关更多信息，请参阅 IAM 用户指南中的[使用 IAM Access Analyzer API 预览访问权限](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListAccessPreviews](#)中的。

## list-analyzed-resources

以下代码示例演示了如何使用 list-analyzed-resources。

### AWS CLI

列出可用的小部件

以下 list-analyzed-resources 示例列出了您 AWS 账户中可用的微件。

```
aws accessanalyzer list-analyzed-resources \
  --analyzer-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/
  ConsoleAnalyzer-account \
  --resource-type AWS::IAM::Role
```

输出：

```
{
  "analyzedResources": [
    {
      "resourceArn": "arn:aws:sns:us-west-2:111122223333:Validation-Email",
      "resourceOwnerAccount": "111122223333",
      "resourceType": "AWS::SNS::Topic"
    },
    {
      "resourceArn": "arn:aws:sns:us-west-2:111122223333:admin-alerts",
      "resourceOwnerAccount": "111122223333",
      "resourceType": "AWS::SNS::Topic"
    },
    {
      "resourceArn": "arn:aws:sns:us-west-2:111122223333:config-topic",
      "resourceOwnerAccount": "111122223333",
      "resourceType": "AWS::SNS::Topic"
    }
  ]
}
```

```
    },
    {
      "resourceArn": "arn:aws:sns:us-west-2:111122223333:inspector-topic",
      "resourceOwnerAccount": "111122223333",
      "resourceType": "AWS::SNS::Topic"
    }
  ]
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[使用 I AWS identity and Access Management 访问分析器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListAnalyzedResources](#)中的。

## list-analyzers

以下代码示例演示了如何使用 list-analyzers。

### AWS CLI

#### 检索分析器列表

以下 list-analyzers 示例检索您 AWS 账户中的分析器列表。

```
aws accessanalyzer list-analyzers
```

输出：

```
{
  "analyzers": [
    {
      "arn": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/UnusedAccess-ConsoleAnalyzer-organization",
      "createdAt": "2024-02-15T00:46:40+00:00",
      "name": "UnusedAccess-ConsoleAnalyzer-organization",
      "status": "ACTIVE",
      "tags": {
        "auto-delete": "no"
      },
      "type": "ORGANIZATION_UNUSED_ACCESS"
    },
    {
```

```

    "arn": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/
ConsoleAnalyzer-organization",
    "createdAt": "2020-04-25T07:43:28+00:00",
    "lastResourceAnalyzed": "arn:aws:s3::DOC-EXAMPLE-BUCKET",
    "lastResourceAnalyzedAt": "2024-02-15T21:51:56.517000+00:00",
    "name": "ConsoleAnalyzer-organization",
    "status": "ACTIVE",
    "tags": {
      "auto-delete": "no"
    },
    "type": "ORGANIZATION"
  },
  {
    "arn": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/
ConsoleAnalyzer-account",
    "createdAt": "2019-12-03T07:28:17+00:00",
    "lastResourceAnalyzed": "arn:aws:sns:us-west-2:111122223333:config-
topic",
    "lastResourceAnalyzedAt": "2024-02-15T18:01:53.003000+00:00",
    "name": "ConsoleAnalyzer-account",
    "status": "ACTIVE",
    "tags": {
      "auto-delete": "no"
    },
    "type": "ACCOUNT"
  }
]
}

```

有关更多信息，请参阅 AWS IAM 用户指南中的[使用 I AWS identity and Access Management 访问分析器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListAnalyzers](#)中的。

## list-archive-rules

以下代码示例演示了如何使用 list-archive-rules。

### AWS CLI

检索为指定分析器创建的存档规则列表

以下list-archive-rules示例检索您 AWS 账户中为指定分析器创建的存档规则列表。

```
aws accessanalyzer list-archive-rules \  
  --analyzer-name UnusedAccess-ConsoleAnalyzer-organization
```

输出：

```
{  
  "archiveRules": [  
    {  
      "createdAt": "2024-02-15T00:49:27+00:00",  
      "filter": {  
        "resource": {  
          "contains": [  
            "Cognito"  
          ]  
        },  
        "resourceType": {  
          "eq": [  
            "AWS::IAM::Role"  
          ]  
        }  
      },  
      "ruleName": "MyArchiveRule",  
      "updatedAt": "2024-02-15T00:49:27+00:00"  
    },  
    {  
      "createdAt": "2024-02-15T23:27:45+00:00",  
      "filter": {  
        "findingType": {  
          "eq": [  
            "UnusedIAMUserAccessKey"  
          ]  
        }  
      },  
      "ruleName": "ArchiveRule-56125a39-e517-4ff8-afb1-ef06f58db612",  
      "updatedAt": "2024-02-15T23:27:45+00:00"  
    }  
  ]  
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[使用 I AWS identity and Access Management 访问分析器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListArchiveRules](#)中的。

## list-findings-v2

以下代码示例演示了如何使用 `list-findings-v2`。

### AWS CLI

#### 检索指定分析器生成的结果列表

以下 `list-findings-v2` 示例检索您的 AWS 账户中指定分析器生成的结果列表。此示例筛选结果以仅包含名称包含的 IAM 角色 `Cognito`。

```
aws accessanalyzer list-findings-v2 \  
  --analyzer-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/  
ConsoleAnalyzer-account \  
  --filter '{"resource": {"contains": ["Cognito"]}, "resourceType": {"eq":  
["AWS::IAM::Role"]}]'
```

输出：

```
{  
  "findings": [  
    {  
      "analyzedAt": "2024-02-16T18:17:47.888000+00:00",  
      "createdAt": "2021-02-26T21:17:24.710000+00:00",  
      "id": "597f3bc2-3adc-4c18-9879-5c4b23485e46",  
      "resource": "arn:aws:iam::111122223333:role/  
Cognito_testpoolUnauth_Role",  
      "resourceType": "AWS::IAM::Role",  
      "resourceOwnerAccount": "111122223333",  
      "status": "ACTIVE",  
      "updatedAt": "2021-02-26T21:17:24.710000+00:00",  
      "findingType": "ExternalAccess"  
    },  
    {  
      "analyzedAt": "2024-02-16T18:17:47.888000+00:00",  
      "createdAt": "2021-02-26T21:17:50.905000+00:00",  
      "id": "ce0e221a-85b9-4d52-91ff-d7678075442f",  
      "resource": "arn:aws:iam::111122223333:role/Cognito_testpoolAuth_Role",  
      "resourceType": "AWS::IAM::Role",  
      "resourceOwnerAccount": "111122223333",  
      "status": "ACTIVE",  
      "updatedAt": "2021-02-26T21:17:50.905000+00:00",  
      "findingType": "ExternalAccess"  
    }  
  ]  
}
```



```
    }
  ]
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[使用 I AWS identity and Access Management 访问分析器](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的[ListFindingsV2](#)。

## list-findings

以下代码示例演示了如何使用 list-findings。

### AWS CLI

检索指定分析器生成的结果列表

以下list-findings示例检索您的 AWS 账户中指定分析器生成的结果列表。此示例筛选结果以仅包含名称包含的 IAM 角色Cognito。

```
aws accessanalyzer list-findings \
  --analyzer-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/
  ConsoleAnalyzer-account \
  --filter '{"resource": {"contains": ["Cognito"]}, "resourceType": {"eq":
  ["AWS::IAM::Role"]}]'
```

输出：

```
{
  "findings": [
    {
      "id": "597f3bc2-3adc-4c18-9879-5c4b23485e46",
      "principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "action": [
        "sts:AssumeRoleWithWebIdentity"
      ],
      "resource": "arn:aws:iam::111122223333:role/
      Cognito_testpoolUnauth_Role",
      "isPublic": false,
```

```

    "resourceType": "AWS::IAM::Role",
    "condition": {
      "cognito-identity.amazonaws.com:aud": "us-
west-2:EXAMPLE0-0000-0000-0000-000000000000"
    },
    "createdAt": "2021-02-26T21:17:24.710000+00:00",
    "analyzedAt": "2024-02-16T18:17:47.888000+00:00",
    "updatedAt": "2021-02-26T21:17:24.710000+00:00",
    "status": "ACTIVE",
    "resourceOwnerAccount": "111122223333"
  },
  {
    "id": "ce0e221a-85b9-4d52-91ff-d7678075442f",
    "principal": {
      "Federated": "cognito-identity.amazonaws.com"
    },
    "action": [
      "sts:AssumeRoleWithWebIdentity"
    ],
    "resource": "arn:aws:iam::111122223333:role/Cognito_testpoolAuth_Role",
    "isPublic": false,
    "resourceType": "AWS::IAM::Role",
    "condition": {
      "cognito-identity.amazonaws.com:aud": "us-
west-2:EXAMPLE0-0000-0000-0000-000000000000"
    },
    "createdAt": "2021-02-26T21:17:50.905000+00:00",
    "analyzedAt": "2024-02-16T18:17:47.888000+00:00",
    "updatedAt": "2021-02-26T21:17:50.905000+00:00",
    "status": "ACTIVE",
    "resourceOwnerAccount": "111122223333"
  }
]
}

```

有关更多信息，请参阅 AWS IAM 用户指南中的[使用 I AWS identity and Access Management 访问分析器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListFindings](#)中的。

## list-policy-generations

以下代码示例演示了如何使用 list-policy-generations。

## AWS CLI

列出过去七天内申请的所有保单世代

以下`list-policy-generations`示例列出了过去七天内您 AWS 账户中请求的所有保单生成。

```
aws accessanalyzer list-policy-generations
```

输出：

```
{
  "policyGenerations": [
    {
      "completedOn": "2024-02-14T23:43:38+00:00",
      "jobId": "923a56b0-ebb8-4e80-8a3c-a11ccfbcd6f2",
      "principalArn": "arn:aws:iam::111122223333:role/Admin",
      "startedOn": "2024-02-14T23:43:02+00:00",
      "status": "CANCELED"
    },
    {
      "completedOn": "2024-02-14T22:47:01+00:00",
      "jobId": "c557dc4a-0338-4489-95dd-739014860ff9",
      "principalArn": "arn:aws:iam::111122223333:role/Admin",
      "startedOn": "2024-02-14T22:44:41+00:00",
      "status": "SUCCEEDED"
    }
  ]
}
```

有关更多信息，请参阅 [IAM 用户指南中的 I AWS AM 访问分析器策略生成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListPolicyGenerations](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

## AWS CLI

检索应用于指定资源的标签列表

以下`list-tags-for-resource`示例检索应用于您 AWS 账户中指定资源的标签列表。

```
aws accessanalyzer list-tags-for-resource \  
  --resource-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/  
  ConsoleAnalyzer-account
```

输出：

```
{  
  "tags": {  
    "Zone-of-trust": "Account",  
    "Name": "ConsoleAnalyzer"  
  }  
}
```

有关更多信息，请参阅 [IAM 用户指南中的 I AWS AM 访问分析器策略生成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## start-policy-generation

以下代码示例演示了如何使用 start-policy-generation。

### AWS CLI

启动策略生成请求

以下 start-policy-generation 示例在您的 AWS 账户中启动策略生成请求。

```
aws accessanalyzer start-policy-generation \  
  --policy-generation-details '{"principalArn":"arn:aws:iam::111122223333:role/  
  Admin"}' \  
  --cloud-trail-details file://myfile.json
```

myfile.json 的内容：

```
{  
  "accessRole": "arn:aws:iam::111122223333:role/service-role/  
  AccessAnalyzerMonitorServiceRole",  
  "startTime": "2024-02-13T00:30:00Z",  
  "trails": [  
    {
```

```
        "allRegions": true,
        "cloudTrailArn": "arn:aws:cloudtrail:us-west-2:111122223333:trail/my-
trail"
      }
    ]
  }
}
```

输出：

```
{
  "jobId": "c557dc4a-0338-4489-95dd-739014860ff9"
}
```

有关更多信息，请参阅 [IAM 用户指南中的 | AWS AM 访问分析器策略生成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartPolicyGeneration](#) 中的。

## start-resource-scan

以下代码示例演示了如何使用 start-resource-scan。

### AWS CLI

立即开始扫描应用于指定资源的策略

以下 start-resource-scan 示例立即开始扫描应用于您 AWS 账户中指定资源的策略。

```
aws accessanalyzer start-resource-scan \
  --analyzer-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/
ConsoleAnalyzer-account \
  --resource-arn arn:aws:iam::111122223333:role/Cognito_testpoolAuth_Role
```

此命令不生成任何输出。

有关更多信息，请参阅 [IAM 用户指南中的 | AWS AM 访问分析器策略生成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartResourceScan](#) 中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

## AWS CLI

为指定资源添加标签

以下tag-resource示例将标签添加到您 AWS 账户中的指定资源。

```
aws accessanalyzer tag-resource \  
  --resource-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/  
ConsoleAnalyzer-account \  
  --tags Environment=dev,Purpose=testing
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IAM 用户指南中的[使用 | AWS identity and Access Management 访问分析器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

## AWS CLI

从指定资源中移除标签

以下untag-resource示例从您 AWS 账户中的指定资源中移除标签。

```
aws accessanalyzer untag-resource \  
  --resource-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/  
ConsoleAnalyzer-account \  
  --tag-keys Environment Purpose
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IAM 用户指南中的[使用 | AWS identity and Access Management 访问分析器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-archive-rule

以下代码示例演示了如何使用 update-archive-rule。

## AWS CLI

### 更新指定存档规则的条件和值

以下update-archive-rule示例更新了您 AWS 账户中指定存档规则的条件和值。

```
aws accessanalyzer update-archive-rule \  
  --analyzer-name UnusedAccess-ConsoleAnalyzer-organization \  
  --rule-name MyArchiveRule \  
  --filter '{"resource": {"contains": ["Cognito"]}, "resourceType": {"eq":  
  ["AWS::IAM::Role"]}]'
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IAM 用户指南中的[存档规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateArchiveRule](#)中的。

## update-findings

以下代码示例演示了如何使用 update-findings。

### AWS CLI

#### 更新指定发现的状态

以下update-findings示例更新了您 AWS 账户中指定发现的状态。

```
aws accessanalyzer update-findings \  
  --analyzer-arn arn:aws:access-analyzer:us-west-2:111122223333:analyzer/  
UnusedAccess-ConsoleAnalyzer-organization \  
  --ids 4f319ac3-2e0c-4dc4-bf51-7013a086b6ae 780d586a-2cce-4f72-aff6-359d450e7500  
 \  
  --status ARCHIVED
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IAM 用户指南中的[使用 I AWS identity and Access Management 访问分析器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateFindings](#)中的。

## validate-policy

以下代码示例演示了如何使用 `validate-policy`。

### AWS CLI

请求验证策略并返回调查结果列表

以下 `validate-policy` 示例请求验证策略并返回结果列表。示例中的策略是用于网络联合身份验证的 Amazon Cognito 角色的角色信任策略。信任策略生成的结果与空 `Sid` 元素值和不匹配的策略主体有关，这是因为使用的代入角色操作不正确，`sts:AssumeRole` 与 Cognito 一起使用的正确假设角色操作是 `sts:AssumeRoleWithWebIdentity`

```
aws accessanalyzer validate-policy \  
  --policy-document file://myfile.json \  
  --policy-type RESOURCE_POLICY
```

`myfile.json` 的内容：

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "",  
      "Effect": "Allow",  
      "Principal": {  
        "Federated": "cognito-identity.amazonaws.com"  
      },  
      "Action": [  
        "sts:AssumeRole",  
        "sts:TagSession"  
      ],  
      "Condition": {  
        "StringEquals": {  
          "cognito-identity.amazonaws.com:aud": "us-west-2_EXAMPLE"  
        }  
      }  
    }  
  ]  
}
```

输出：



```
{
  "findings": [
    {
      "findingDetails": "Add a value to the empty string in the Sid element.",
      "findingType": "SUGGESTION",
      "issueCode": "EMPTY_SID_VALUE",
      "learnMoreLink": "https://docs.aws.amazon.com/IAM/latest/UserGuide/
access-analyzer-reference-policy-checks.html#access-analyzer-reference-policy-
checks-suggestion-empty-sid-value",
      "locations": [
        {
          "path": [
            {
              "value": "Statement"
            },
            {
              "index": 0
            },
            {
              "value": "Sid"
            }
          ],
          "span": {
            "end": {
              "column": 21,
              "line": 5,
              "offset": 81
            },
            "start": {
              "column": 19,
              "line": 5,
              "offset": 79
            }
          }
        }
      ]
    },
    {
      "findingDetails": "The sts:AssumeRole action is invalid with the
following principal(s): cognito-identity.amazonaws.com. Use a SAML provider
principal with the sts:AssumeRoleWithSAML action or use an OIDC provider principal
with the sts:AssumeRoleWithWebIdentity action. Ensure the provider is Federated if
you use either of the two options.",
```

```
    "findingType": "ERROR",
    "issueCode": "MISMATCHED_ACTION_FOR_PRINCIPAL",
    "learnMoreLink": "https://docs.aws.amazon.com/IAM/latest/UserGuide/
access-analyzer-reference-policy-checks.html#access-analyzer-reference-policy-
checks-error-mismatched-action-for-principal",
    "locations": [
      {
        "path": [
          {
            "value": "Statement"
          },
          {
            "index": 0
          },
          {
            "value": "Action"
          },
          {
            "index": 0
          }
        ],
        "span": {
          "end": {
            "column": 32,
            "line": 11,
            "offset": 274
          },
          "start": {
            "column": 16,
            "line": 11,
            "offset": 258
          }
        }
      },
      {
        "path": [
          {
            "value": "Statement"
          },
          {
            "index": 0
          },
          {
            "value": "Principal"
          }
        ]
      }
    ]
  }
}
```

```

    },
    {
      "value": "Federated"
    }
  ],
  "span": {
    "end": {
      "column": 61,
      "line": 8,
      "offset": 202
    },
    "start": {
      "column": 29,
      "line": 8,
      "offset": 170
    }
  }
}
]
},
{
  "findingDetails": "The following actions: sts:TagSession are not
supported by the condition key cognito-identity.amazonaws.com:aud. The condition
will not be evaluated for these actions. We recommend that you move these actions
to a different statement without this condition key.",
  "findingType": "ERROR",
  "issueCode": "UNSUPPORTED_ACTION_FOR_CONDITION_KEY",
  "learnMoreLink": "https://docs.aws.amazon.com/IAM/latest/UserGuide/
access-analyzer-reference-policy-checks.html#access-analyzer-reference-policy-
checks-error-unsupported-action-for-condition-key",
  "locations": [
    {
      "path": [
        {
          "value": "Statement"
        },
        {
          "index": 0
        },
        {
          "value": "Action"
        },
        {
          "index": 1
        }
      ]
    }
  ]
}

```

```
    }
  ],
  "span": {
    "end": {
      "column": 32,
      "line": 12,
      "offset": 308
    },
    "start": {
      "column": 16,
      "line": 12,
      "offset": 292
    }
  }
},
{
  "path": [
    {
      "value": "Statement"
    },
    {
      "index": 0
    },
    {
      "value": "Condition"
    },
    {
      "value": "StringEquals"
    },
    {
      "value": "cognito-identity.amazonaws.com:aud"
    }
  ],
  "span": {
    "end": {
      "column": 79,
      "line": 16,
      "offset": 464
    },
    "start": {
      "column": 58,
      "line": 16,
      "offset": 443
    }
  }
}
```

```
}  
    ]  
  }  
}
```

有关更多信息，请参阅 AWS IAM 用户指南中的[策略验证检查](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ValidatePolicy](#) 中的。

## 使用的 Image Builder 示例 AWS CLI

以下代码示例向您展示了如何使用与 Image Builder AWS Command Line Interface 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### create-component

以下代码示例演示了如何使用 create-component。

AWS CLI

创建组件

以下 create-component 示例创建一个使用 JSON 文档文件的组件，并引用上传到 Amazon S3 存储桶的 YAML 格式的组件文档。

```
aws imagebuilder create-component \  
  --cli-input-json file://create-component.json
```

`create-component.json` 的内容：

```
{
  "name": "MyExampleComponent",
  "semanticVersion": "2019.12.02",
  "description": "An example component that builds, validates and tests an image",
  "changeDescription": "Initial version.",
  "platform": "Windows",
  "uri": "s3://s3-bucket-name/s3-bucket-path/component.yaml"
}
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "componentBuildVersionArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/examplecomponent/2019.12.02/1"
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateComponent](#) 中的。

## `create-distribution-configuration`

以下代码示例演示了如何使用 `create-distribution-configuration`。

### AWS CLI

#### 创建分发配置

以下 `create-distribution-configuration` 示例使用 JSON 文件创建分发配置。

```
aws imagebuilder create-distribution-configuration \
  --cli-input-json file:/create-distribution-configuration.json
```

`create-distribution-configuration.json` 的内容：

```
{
  "name": "MyExampleDistribution",
```

```

"description": "Copies AMI to eu-west-1",
"distributions": [
  {
    "region": "us-west-2",
    "amiDistributionConfiguration": {
      "name": "Name {{imagebuilder:buildDate}}",
      "description": "An example image name with parameter references",
      "amiTags": {
        "KeyName": "{{ssm:parameter_name}}"
      },
      "launchPermission": {
        "userIds": [
          "123456789012"
        ]
      }
    }
  },
  {
    "region": "eu-west-1",
    "amiDistributionConfiguration": {
      "name": "My {{imagebuilder:buildVersion}} image
{{imagebuilder:buildDate}}",
      "amiTags": {
        "KeyName": "Value"
      },
      "launchPermission": {
        "userIds": [
          "123456789012"
        ]
      }
    }
  }
]
}

```

输出：

```

{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/myexampledistribution"
}

```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDistributionConfiguration](#) 中的。

## create-image-pipeline

以下代码示例演示了如何使用 create-image-pipeline。

### AWS CLI

#### 创建图像管道

以下 create-image-pipeline 示例使用 JSON 文件创建图像管道。

```
aws imagebuilder create-image-pipeline \  
  --cli-input-json file://create-image-pipeline.json
```

create-image-pipeline.json 的内容：

```
{  
  "name": "MyWindows2016Pipeline",  
  "description": "Builds Windows 2016 Images",  
  "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/  
mybasicrecipe/2019.12.03",  
  "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-  
west-2:123456789012:infrastructure-configuration/myexampleinfrastructure",  
  "distributionConfigurationArn": "arn:aws:imagebuilder:us-  
west-2:123456789012:distribution-configuration/myexampledistribution",  
  "imageTestsConfiguration": {  
    "imageTestsEnabled": true,  
    "timeoutMinutes": 60  
  },  
  "schedule": {  
    "scheduleExpression": "cron(0 0 * * SUN)",  
    "pipelineExecutionStartCondition":  
"EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"  
  },  
  "status": "ENABLED"  
}
```

输出：



```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "imagePipelineArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/mywindows2016pipeline"
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateImagePipeline](#) 中的。

## create-image-recipe

以下代码示例演示了如何使用 create-image-recipe。

### AWS CLI

#### 创建食谱

以下 create-image-recipe 示例使用 JSON 文件创建图像配方。组件是按指定顺序安装的。

```
aws imagebuilder create-image-recipe \
  --cli-input-json file://create-image-recipe.json
```

create-image-recipe.json 的内容：

```
{
  "name": "MyBasicRecipe",
  "description": "This example image recipe creates a Windows 2016 image.",
  "semanticVersion": "2019.12.03",
  "components":
  [
    {
      "componentArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/myexamplecomponent/2019.12.02/1"
    },
    {
      "componentArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/myimportedcomponent/1.0.0/1"
    }
  ],
}
```

```
"parentImage": "arn:aws:imagebuilder:us-west-2:aws:image/windows-server-2016-english-full-base-x86/xxxx.x.x"
}
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/mybasicrecipe/2019.12.03"
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateImageRecipe](#) 中的。

## create-image

以下代码示例演示了如何使用 create-image。

### AWS CLI

#### 创建镜像

以下 create-image 示例创建了一张图像。

```
aws imagebuilder create-image \
  --image-recipe-arn arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/mybasicrecipe/2019.12.03 \
  --infrastructure-configuration-arn arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/myexampleinfrastructure
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "imageBuildVersionArn": "arn:aws:imagebuilder:us-west-2:123456789012:image/mybasicrecipe/2019.12.03/1"
}
```

```
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateImage](#) 中的。

## create-infrastructure-configuration

以下代码示例演示了如何使用 create-infrastructure-configuration。

### AWS CLI

#### 创建基础架构配置

以下 create-infrastructure-configuration 示例使用 JSON 文件创建基础设施配置。

```
aws imagebuilder create-infrastructure-configuration \  
  --cli-input-json file://create-infrastructure-configuration.json
```

create-infrastructure-configuration.json 的内容：

```
{  
  "name": "MyExampleInfrastructure",  
  "description": "An example that will retain instances of failed builds",  
  "instanceTypes": [  
    "m5.large", "m5.xlarge"  
  ],  
  "instanceProfileName": "EC2InstanceProfileForImageBuilder",  
  "securityGroupIds": [  
    "sg-a1b2c3d4"  
  ],  
  "subnetId": "subnet-a1b2c3d4",  
  "logging": {  
    "s3Logs": {  
      "s3BucketName": "bucket-name",  
      "s3KeyPrefix": "bucket-path"  
    }  
  },  
  "keyPair": "key-pair-name",  
  "terminateInstanceOnFailure": false,  
  "snsTopicArn": "arn:aws:sns:us-west-2:123456789012:sns-topic-name"
```

```
}
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/myexampleinfrastructure"
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateInfrastructureConfiguration](#) 中的。

## delete-component

以下代码示例演示了如何使用 delete-component。

### AWS CLI

#### 删除组件

以下 delete-component 示例通过指定组件编译版本的 ARN 来删除该版本。

```
aws imagebuilder delete-component \
  --component-build-version-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/myexamplecomponent/2019.12.02/1
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "componentBuildVersionArn": "arn:aws:imagebuilder:us-
west-2:123456789012:component/myexamplecomponent/2019.12.02/1"
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteComponent](#) 中的。

## delete-image-pipeline

以下代码示例演示了如何使用 delete-image-pipeline。

### AWS CLI

#### 删除图像管道

以下delete-image-pipeline示例通过指定图像管道的 ARN 来删除该管道。

```
aws imagebuilder delete-image-pipeline \  
  --image-pipeline-arn arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/  
  my-example-pipeline
```

输出：

```
{  
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
  "imagePipelineArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/  
  mywindows2016pipeline"  
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteImagePipeline](#)中的。

## delete-image-recipe

以下代码示例演示了如何使用 delete-image-recipe。

### AWS CLI

#### 删除图片配方

以下delete-image-recipe示例通过指定图像配方的 ARN 来删除该图像配方。

```
aws imagebuilder delete-image-recipe \  
  --image-recipe-arn arn:aws:imagebuilder:us-east-1:123456789012:image-recipe/  
  mybasicrecipe/2019.12.03
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/
mybasicrecipe/2019.12.03"
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteImageRecipe](#) 中的。

## delete-image

以下代码示例演示了如何使用 delete-image。

### AWS CLI

要删除图像

以下 delete-image 示例通过指定映像版本的 ARN 来删除该版本的 ARN。

```
aws imagebuilder delete-image \
  --image-build-version-arn arn:aws:imagebuilder:us-west-2:123456789012:image/my-
example-image/2019.12.02/1
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "imageBuildVersionArn": "arn:aws:imagebuilder:us-west-2:123456789012:image/
mybasicrecipe/2019.12.03/1"
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteImage](#) 中的。

## delete-infrastructure-configuration

以下代码示例演示了如何使用 delete-infrastructure-configuration。

## AWS CLI

### 删除基础架构配置

以下delete-infrastructure-configuration示例通过指定图像管道的 ARN 来删除该管道。

```
aws imagebuilder delete-infrastructure-configuration \  
  --infrastructure-configuration-arn arn:aws:imagebuilder:us-  
east-1:123456789012:infrastructure-configuration/myexampleinfrastructure
```

输出：

```
{  
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
  "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-  
west-2:123456789012:infrastructure-configuration/myexampleinfrastructure"  
}
```

有关更多信息，请参阅 [《EC2 Image Builder 用户指南》](#) 中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInfrastructureConfiguration](#)中的。

### get-component-policy

以下代码示例演示了如何使用 get-component-policy。

## AWS CLI

### 获取组件策略详细信息

以下get-component-policy示例通过指定 ARN 列出了组件策略的详细信息。

```
aws imagebuilder get-component-policy \  
  --component-arn arn:aws:imagebuilder:us-west-2:123456789012:component/my-  
example-component/2019.12.03/1
```

输出：

```
{
```

```
"Policy": "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Effect\":
\"Allow\", \"Principal\": { \"AWS\": [ \"123456789012\" ] }, \"Action\":
[ \"imagebuilder:GetComponent\", \"imagebuilder:ListComponents\" ], \"Resource\":
[ \"arn:aws:imagebuilder:us-west-2:123456789012:component/my-example-
component/2019.12.03/1\" ] } ] }"
}
```

有关更多信息，请参阅 EC2 Image Builder 用户指南中的使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道 < <https://docs.aws.amazon.com/imagebuilder/latest/userguide/managing-image-builder-cli.html> > 。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetComponentPolicy](#) 中的。

## get-component

以下代码示例演示了如何使用 get-component。

### AWS CLI

#### 获取组件详细信息

以下 get-component 示例通过指定组件的 ARN 列出了组件的详细信息。

```
aws imagebuilder get-component \
  --component-build-version-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/component-name/1.0.0/1
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "component": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/component-
name/1.0.0/1",
    "name": "component-name",
    "version": "1.0.0",
    "type": "TEST",
    "platform": "Linux",
    "owner": "123456789012",
    "data": "name: HelloWorldTestingDocument\ndescription: This is hello world
testing document.\nschemaVersion: 1.0\n\nphases:\n - name: test\n   steps:\n
- name: HelloWorldStep\n       action: ExecuteBash\n       inputs:\n
commands:\n       - echo \"Hello World! Test.\\\"\\n\",
```



```

    "encrypted": true,
    "dateCreated": "2020-01-27T20:43:30.306Z",
    "tags": {}
  }
}

```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetComponent](#) 中的。

## get-distribution-configuration

以下代码示例演示了如何使用 get-distribution-configuration。

### AWS CLI

获取分发配置的详细信息

以下 get-distribution-configuration 示例通过指定 ARN 来显示分发配置的详细信息。

```

aws imagebuilder get-distribution-configuration \
  --distribution-configuration-arn arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/myexampledistribution

```

输出：

```

{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "distributionConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:distribution-
configuration/myexampledistribution",
    "name": "MyExampleDistribution",
    "description": "Copies AMI to eu-west-1 and exports to S3",
    "distributions": [
      {
        "region": "us-west-2",
        "amiDistributionConfiguration": {
          "name": "Name {{imagebuilder:buildDate}}",
          "description": "An example image name with parameter
references",
          "amiTags": {
            "KeyName": "{{ssm:parameter_name}}"

```

```

        },
        "launchPermission": {
            "userIds": [
                "123456789012"
            ]
        }
    },
    {
        "region": "eu-west-1",
        "amiDistributionConfiguration": {
            "name": "My {{imagebuilder:buildVersion}} image
{{imagebuilder:buildDate}}",
            "amiTags": {
                "KeyName": "Value"
            },
            "launchPermission": {
                "userIds": [
                    "123456789012"
                ]
            }
        }
    }
],
"dateCreated": "2020-02-19T18:40:10.529Z",
"tags": {}
}
}

```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDistributionConfiguration](#) 中的。

## get-image-pipeline

以下代码示例演示了如何使用 `get-image-pipeline`。

### AWS CLI

获取图像管道详细信息

以下 `get-image-pipeline` 示例通过指定 ARN 列出了图像管道的详细信息。

```
aws imagebuilder get-image-pipeline \  
  --image-pipeline-arn arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/  
mywindows2016pipeline
```

输出：

```
{  
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
  "imagePipeline": {  
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/  
mywindows2016pipeline",  
    "name": "MyWindows2016Pipeline",  
    "description": "Builds Windows 2016 Images",  
    "platform": "Windows",  
    "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/  
mybasicrecipe/2019.12.03",  
    "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-  
west-2:123456789012:infrastructure-configuration/myexampleinfrastructure",  
    "distributionConfigurationArn": "arn:aws:imagebuilder:us-  
west-2:123456789012:distribution-configuration/myexampledistribution",  
    "imageTestsConfiguration": {  
      "imageTestsEnabled": true,  
      "timeoutMinutes": 60  
    },  
    "schedule": {  
      "scheduleExpression": "cron(0 0 * * SUN)",  
      "pipelineExecutionStartCondition":  
"EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"  
    },  
    "status": "ENABLED",  
    "dateCreated": "2020-02-19T19:04:01.253Z",  
    "dateUpdated": "2020-02-19T19:04:01.253Z",  
    "tags": {}  
  }  
}
```

有关更多信息，请参阅 [《EC2 Image Builder 用户指南》](#) 中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetImagePipeline](#) 中的。

## get-image-policy

以下代码示例演示了如何使用 `get-image-policy`。

### AWS CLI

获取图片政策详情

以下`get-image-policy`示例通过指定 ARN 列出了图片策略的详细信息。

```
aws imagebuilder get-image-policy \  
  --image-arn arn:aws:imagebuilder:us-west-2:123456789012:image/my-example-  
image/2019.12.03/1
```

输出：

```
{  
  "Policy": "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Effect\": \"Allow\",  
  \"Principal\": { \"AWS\": [ \"123456789012\" ] }, \"Action\": [ \"imagebuilder:GetImage\",  
  \"imagebuilder:ListImages\" ], \"Resource\": [ \"arn:aws:imagebuilder:us-  
west-2:123456789012:image/my-example-image/2019.12.03/1\" ] } ] }"
```

有关更多信息，请参阅 [《EC2 Image Builder 用户指南》](#) 中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetImagePolicy](#) 中的。

## get-image-recipe-policy

以下代码示例演示了如何使用 `get-image-recipe-policy`。

### AWS CLI

获取图片配方政策详情

以下`get-image-recipe-policy`示例通过指定 ARN 列出了图片配方策略的详细信息。

```
aws imagebuilder get-image-recipe-policy \  
  --image-recipe-arn arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-  
example-image-recipe/2019.12.03/1
```

输出：

```
{
  "Policy": "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Effect\":
  \"Allow\", \"Principal\": { \"AWS\": [ \"123456789012\" ] }, \"Action\":
  [ \"imagebuilder:GetImageRecipe\", \"imagebuilder:ListImageRecipes\" ], \"Resource\":
  [ \"arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-image-
  recipe/2019.12.03/1\" ] } ] }"
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetImageRecipePolicy](#) 中的。

## get-image

以下代码示例演示了如何使用 get-image。

### AWS CLI

获取图片详情

以下 get-image 示例通过指定 ARN 列出图像的详细信息。

```
aws imagebuilder get-image \
  --image-build-version-arn arn:aws:imagebuilder:us-west-2:123456789012:image/
  mybasicrecipe/2019.12.03/1
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "image": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/
  mybasicrecipe/2019.12.03/1",
    "name": "MyBasicRecipe",
    "version": "2019.12.03/1",
    "platform": "Windows",
    "state": {
      "status": "BUILDING"
    },
    "imageRecipe": {
```

```
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/
mybasicrecipe/2019.12.03",
    "name": "MyBasicRecipe",
    "description": "This example image recipe creates a Windows 2016
image.",
    "platform": "Windows",
    "version": "2019.12.03",
    "components": [
      {
        "componentArn": "arn:aws:imagebuilder:us-
west-2:123456789012:component/myexamplecomponent/2019.12.02/1"
      },
      {
        "componentArn": "arn:aws:imagebuilder:us-
west-2:123456789012:component/myimportedcomponent/1.0.0/1"
      }
    ],
    "parentImage": "arn:aws:imagebuilder:us-west-2:aws:image/windows-
server-2016-english-full-base-x86/2019.12.17/1",
    "dateCreated": "2020-02-14T19:46:16.904Z",
    "tags": {}
  },
  "infrastructureConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-
configuration/myexampleinfrastructure",
    "name": "MyExampleInfrastructure",
    "description": "An example that will retain instances of failed builds",
    "instanceTypes": [
      "m5.large",
      "m5.xlarge"
    ],
    "instanceProfileName": "EC2InstanceProfileForImageFactory",
    "securityGroupIds": [
      "sg-a1b2c3d4"
    ],
    "subnetId": "subnet-a1b2c3d4",
    "logging": {
      "s3Logs": {
        "s3BucketName": "bucket-name",
        "s3KeyPrefix": "bucket-path"
      }
    }
  },
  "keyPair": "Sam",
  "terminateInstanceOnFailure": false,
```

```
    "snsTopicArn": "arn:aws:sns:us-west-2:123456789012:sns-name",
    "dateCreated": "2020-02-14T21:21:05.098Z",
    "tags": {}
  },
  "imageTestsConfiguration": {
    "imageTestsEnabled": true,
    "timeoutMinutes": 720
  },
  "dateCreated": "2020-02-14T23:14:13.597Z",
  "outputResources": {
    "amis": []
  },
  "tags": {}
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetImage](#) 中的。

## get-infrastructure-configuration

以下代码示例演示了如何使用 `get-infrastructure-configuration`。

### AWS CLI

获取基础架构配置的详细信息

以下 `get-infrastructure-configuration` 示例通过指定 ARN 列出了基础设施配置的详细信息。

```
aws imagebuilder get-infrastructure-configuration \
  --infrastructure-configuration-arn arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/myexampleinfrastructure
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "infrastructureConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-
configuration/myexampleinfrastructure",
```

```
    "name": "MyExampleInfrastructure",
    "description": "An example that will retain instances of failed builds",
    "instanceTypes": [
      "m5.large",
      "m5.xlarge"
    ],
    "instanceProfileName": "EC2InstanceProfileForImageBuilder",
    "securityGroupIds": [
      "sg-a48c95ef"
    ],
    "subnetId": "subnet-a48c95ef",
    "logging": {
      "s3Logs": {
        "s3BucketName": "bucket-name",
        "s3KeyPrefix": "bucket-path"
      }
    },
    "keyPair": "Name",
    "terminateInstanceOnFailure": false,
    "snsTopicArn": "arn:aws:sns:us-west-2:123456789012:sns-name",
    "dateCreated": "2020-02-19T19:11:51.858Z",
    "tags": {}
  }
}
```

有关更多信息，请参阅 [《EC2 Image Builder 用户指南》](#) 中的“使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetInfrastructureConfiguration](#) 中的。

## import-component

以下代码示例演示了如何使用 import-component。

### AWS CLI

#### 导入组件

以下 import-component 示例使用 JSON 文件导入先前存在的脚本。

```
aws imagebuilder import-component \  
  --cli-input-json file://import-component.json
```



`import-component.json` 的内容：

```
{
  "name": "MyImportedComponent",
  "semanticVersion": "1.0.0",
  "description": "An example of how to import a component",
  "changeDescription": "First commit message.",
  "format": "SHELL",
  "platform": "Windows",
  "type": "BUILD",
  "uri": "s3://s3-bucket-name/s3-bucket-path/component.yaml"
}
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "componentBuildVersionArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/myimportedcomponent/1.0.0/1"
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ImportComponent](#) 中的。

## `list-component-build-versions`

以下代码示例演示了如何使用 `list-component-build-versions`。

AWS CLI

列出组件版本版本

以下 `list-component-build-versions` 示例列出了具有特定语义版本的组件版本号。

```
aws imagebuilder list-component-build-versions --component-version-arn
arn:aws:imagebuilder:us-west-2:123456789012:component/myexamplecomponent/2019.12.02
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "componentSummaryList": [
    {
      "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/myexamplecomponent/2019.12.02/1",
      "name": "MyExampleComponent",
      "version": "2019.12.02",
      "platform": "Windows",
      "type": "BUILD",
      "owner": "123456789012",
      "description": "An example component that builds, validates and tests an image",
      "changeDescription": "Initial version.",
      "dateCreated": "2020-02-19T18:53:45.940Z",
      "tags": {
        "KeyName": "KeyValue"
      }
    }
  ]
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListComponentBuildVersions](#) 中的。

## list-components

以下代码示例演示了如何使用 list-components。

### AWS CLI

列出所有组件语义版本

以下 list-components 示例列出了您有权访问的所有组件语义版本。您可以选择筛选是否列出您拥有的组件、由 Amazon 拥有的组件，还是其他账户与您共享的组件。

```
aws imagebuilder list-components
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "componentVersionList": [
    {
      "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/component-
name/1.0.0",
      "name": "component-name",
      "version": "1.0.0",
      "platform": "Linux",
      "type": "TEST",
      "owner": "123456789012",
      "dateCreated": "2020-01-27T20:43:30.306Z"
    }
  ]
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListComponents](#) 中的。

## list-distribution-configurations

以下代码示例演示了如何使用 list-distribution-configurations。

### AWS CLI

列出发行版

以下 list-distribution-configurations 示例列出了您的所有发行版。

```
aws imagebuilder list-distribution-configurations
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "distributionConfigurationSummaryList": [
    {
      "arn": "arn:aws:imagebuilder:us-west-2:123456789012:distribution-
configuration/myexampledistribution",
```

```
    "name": "MyExampleDistribution",
    "description": "Copies AMI to eu-west-1 and exports to S3",
    "dateCreated": "2020-02-19T18:40:10.529Z",
    "tags": {
      "KeyName": "KeyValue"
    }
  ]
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDistributionConfigurations](#) 中的。

## list-image-build-versions

以下代码示例演示了如何使用 `list-image-build-versions`。

### AWS CLI

列出映像版本版本

以下 `list-image-build-versions` 示例列出了所有具有语义版本的映像构建版本。

```
aws imagebuilder list-image-build-versions \
  --image-version-arn arn:aws:imagebuilder:us-west-2:123456789012:image/
mybasicrecipe/2019.12.03
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "imageSummaryList": [
    {
      "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/
mybasicrecipe/2019.12.03/7",
      "name": "MyBasicRecipe",
      "version": "2019.12.03/7",
      "platform": "Windows",
      "state": {
        "status": "FAILED",
```

```

        "reason": "Can't start SSM Automation for arn
arn:aws:imagebuilder:us-west-2:123456789012:image/mybasicrecipe/2019.12.03/7 during
building. Parameter \"iamInstanceProfileName\" has a null value."
    },
    "owner": "123456789012",
    "dateCreated": "2020-02-19T18:56:11.511Z",
    "outputResources": {
        "amis": []
    },
    },
    {
        "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/
mybasicrecipe/2019.12.03/6",
        "name": "MyBasicRecipe",
        "version": "2019.12.03/6",
        "platform": "Windows",
        "state": {
            "status": "FAILED",
            "reason": "An internal error has occurred."
        },
        "owner": "123456789012",
        "dateCreated": "2020-02-18T22:49:08.142Z",
        "outputResources": {
            "amis": [
                {
                    "region": "us-west-2",
                    "image": "ami-a1b2c3d4567890ab",
                    "name": "MyBasicRecipe 2020-02-18T22-49-38.704Z",
                    "description": "This example image recipe creates a Windows
2016 image."
                },
                {
                    "region": "us-west-2",
                    "image": "ami-a1b2c3d4567890ab",
                    "name": "Name 2020-02-18T22-49-08.131Z",
                    "description": "Copies AMI to eu-west-2 and exports to S3"
                },
                {
                    "region": "eu-west-2",
                    "image": "ami-a1b2c3d4567890ab",
                    "name": "My 6 image 2020-02-18T22-49-08.131Z",
                    "description": "Copies AMI to eu-west-2 and exports to S3"
                }
            ]
        }
    }
}

```

```
    ]
  },
  "tags": {}
},
{
  "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/
mybasicrecipe/2019.12.03/5",
  "name": "MyBasicRecipe",
  "version": "2019.12.03/5",
  "platform": "Windows",
  "state": {
    "status": "AVAILABLE"
  },
  "owner": "123456789012",
  "dateCreated": "2020-02-18T16:51:48.403Z",
  "outputResources": {
    "amis": [
      {
        "region": "us-west-2",
        "image": "ami-a1b2c3d4567890ab",
        "name": "MyBasicRecipe 2020-02-18T16-52-18.965Z",
        "description": "This example image recipe creates a Windows
2016 image."
      }
    ]
  },
  "tags": {}
},
{
  "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/
mybasicrecipe/2019.12.03/4",
  "name": "MyBasicRecipe",
  "version": "2019.12.03/4",
  "platform": "Windows",
  "state": {
    "status": "AVAILABLE"
  },
  "owner": "123456789012",
  "dateCreated": "2020-02-18T16:50:01.827Z",
  "outputResources": {
    "amis": [
      {
        "region": "us-west-2",
        "image": "ami-a1b2c3d4567890ab",
```

```

                "name": "MyBasicRecipe 2020-02-18T16-50-32.280Z",
                "description": "This example image recipe creates a Windows
2016 image."
            }
        ]
    },
    "tags": {}
},
{
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/
mybasicrecipe/2019.12.03/3",
    "name": "MyBasicRecipe",
    "version": "2019.12.03/3",
    "platform": "Windows",
    "state": {
        "status": "AVAILABLE"
    },
    "owner": "123456789012",
    "dateCreated": "2020-02-14T23:14:13.597Z",
    "outputResources": {
        "amis": [
            {
                "region": "us-west-2",
                "image": "ami-a1b2c3d4567890ab",
                "name": "MyBasicRecipe 2020-02-14T23-14-44.243Z",
                "description": "This example image recipe creates a Windows
2016 image."
            }
        ]
    },
    "tags": {}
},
{
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/
mybasicrecipe/2019.12.03/2",
    "name": "MyBasicRecipe",
    "version": "2019.12.03/2",
    "platform": "Windows",
    "state": {
        "status": "FAILED",
        "reason": "SSM execution 'a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'
failed with status = 'Failed' and failure message = 'Step fails when it is
verifying the command has completed. Command a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
returns unexpected invocation result: \n{Status=[Failed], ResponseCode=[1],

```

```

Output=[\n-----ERROR-----\nfailed to run commands: exit status 1],
OutputPayload=[{"Status\":"Failed","\nResponseCode\":1,\nOutput\":"\n-----ERROR-----\nfailed to run commands: exit status 1","\nCommandId\":"a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"}], CommandId=[a1b2c3d4-5678-90ab-cdef-EXAMPLE11111]}. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.'"
    },
    "owner": "123456789012",
    "dateCreated": "2020-02-14T22:57:42.593Z",
    "outputResources": {
      "amis": []
    },
    "tags": {}
  }
]
}

```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListImageBuildVersions](#) 中的。

## list-image-pipeline-images

以下代码示例演示了如何使用 `list-image-pipeline-images`。

### AWS CLI

列出图像管道管道图像

以下 `list-image-pipeline-images` 示例列出了由特定图像管道创建的所有图像。

```

aws imagebuilder list-image-pipeline-images \
  --image-pipeline-arn arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/
  mywindows2016pipeline

```

输出：

```

{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "imagePipelineList": [
    {

```



```
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/
mywindows2016pipeline",
    "name": "MyWindows2016Pipeline",
    "description": "Builds Windows 2016 Images",
    "platform": "Windows",
    "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-
recipe/mybasicrecipe/2019.12.03",
    "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/myexampleinfrastructure",
    "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/myexampledistribution",
    "imageTestsConfiguration": {
        "imageTestsEnabled": true,
        "timeoutMinutes": 60
    },
    "schedule": {
        "scheduleExpression": "cron(0 0 * * SUN)",
        "pipelineExecutionStartCondition":
"EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
    },
    "status": "ENABLED",
    "dateCreated": "2020-02-19T19:04:01.253Z",
    "dateUpdated": "2020-02-19T19:04:01.253Z",
    "tags": {
        "KeyName": "KeyValue"
    }
},
{
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/sam",
    "name": "PipelineName",
    "platform": "Linux",
    "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-
recipe/recipe-name-a1b2c3d45678/1.0.0",
    "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/infrastructureconfiguration-name-
a1b2c3d45678",
    "imageTestsConfiguration": {
        "imageTestsEnabled": true,
        "timeoutMinutes": 720
    },
    "status": "ENABLED",
    "dateCreated": "2019-12-16T18:19:02.068Z",
    "dateUpdated": "2019-12-16T18:19:02.068Z",
    "tags": {
```

```
        "KeyName": "KeyValue"
      }
    }
  ]
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListImagePipelineImages](#) 中的。

## list-image-recipes

以下代码示例演示了如何使用 `list-image-recipes`。

### AWS CLI

列出图片食谱

以下 `list-image-recipes` 示例列出了您的所有图片配方。

```
aws imagebuilder list-image-recipes
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "imageRecipeSummaryList": [
    {
      "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/mybasicrecipe/2019.12.03",
      "name": "MyBasicRecipe",
      "platform": "Windows",
      "owner": "123456789012",
      "parentImage": "arn:aws:imagebuilder:us-west-2:aws:image/windows-server-2016-english-full-base-x86/2019.x.x",
      "dateCreated": "2020-02-19T18:54:25.975Z",
      "tags": {
        "KeyName": "KeyValue"
      }
    },
    {
```

```
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/recipe-
name-a1b2c3d45678/1.0.0",
    "name": "recipe-name-a1b2c3d45678",
    "platform": "Linux",
    "owner": "123456789012",
    "parentImage": "arn:aws:imagebuilder:us-west-2:aws:image/amazon-linux-2-
x86/2019.11.21",
    "dateCreated": "2019-12-16T18:19:00.120Z",
    "tags": {
      "KeyName": "KeyValue"
    }
  }
]
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListImageRecipes](#) 中的。

## list-images

以下代码示例演示了如何使用 list-images。

### AWS CLI

列出图片

以下 list-images 示例列出了您有权访问的所有语义版本。

```
aws imagebuilder list-images
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "imageVersionList": [
    {
      "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/
mybasicrecipe/2019.12.03",
      "name": "MyBasicRecipe",
      "version": "2019.12.03",
```

```
        "platform": "Windows",
        "owner": "123456789012",
        "dateCreated": "2020-02-14T21:29:18.810Z"
    }
]
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListImages](#) 中的。

## list-infrastructure-configurations

以下代码示例演示了如何使用 `list-infrastructure-configurations`。

### AWS CLI

#### 列出基础架构配置

以下 `list-infrastructure-configurations` 示例列出了您的所有基础设施配置。

```
aws imagebuilder list-infrastructure-configurations
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "infrastructureConfigurationSummaryList": [
    {
      "arn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/myexampleinfrastructure",
      "name": "MyExampleInfrastructure",
      "description": "An example that will retain instances of failed builds",
      "dateCreated": "2020-02-19T19:11:51.858Z",
      "tags": {}
    },
    {
      "arn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/infrastructureconfiguration-name-a1b2c3d45678",
      "name": "infrastructureConfiguration-name-a1b2c3d45678",
      "dateCreated": "2019-12-16T18:19:01.038Z",
```

```
        "tags": {
            "KeyName": "KeyValue"
        }
    ]
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListInfrastructureConfigurations](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出特定资源的标签

以下`list-tags-for-resource`示例列出了特定资源的所有标签。

```
aws imagebuilder list-tags-for-resource \
  --resource-arn arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/
  mywindows2016pipeline
```

输出：

```
{
  "tags": {
    "KeyName": "KeyValue"
  }
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## put-component-policy

以下代码示例演示了如何使用 `put-component-policy`。

## AWS CLI

### 将资源策略应用于组件

以下 `put-component-policy` 命令将资源策略应用于构建组件，以实现构建组件的跨账户共享。我们建议您使用 RAM CLI 命令 `create-resource-share`。如果您使用 EC2 Image Builder CLI 命令 `put-component-policy`，则还必须使用 RAM CLI 命令 `promote-resource-share-create-from-policy` 才能使与之共享资源的所有委托人都能看到该资源。

```
aws imagebuilder put-component-policy \
  --component-arn arn:aws:imagebuilder:us-west-2:123456789012:component/
  examplecomponent/2019.12.02/1 \
  --policy '{ "Version": "2012-10-17", "Statement": [ { "Effect":
  "Allow", "Principal": { "AWS": [ "123456789012" ] }, "Action":
  [ "imagebuilder:GetComponent", "imagebuilder:ListComponents" ],
  "Resource": [ "arn:aws:imagebuilder:us-west-2:123456789012:component/
  examplecomponent/2019.12.02/1" ] } ] }'
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "componentArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/
  examplecomponent/2019.12.02/1"
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutComponentPolicy](#) 中的。

## **put-image-policy**

以下代码示例演示了如何使用 `put-image-policy`。

## AWS CLI

### 将资源策略应用于图像

以下 `put-image-policy` 命令将资源策略应用于图像以实现跨账户共享图像。我们建议您使用 RAM CLI 命令 `create-resource-share`。如果您使用 EC2 Image Builder CLI 命令 `put-image-`

policy，则还必须使用 RAM CLI 命令 `promote-resource-share-create-from-policy`，以便与之共享资源的所有委托人都能看到该资源。

```
aws imagebuilder put-image-policy \  
  --image-arn arn:aws:imagebuilder:us-west-2:123456789012:image/example-  
image/2019.12.02/1 \  
  --policy '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow",  
"Principal": { "AWS": [ "123456789012" ] }, "Action": [ "imagebuilder:GetImage",  
"imagebuilder:ListImages" ], "Resource": [ "arn:aws:imagebuilder:us-  
west-2:123456789012:image/example-image/2019.12.02/1" ] } ] }'
```

输出：

```
{  
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
  "imageArn": "arn:aws:imagebuilder:us-west-2:123456789012:image/example-  
image/2019.12.02/1"  
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutImagePolicy](#) 中的。

## put-image-recipe-policy

以下代码示例演示了如何使用 `put-image-recipe-policy`。

### AWS CLI

将资源策略应用于图像配方

以下 `put-image-recipe-policy` 命令将资源策略应用于图像配方，以启用图像配方的跨账户共享。我们建议您使用 RAM CLI 命令 `create-resource-share`。如果您使用 EC2 Image Builder CLI 命令 `put-image-recipe-policy`，则还必须使用 RAM CLI 命令 `promote-resource-share-create-from-policy` 才能使与之共享资源的所有委托人都能看到该资源。

```
aws imagebuilder put-image-recipe-policy \  
  --image-recipe-arn arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/  
example-image-recipe/2019.12.02 \  
  --policy '{ "Version": "2012-10-17", "Statement": [ { "Effect":  
"Allow", "Principal": { "AWS": [ "123456789012" ] }, "Action":
```

```
[ "imagebuilder:GetImageRecipe", "imagebuilder:ListImageRecipes" ], "Resource":  
[ "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/example-image-  
recipe/2019.12.02" ] ] ] }'
```

输出：

```
{  
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
  "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/  
example-image-recipe/2019.12.02/1"  
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutImageRecipePolicy](#) 中的。

## start-image-pipeline-execution

以下代码示例演示了如何使用 start-image-pipeline-execution。

### AWS CLI

#### 手动启动图像管道

以下 start-image-pipeline-execution 示例手动启动图像管道。

```
aws imagebuilder start-image-pipeline-execution \  
  --image-pipeline-arn arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/  
mywindows2016pipeline
```

输出：

```
{  
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
  "clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
  "imageBuildVersionArn": "arn:aws:imagebuilder:us-west-2:123456789012:image/  
mybasicrecipe/2019.12.03/1"  
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartImagePipelineExecution](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源添加标签

以下tag-resource示例使用 JSON 文件向 EC2 Image Builder 添加资源并将其标记为。

```
aws imagebuilder tag-resource \  
  --cli-input-json file://tag-resource.json
```

tag-resource.json 的内容：

```
{  
  "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/  
mywindows2016pipeline",  
  "tags": {  
    "KeyName": "KeyValue"  
  }  
}
```

此命令不生成任何输出。

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从资源中移除标签

以下untag-resource示例使用 JSON 文件从资源中删除标签。

```
aws imagebuilder untag-resource \  
  --cli-input-json file://tag-resource.json
```

untag-resource.json 的内容：

```
{  
  "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/  
mywindows2016pipeline",  
  "tagKeys": [  
    "KeyName"  
  ]  
}
```

此命令不生成任何输出。

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-distribution-configuration

以下代码示例演示了如何使用 update-distribution-configuration。

### AWS CLI

#### 更新分发配置

以下 update-distribution-configuration 示例使用 JSON 文件更新分发配置。

```
aws imagebuilder update-distribution-configuration \  
  --cli-input-json file://update-distribution-configuration.json
```

update-distribution-configuration.json 的内容：

```
{  
  "distributionConfigurationArn": "arn:aws:imagebuilder:us-  
west-2:123456789012:distribution-configuration/myexampledistribution",  
  "description": "Copies AMI to eu-west-2 and exports to S3",  
  "distributions": [  
    {  
      "region": "us-west-2",
```

```

        "amiDistributionConfiguration": {
            "name": "Name {{imagebuilder:buildDate}}",
            "description": "An example image name with parameter references"
        }
    },
    {
        "region": "eu-west-2",
        "amiDistributionConfiguration": {
            "name": "My {{imagebuilder:buildVersion}} image
{{imagebuilder:buildDate}}"
        }
    }
]
}

```

输出：

```

{
    "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}

```

有关更多信息，请参阅 [《EC2 Image Builder 用户指南》](#) 中的“使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateDistributionConfiguration](#) 中的。

## update-image-pipeline

以下代码示例演示了如何使用 update-image-pipeline。

### AWS CLI

#### 更新图像管道

以下 update-image-pipeline 示例使用 JSON 文件更新图像管道。

```

aws imagebuilder update-image-pipeline \
    --cli-input-json file://update-image-pipeline.json

```

update-image-pipeline.json 的内容：

```

{

```

```
"imagePipelineArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/mywindows2016pipeline",
  "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/mybasicrecipe/2019.12.03",
  "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/myexampleinfrastructure",
  "distributionConfigurationArn": "arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/myexampledistribution",
  "imageTestsConfiguration": {
    "imageTestsEnabled": true,
    "timeoutMinutes": 120
  },
  "schedule": {
    "scheduleExpression": "cron(0 0 * * MON)",
    "pipelineExecutionStartCondition":
"EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
  },
  "status": "DISABLED"
}
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateImagePipeline](#) 中的。

## update-infrastructure-configuration

以下代码示例演示了如何使用 update-infrastructure-configuration。

### AWS CLI

#### 更新基础架构配置

以下 update-infrastructure-configuration 示例使用 JSON 文件更新基础设施配置。

```
aws imagebuilder update-infrastructure-configuration \
```

```
--cli-input-json file:/update-infrastructure-configuration.json
```

update-infrastructure-configuration.json 的内容：

```
{
  "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/myexampleinfrastructure",
  "description": "An example that will terminate instances of failed builds",
  "instanceTypes": [
    "m5.large", "m5.2xlarge"
  ],
  "instanceProfileName": "EC2InstanceProfileForImageFactory",
  "securityGroupIds": [
    "sg-a48c95ef"
  ],
  "subnetId": "subnet-a48c95ef",
  "logging": {
    "s3Logs": {
      "s3BucketName": "bucket-name",
      "s3KeyPrefix": "bucket-path"
    }
  },
  "terminateInstanceOnFailure": true,
  "snsTopicArn": "arn:aws:sns:us-west-2:123456789012:sns-name"
}
```

输出：

```
{
  "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

有关更多信息，请参阅《[EC2 Image Builder 用户指南](#)》中的“[使用 AWS CLI 设置和管理 EC2 Image Builder 映像管道](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateInfrastructureConfiguration](#) 中的。

## 使用的事件管理器示例 AWS CLI

以下代码示例向您展示了如何通过使用事件管理器来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## **create-replication-set**

以下代码示例演示了如何使用 `create-replication-set`。

AWS CLI

### 创建复制集

以下 `create-replication-set` 示例创建了事件管理器用来复制和加密您的 Amazon Web Services 账户中的数据的复制集。此示例在创建复制集时使用 `us-east-1` 和 `us-east-2` 区域。

```
aws ssm-incidents create-replication-set \  
  --regions '{"us-east-1": {"sseKmsKeyId": "arn:aws:kms:us-  
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"}, "us-east-2":  
  {"sseKmsKeyId": "arn:aws:kms:us-  
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"}}'
```

输出：

```
{  
  "replicationSetArns": [  
    "arn:aws:ssm-incidents::111122223333:replication-set/c4bcb603-4bf9-  
bb3f-413c-08df53673b57"  
  ]  
}
```

有关更多信息，请参阅[事件管理器用户指南中的使用事件管理器复制集](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateReplicationSet](#)中的。

## create-response-plan

以下代码示例演示了如何使用 create-response-plan。

### AWS CLI

#### 创建响应计划

以下create-response-plan示例创建了包含指定详细信息的响应计划。

```
aws ssm-incidents create-response-plan \
  --chat-channel '{"chatbotSns": ["arn:aws:sns:us-east-1:111122223333:Standard_User"]}' \
  --display-name "Example response plan" \
  --incident-template '{"impact": 5, "title": "example-incident"}' \
  --name "example-response" \
  --actions '[{"ssmAutomation": {"documentName": "AWSIncidents-CriticalIncidentRunbookTemplate", "documentVersion": "$DEFAULT", "roleArn": "arn:aws:iam::111122223333:role/aws-service-role/ssm-incidents.amazonaws.com/AWSServiceRoleForIncidentManager", "targetAccount": "RESPONSE_PLAN_OWNER_ACCOUNT"}}]' \
  --engagements '["arn:aws:ssm-contacts:us-east-1:111122223333:contact/example"]'
```

输出：

```
{
  "arn": "arn:aws:ssm-incidents::111122223333:response-plan/example-response"
}
```

有关更多信息，请参阅《[事件管理器用户指南](#)》中的事件准备。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateResponsePlan](#)中的。

## create-timeline-event

以下代码示例演示了如何使用 create-timeline-event。

### AWS CLI

#### 示例 1：创建自定义时间轴事件

以下`create-timeline-event`示例在指定时间为指定事件创建自定义时间轴事件。

```
aws ssm-incidents create-timeline-event \  
  --event-data "\"example timeline event\"" \  
  --event-time 2022-10-01T20:30:00.000 \  
  --event-type "Custom Event" \  
  --incident-record-arn "arn:aws:ssm-incidents::111122223333:incident-record/  
Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4EXAMPLE"
```

输出：

```
{  
  "eventId": "c0bcc885-a41d-eb01-b4ab-9d2deEXAMPLE",  
  "incidentRecordArn": "arn:aws:ssm-incidents::111122223333:incident-record/  
Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4EXAMPLE"  
}
```

示例 2：创建带有事件备注的时间轴事件

以下`create-timeline-event`示例创建了一个在“事件备注”面板中列出的时间轴事件。

```
aws ssm-incidents create-timeline-event \  
  --event-data "\"New Note\"" \  
  --event-type "Note" \  
  --incident-record-arn "arn:aws:ssm-incidents::111122223333:incident-record/  
Test/6cc46130-ca6c-3b38-68f1-f6abeEXAMPLE" \  
  --event-time 2023-06-20T12:06:00.000 \  
  --event-references ' [{"resource": "arn:aws:ssm-incidents::111122223333:incident-  
record/Test/6cc46130-ca6c-3b38-68f1-f6abeEXAMPLE"} ]'
```

输出：

```
{  
  "eventId": "a41dc885-c0bc-b4ab-eb01-de9d2EXAMPLE",  
  "incidentRecordArn": "arn:aws:ssm-incidents::111122223333:incident-record/  
Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4EXAMPLE"  
}
```

有关更多信息，请参阅[事件管理器用户指南中的事件详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTimelineEvent](#)中的。



## delete-incident-record

以下代码示例演示了如何使用 delete-incident-record。

### AWS CLI

#### 删除事件记录

以下delete-incident-record示例删除了指定的事件记录。

```
aws ssm-incidents delete-incident-record \  
  --arn "arn:aws:ssm-incidents::111122223333:incident-record/Example-Response-  
  Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"
```

此命令不生成任何输出。

有关更多信息，请参阅[事件管理器用户指南中的事件跟踪](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteIncidentRecord](#)中的。

## delete-replication-set

以下代码示例演示了如何使用 delete-replication-set。

### AWS CLI

#### 删除复制集

以下delete-replication-set示例从您的亚马逊 Web Services 账户中删除复制集。删除复制集还会删除所有事件管理器数据。此操作无法撤消。

```
aws ssm-incidents delete-replication-set \  
  --arn "arn:aws:ssm-incidents::111122223333:replication-set/c4bcb603-4bf9-  
  bb3f-413c-08df53673b57"
```

此命令不生成任何输出。

有关更多信息，请参阅[事件管理器用户指南中的使用事件管理器复制集](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteReplicationSet](#)中的。

## delete-resource-policy

以下代码示例演示了如何使用 delete-resource-policy。

### AWS CLI

#### 删除资源策略

以下delete-resource-policy示例从响应计划中删除资源策略。这将撤消与之共享响应计划的委托人或组织的访问权限。

```
aws ssm-incidents delete-resource-policy \  
  --policy-id "be8b57191f0371f1c6827341aa3f0a03" \  
  --resource-arn "arn:aws:ssm-incidents::111122223333:response-plan/Example-Response-Plan"
```

此命令不生成任何输出。

有关更多信息，[请参阅事件管理器用户指南中的使用共享联系人和响应计划。](#)

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteResourcePolicy](#)中的。

## delete-response-plan

以下代码示例演示了如何使用 delete-response-plan。

### AWS CLI

#### 删除响应计划

以下delete-response-plan示例删除了指定的响应计划。

```
aws ssm-incidents delete-response-plan \  
  --arn "arn:aws:ssm-incidents::111122223333:response-plan/example-response"
```

此命令不生成任何输出。

有关更多信息，请参阅《[事件管理器用户指南](#)》中的[事件准备](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteResponsePlan](#)中的。

## delete-timeline-event

以下代码示例演示了如何使用 delete-timeline-event。

### AWS CLI

#### 删除时间轴事件

以下delete-timeline-event示例从指定的事件记录中删除自定义时间轴事件。

```
aws ssm-incidents delete-timeline-event \  
  --event-id "c0bcc885-a41d-eb01-b4ab-9d2de193643c" \  
  --incident-record-arn "arn:aws:ssm-incidents::111122223333:incident-record/  
Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"
```

此命令不生成任何输出。

有关更多信息，请参阅[事件管理器用户指南中的事件详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTimelineEvent](#)中的。

## get-incident-record

以下代码示例演示了如何使用 get-incident-record。

### AWS CLI

#### 要获取事件记录

以下get-incident-record示例获取有关指定事件记录的详细信息。

```
aws ssm-incidents get-incident-record \  
  --arn "arn:aws:ssm-incidents::111122223333:incident-record/Example-Response-  
Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"
```

输出：

```
{  
  "incidentRecord": {  
    "arn": "arn:aws:ssm-incidents::111122223333:incident-record/Example-  
Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308",  
    "automationExecutions": [],
```

```

    "creationTime": "2021-05-21T18:16:57.579000+00:00",
    "dedupeString": "c4bcc812-85e7-938d-2b78-17181176ee1a",
    "impact": 5,
    "incidentRecordSource": {
      "createdBy": "arn:aws:iam::111122223333:user/drალიატპ",
      "invokedBy": "arn:aws:iam::111122223333:user/drალიატპ",
      "source": "aws.ssm-incidents.custom"
    },
    "lastModifiedBy": "arn:aws:iam::111122223333:user/drალიატპ",
    "lastModifiedTime": "2021-05-21T18:16:59.149000+00:00",
    "notificationTargets": [],
    "status": "OPEN",
    "title": "Example-Incident"
  }
}

```

有关更多信息，请参阅[事件管理器用户指南中的事件详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetIncidentRecord](#)中的。

## get-replication-set

以下代码示例演示了如何使用 get-replication-set。

### AWS CLI

#### 获取复制集

以下get-replication-set示例详细介绍了事件管理器用于复制和加密您的 Amazon Web Services 账户中的数据的数据的复制集。

```

aws ssm-incidents get-replication-set \
  --arn "arn:aws:ssm-incidents::111122223333:replication-set/c4bcb603-4bf9-
bb3f-413c-08df53673b57"

```

输出：

```

{
  "replicationSet": {
    "createdBy": "arn:aws:sts::111122223333:assumed-role/Admin/username",
    "createdTime": "2021-05-14T17:57:22.010000+00:00",
    "deletionProtected": false,

```

```

    "lastModifiedBy": "arn:aws:sts::111122223333:assumed-role/Admin/username",
    "lastModifiedTime": "2021-05-14T17:57:22.010000+00:00",
    "regionMap": {
      "us-east-1": {
        "sseKmsKeyId": "DefaultKey",
        "status": "ACTIVE"
      },
      "us-east-2": {
        "sseKmsKeyId": "DefaultKey",
        "status": "ACTIVE",
        "statusMessage": "Tagging inaccessible"
      }
    },
    "status": "ACTIVE"
  }
}

```

有关更多信息，请参阅[事件管理器用户指南中的使用事件管理器复制集](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetReplicationSet](#)中的。

## get-resource-policies

以下代码示例演示了如何使用 get-resource-policies。

### AWS CLI

列出响应计划的资源策略

以下command-name示例列出了与指定响应计划关联的资源策略。

```

aws ssm-incidents get-resource-policies \
--resource-arn "arn:aws:ssm-incidents::111122223333:response-plan/Example-Response-Plan"

```

输出：

```

{
  "resourcePolicies": [
    {
      "policyDocument": "{\"Version\":\"2012-10-17\",\"Statement\": [{\"Sid\": \"d901b37a-dbb0-458a-8842-75575c464219-external-principals\", \"Effect\": \"Allow\", \"Principal\": {\"AWS\": \"arn:aws:iam::222233334444:root\"}, \"Action

```

```

\":[\"ssm-incidents:GetResponsePlan\", \"ssm-incidents:StartIncident\", \"ssm-
incidents:UpdateIncidentRecord\", \"ssm-incidents:GetIncidentRecord\", \"ssm-
incidents:CreateTimelineEvent\", \"ssm-incidents:UpdateTimelineEvent\", \"ssm-
incidents:GetTimelineEvent\", \"ssm-incidents:ListTimelineEvents\", \"ssm-
incidents:UpdateRelatedItems\", \"ssm-incidents:ListRelatedItems\"], \"Resource\":
[\"arn:aws:ssm-incidents:*:111122223333:response-plan/Example-Response-Plan\",
\"arn:aws:ssm-incidents:*:111122223333:incident-record/Example-Response-Plan/*
\"]}]\"],
    \"policyId\": \"be8b57191f0371f1c6827341aa3f0a03\",
    \"ramResourceShareRegion\": \"us-east-1\"
  }
]
}

```

有关更多信息，[请参阅事件管理器用户指南中的使用共享联系人和响应计划](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetResourcePolicies](#)中的。

## get-response-plan

以下代码示例演示了如何使用 get-response-plan。

### AWS CLI

获取响应计划的详细信息

以下command-name示例获取有关您 AWS 账户中指定响应计划的详细信息。

```

aws ssm-incidents get-response-plan \
  --arn "arn:aws:ssm-incidents::111122223333:response-plan/Example-Response-Plan"

```

输出：

```

{
  "actions": [
    {
      "ssmAutomation": {
        "documentName": "AWSIncidents-CriticalIncidentRunbookTemplate",
        "documentVersion": "$DEFAULT",
        "roleArn": "arn:aws:iam::111122223333:role/aws-service-role/ssm-
incidents.amazonaws.com/AWSServiceRoleForIncidentManager",
        "targetAccount": "RESPONSE_PLAN_OWNER_ACCOUNT"
      }
    }
  ]
}

```

```

    }
  ],
  "arn": "arn:aws:ssm-incidents::111122223333:response-plan/Example-Response-Plan",
  "chatChannel": {
    "chatbotSns": [
      "arn:aws:sns:us-east-1:111122223333:Standard_User"
    ]
  },
  "displayName": "Example response plan",
  "engagements": [
    "arn:aws:ssm-contacts:us-east-1:111122223333:contact/example"
  ],
  "incidentTemplate": {
    "impact": 5,
    "title": "Example-Incident"
  },
  "name": "Example-Response-Plan"
}

```

有关更多信息，请参阅 [《事件管理器用户指南》中的事件准备](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetResponsePlan](#) 中的。

## get-timeline-event

以下代码示例演示了如何使用 `get-timeline-event`。

### AWS CLI

获取时间轴事件的详细信息

以下 `get-timeline-event` 示例返回指定时间轴事件的详细信息。

```

aws ssm-incidents get-timeline-event \
  --event-id 20bcc812-8a94-4cd7-520c-0ff742111424 \
  --incident-record-arn "arn:aws:ssm-incidents::111122223333:incident-record/Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"

```

输出：

```

{
  "event": {

```

```

    "eventData": "\"Incident Started\"",
    "eventId": "20bcc812-8a94-4cd7-520c-0ff742111424",
    "eventTime": "2021-05-21T18:16:57+00:00",
    "eventType": "Custom Event",
    "eventUpdatedTime": "2021-05-21T18:16:59.944000+00:00",
    "incidentRecordArn": "arn:aws:ssm-incidents::111122223333:incident-record/
Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"
  }
}

```

有关更多信息，请参阅[事件管理器用户指南中的事件详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTimelineEvent](#)中的。

## list-incident-records

以下代码示例演示了如何使用 list-incident-records。

### AWS CLI

#### 列出事件记录

以下command-name示例列出了您的亚马逊 Web Services 账户中的事件记录。

```
aws ssm-incidents list-incident-records
```

输出：

```

{
  "incidentRecordSummaries": [
    {
      "arn": "arn:aws:ssm-incidents::111122223333:incident-record/Example-
Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308",
      "creationTime": "2021-05-21T18:16:57.579000+00:00",
      "impact": 5,
      "incidentRecordSource": {
        "createdBy": "arn:aws:iam::111122223333:user/draliatp",
        "invokedBy": "arn:aws:iam::111122223333:user/draliatp",
        "source": "aws.ssm-incidents.custom"
      },
      "status": "OPEN",
      "title": "Example-Incident"
    }
  ]
}

```



```
]
}
```

有关更多信息，请参阅[事件管理器用户指南中的事件列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListIncidentRecords](#)中的。

## list-related-items

以下代码示例演示了如何使用 list-related-items。

### AWS CLI

列出相关商品

以下list-related-items示例列出了指定事件的相关项目。

```
aws ssm-incidents list-related-items \
  --incident-record-arn "arn:aws:ssm-incidents::111122223333:incident-record/
  Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"
```

输出：

```
{
  "relatedItems": [
    {
      "identifier": {
        "type": "OTHER",
        "value": {
          "url": "https://console.aws.amazon.com/systems-manager/opsitems/
oi-8ef82158e190/workbench?region=us-east-1"
        }
      },
      "title": "Example related item"
    },
    {
      "identifier": {
        "type": "PARENT",
        "value": {
          "arn": "arn:aws:ssm:us-east-1:111122223333:opsitem/
oi-8084126392ac"
        }
      },
    },
  ]
}
```

```
        "title": "parentItem"
      }
    ]
  }
```

有关更多信息，请参阅[事件管理器用户指南中的事件详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRelatedItems](#)中的。

## list-replication-sets

以下代码示例演示了如何使用 list-replication-sets。

### AWS CLI

#### 列出复制集

以下list-replication-set示例列出了 Incident Manager 用来复制和加密您 AWS 账户中的数据的复制集。

```
aws ssm-incidents list-replication-sets
```

输出：

```
{
  "replicationSetArns": [
    "arn:aws:ssm-incidents::111122223333:replication-set/c4bcb603-4bf9-
    bb3f-413c-08df53673b57"
  ]
}
```

有关更多信息，请参阅[事件管理器用户指南中的使用事件管理器复制集](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListReplicationSets](#)中的。

## list-response-plans

以下代码示例演示了如何使用 list-response-plans。

### AWS CLI

#### 列出可用的响应计划

以下 `list-response-plans` 示例列出了您的亚马逊 Web Services 账户中可用的响应计划。

```
aws ssm-incidents list-response-plans
```

输出：

```
{
  "responsePlanSummaries": [
    {
      "arn": "arn:aws:ssm-incidents::111122223333:response-plan/Example-Response-Plan",
      "displayName": "Example response plan",
      "name": "Example-Response-Plan"
    }
  ]
}
```

有关更多信息，请参阅 [《事件管理器用户指南》中的事件准备](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListResponsePlans](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

AWS CLI

列出响应计划的标签

以下 `list-tags-for-resource` 示例列出了与指定响应计划关联的标签。

```
aws ssm-incidents list-tags-for-resource \
  --resource-arn "arn:aws:ssm-incidents::111122223333:response-plan/Example-Response-Plan"
```

输出：

```
{
  "tags": {
    "group1": "1"
  }
}
```

```
}
```

有关更多信息，请参阅事件管理器用户指南中的[标记](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## list-timeline-events

以下代码示例演示了如何使用 `list-timeline-events`。

### AWS CLI

列出事件的时间表事件

以下 `command-name` 示例列出了指定事件的时间表事件。

```
aws ssm-incidents list-timeline-events \  
  --incident-record-arn "arn:aws:ssm-incidents::111122223333:incident-record/  
Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"
```

输出：

```
{  
  "eventSummaries": [  
    {  
      "eventId": "8cbcc889-35e1-a42d-2429-d6f100799915",  
      "eventTime": "2021-05-21T22:36:13.766000+00:00",  
      "eventType": "SSM Incident Record Update",  
      "eventUpdatedTime": "2021-05-21T22:36:13.766000+00:00",  
      "incidentRecordArn": "arn:aws:ssm-incidents::111122223333:incident-  
record/Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"  
    },  
    {  
      "eventId": "a2bcc825-aab5-1787-c605-f9bb2640d85b",  
      "eventTime": "2021-05-21T18:58:46.443000+00:00",  
      "eventType": "SSM Incident Record Update",  
      "eventUpdatedTime": "2021-05-21T18:58:46.443000+00:00",  
      "incidentRecordArn": "arn:aws:ssm-incidents::111122223333:incident-  
record/Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"  
    },  
    {  
      "eventId": "5abcc812-89c0-b0a8-9437-1c74223d4685",  
      "eventTime": "2021-05-21T18:16:59.149000+00:00",  
      "eventType": "SSM Incident Record Update",  
      "eventUpdatedTime": "2021-05-21T18:16:59.149000+00:00",  
      "incidentRecordArn": "arn:aws:ssm-incidents::111122223333:incident-  
record/Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"  
    }  
  ]  
}
```

```

        "eventType": "SSM Incident Record Update",
        "eventUpdatedTime": "2021-05-21T18:16:59.149000+00:00",
        "incidentRecordArn": "arn:aws:ssm-incidents::111122223333:incident-
record/Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"
    },
    {
        "eventId": "06bcc812-8820-405e-4065-8d2b14d29b92",
        "eventTime": "2021-05-21T18:16:58+00:00",
        "eventType": "SSM Automation Execution Start Failure for Incident",
        "eventUpdatedTime": "2021-05-21T18:16:58.689000+00:00",
        "incidentRecordArn": "arn:aws:ssm-incidents::111122223333:incident-
record/Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"
    },
    {
        "eventId": "20bcc812-8a94-4cd7-520c-0ff742111424",
        "eventTime": "2021-05-21T18:16:57+00:00",
        "eventType": "Custom Event",
        "eventUpdatedTime": "2021-05-21T18:16:59.944000+00:00",
        "incidentRecordArn": "arn:aws:ssm-incidents::111122223333:incident-
record/Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"
    },
    {
        "eventId": "c0bcc885-a41d-eb01-b4ab-9d2de193643c",
        "eventTime": "2020-10-01T20:30:00+00:00",
        "eventType": "Custom Event",
        "eventUpdatedTime": "2021-05-21T22:28:26.299000+00:00",
        "incidentRecordArn": "arn:aws:ssm-incidents::111122223333:incident-
record/Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"
    }
]
}

```

有关更多信息，请参阅[事件管理器用户指南中的事件详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTimelineEvents](#)中的。

## put-resource-policy

以下代码示例演示了如何使用 put-resource-policy。

### AWS CLI

#### 分享响应计划和事件

以下command-name示例在 Example-Response-Plan 中添加了一个资源策略，该策略与指定的委托人共享响应计划和关联的事件。

```
aws ssm-incidents put-resource-policy \  
  --resource-arn "arn:aws:ssm-incidents::111122223333:response-plan/Example-Response-Plan" \  
  --policy "{\"Version\":\"2012-10-17\",\"Statement\": [{\"Sid\":  
  \"ExampleResourcePolciy\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":  
  \"arn:aws:iam::222233334444:root\"},\"Action\": [\"ssm-incidents:GetResponsePlan  
  \",\"ssm-incidents:StartIncident\",\"ssm-incidents:UpdateIncidentRecord  
  \",\"ssm-incidents:GetIncidentRecord\",\"ssm-incidents:CreateTimelineEvent  
  \",\"ssm-incidents:UpdateTimelineEvent\",\"ssm-incidents:GetTimelineEvent  
  \",\"ssm-incidents:ListTimelineEvents\",\"ssm-incidents:UpdateRelatedItems  
  \",\"ssm-incidents:ListRelatedItems\"]},\"Resource\": [\"arn:aws:ssm-  
  incidents:*:111122223333:response-plan/Example-Response-Plan\", \"arn:aws:ssm-  
  incidents:*:111122223333:incident-record/Example-Response-Plan/*\"]}]}"
```

输出：

```
{  
  "policyId": "be8b57191f0371f1c6827341aa3f0a03"  
}
```

有关更多信息，[请参阅事件管理器用户指南中的使用共享联系人和响应计划](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutResourcePolicy](#)中的。

## start-incident

以下代码示例演示了如何使用 start-incident。

### AWS CLI

开始事件

以下start-incident示例使用指定的响应计划启动事件。

```
aws ssm-incidents start-incident \  
  --response-plan-arn "arn:aws:ssm-incidents::111122223333:response-plan/Example-Response-Plan"
```

输出：

```
{
  "incidentRecordArn": "arn:aws:ssm-incidents::682428703967:incident-record/
Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308"
}
```

有关更多信息，请参阅[事件管理器用户指南中的事件创建](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartIncident](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为响应计划添加标签

以下tag-resource示例使用提供的标签键值对标记了指定的响应计划。

```
aws ssm-incidents tag-resource \
  --resource-arn "arn:aws:ssm-incidents::111122223333:response-plan/Example-
Response-Plan" \
  --tags '{"group1":"1"}
```

此命令不生成任何输出。

有关更多信息，请参阅事件管理器用户指南中的[标记](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从响应计划中移除标签

以下untag-resource示例从响应计划中删除指定的标签。

```
aws ssm-incidents untag-resource \
  --resource-arn "arn:aws:ssm-incidents::111122223333:response-plan/Example-
Response-Plan" \
```

```
--tag-keys '["group1"]'
```

此命令不生成任何输出。

有关更多信息，请参阅事件管理器用户指南中的[标记](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-deletion-protection

以下代码示例演示了如何使用 update-deletion-protection。

### AWS CLI

要更新复制，请设置删除保护

以下update-deletion-protection示例更新了您账户中的删除保护，以防止您删除复制集中的最后一个区域。

```
aws ssm-incidents update-deletion-protection \  
  --arn "arn:aws:ssm-incidents::111122223333:replication-set/  
a2bcc5c9-0f53-8047-7fef-c20749989b40" \  
  --deletion-protected
```

此命令不生成任何输出。

有关更多信息，请参阅[事件管理器用户指南中的使用事件管理器复制集](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDeletionProtection](#)中的。

## update-incident-record

以下代码示例演示了如何使用 update-incident-record。

### AWS CLI

更新事件记录

以下command-name示例解决了指定的事件。

```
aws ssm-incidents update-incident-record \  
  --arn "arn:aws:ssm-incidents::111122223333:incident-record/Example-Response-  
Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308" \  
  --command-name
```



```
--status "RESOLVED"
```

此命令不生成任何输出。

有关更多信息，请参阅[事件管理器用户指南中的事件详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateIncidentRecord](#)中的。

## update-related-items

以下代码示例演示了如何使用 update-related-items。

### AWS CLI

更新与事件相关的项目

以下update-related-item示例从指定的事件记录中移除相关项目。

```
aws ssm-incidents update-related-items \  
  --incident-record-arn "arn:aws:ssm-incidents::111122223333:incident-record/  
Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308" \  
  --related-items-update '{"itemToRemove": {"type": "OTHER", "value": {"url":  
"https://console.aws.amazon.com/systems-manager/opsitems/oi-8ef82158e190/workbench?  
region=us-east-1"}}}'
```

此命令不生成任何输出。

有关更多信息，请参阅[事件管理器用户指南中的事件详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRelatedItems](#)中的。

## update-replication-set

以下代码示例演示了如何使用 update-replication-set。

### AWS CLI

更新复制集

以下command-name示例从复制集中删除 us-east-2 区域。

```
aws ssm-incidents update-replication-set \  
  --arn "arn:aws:ssm-incidents::111122223333:replication-set/  
a2bcc5c9-0f53-8047-7fef-c20749989b40" \  
  --command-name "command-name"
```

```
--actions '[{"deleteRegionAction": {"regionName": "us-east-2"}}]'
```

此命令不生成任何输出。

有关更多信息，请参阅[事件管理器用户指南中的使用事件管理器复制集](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateReplicationSet](#)中的。

## update-response-plan

以下代码示例演示了如何使用 update-response-plan。

### AWS CLI

#### 更新响应计划

以下update-response-plan示例将聊天频道从指定的响应计划中移除。

```
aws ssm-incidents update-response-plan \  
  --arn "arn:aws:ssm-incidents::111122223333:response-plan/Example-Response-Plan" \  
  \  
  --chat-channel '{"empty":{}}'
```

此命令不生成任何输出。

有关更多信息，请参阅《[事件管理器用户指南](#)》中的[事件准备](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateResponsePlan](#)中的。

## update-timeline-event

以下代码示例演示了如何使用 update-timeline-event。

### AWS CLI

#### 更新时间轴事件

以下update-timeline-event示例更新了事件发生的时间。

```
aws ssm-incidents update-timeline-event \  
  --event-id 20bcc812-8a94-4cd7-520c-0ff742111424 \  
  --incident-record-arn "arn:aws:ssm-incidents::111122223333:incident-record/  
Example-Response-Plan/6ebcc812-85f5-b7eb-8b2f-283e4d844308" \  
  --start-time 2017-01-01T00:00:00.000Z
```

```
--event-time "2021-05-21T18:10:57+00:00"
```

此命令不生成任何输出。

有关更多信息，请参阅[事件管理器用户指南中的事件详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateTimelineEvent](#)中的。

## 使用的事件管理器联系人示例 AWS CLI

以下代码示例向您展示了如何使用事件管理器联系人来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### accept-page

以下代码示例演示了如何使用 accept-page。

AWS CLI

在参与期间接受页面和互动

以下accept-page示例使用发送到联系渠道的接受代码来接受页面。

```
aws ssm-contacts accept-page \  
  --page-id "arn:aws:ssm-contacts:us-east-2:682428703967:page/  
akuam/94ea0c7b-56d9-46c3-b84a-a37c8b067ad3" \  
  --accept-type READ \  
  --accept-code 425440
```

此命令不产生任何输出

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AcceptPage](#)中的。

## activate-contact-channel

以下代码示例演示了如何使用 activate-contact-channel。

### AWS CLI

激活联系人的联系渠道

以下activate-contact-channel示例激活联系渠道并将其作为事件的一部分使用。

```
aws ssm-contacts activate-contact-channel \  
  --contact-channel-id "arn:aws:ssm-contacts:us-east-2:111122223333:contact-  
channel/akuam/fc7405c4-46b2-48b7-87b2-93e2f225b90d" \  
  --activation-code "466136"
```

此命令不生成任何输出。

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ActivateContactChannel](#)中的。

## command-name

以下代码示例演示了如何使用 command-name。

### AWS CLI

删除联系人

以下command-name示例删除联系人。任何提及该联系人的升级计划都将无法再联系到该联系人。

```
aws ssm-contacts delete-contact \  
  --contact-id "arn:aws:ssm-contacts:us-east-1:682428703967:contact/alejrr"
```

此命令不生成任何输出。

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CommandName](#)中的。

## create-contact-channel

以下代码示例演示了如何使用 create-contact-channel。

### AWS CLI

#### 创建联系渠道

为联系人 Akua Mansa 创建短信类型的联系渠道。可以创建短信、电子邮件或语音类型的联系渠道。

```
aws ssm-contacts create-contact-channel \
  --contact-id "arn:aws:ssm-contacts:us-east-1:111122223333:contact/akuam" \
  --name "akuas sms-test" \
  --type SMS \
  --delivery-address '{"SimpleAddress": "+15005550199"}'
```

输出：

```
{
  "ContactChannelArn": "arn:aws:ssm-contacts:us-east-1:111122223333:contact-
channel/akuam/02f506b9-ea5d-4764-af89-2daa793ff024"
}
```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateContactChannel](#)中的。

## create-contact

以下代码示例演示了如何使用 create-contact。

### AWS CLI

#### 创建联系人

以下create-contact示例使用空白计划在您的环境中创建联系人。创建联系渠道后，可以更新计划。使用带有此 create-contact-channel 命令输出 ARN 的命令。为该联系人创建联系渠道后，使用 update-contact 更新计划。

```
aws ssm-contacts create-contact \  
  --alias "akuam" \  
  --display-name "Akua Mansa" \  
  --type PERSONAL \  
  --plan '{"Stages": []}'
```

输出：

```
{  
  "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/akuam"  
}
```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateContact](#)中的。

## deactivate-contact-channel

以下代码示例演示了如何使用 deactivate-contact-channel。

### AWS CLI

#### 停用联系人频道

以下deactivate-contact-channel示例停用了联系渠道。停用联系人频道意味着在事件发生期间将不再对该联系人频道进行寻呼。您也可以随时使用activate-contact-channel命令重新激活联系人频道。

```
aws ssm-contacts deactivate-contact-channel \  
  --contact-channel-id "arn:aws:ssm-contacts:us-east-2:111122223333:contact-  
channel/akuam/fc7405c4-46b2-48b7-87b2-93e2f225b90d"
```

此命令不生成任何输出。

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeactivateContactChannel](#)中的。

## delete-contact-channel

以下代码示例演示了如何使用 delete-contact-channel。

## AWS CLI

### 删除联系人频道

以下delete-contact-channel示例删除了一个联系渠道。删除联系人频道可确保在事件发生期间不会对该联系人频道进行寻呼。

```
aws ssm-contacts delete-contact-channel \  
  --contact-channel-id "arn:aws:ssm-contacts:us-east-1:111122223333:contact-  
channel/akuam/13149bad-52ee-45ea-ae1e-45857f78f9b2"
```

此命令不生成任何输出。

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteContactChannel](#)中的。

## delete-contact

以下代码示例演示了如何使用 delete-contact。

## AWS CLI

### 删除联系人

以下delete-contact示例删除联系人。任何提及该联系人的升级计划都将无法再联系到该联系人。

```
aws ssm-contacts delete-contact \  
  --contact-id "arn:aws:ssm-contacts:us-east-1:111122223333:contact/alejrr"
```

此命令不生成任何输出。

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteContact](#)中的。

## describe-engagement

以下代码示例演示了如何使用 describe-engagement。

## AWS CLI

### 描述订婚的细节

以下describe-engagement示例列出了与联系人或升级计划互动的详细信息。主题和内容将发送到联系渠道。

```
aws ssm-contacts describe-engagement \  
  --engagement-id "arn:aws:ssm-contacts:us-east-2:111122223333:engagement/  
example_escalation/69e40ce1-8dbb-4d57-8962-5fbe7fc53356"
```

输出：

```
{  
  "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/  
example_escalation",  
  "EngagementArn": "arn:aws:ssm-contacts:us-east-2:111122223333:engagement/  
example_escalation/69e40ce1-8dbb-4d57-8962-5fbe7fc53356",  
  "Sender": "cli",  
  "Subject": "cli-test",  
  "Content": "Testing engagements via CLI",  
  "PublicSubject": "cli-test",  
  "PublicContent": "Testing engagements va CLI",  
  "StartTime": "2021-05-18T18:25:41.151000+00:00"  
}
```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEngagement](#)中的。

## describe-page

以下代码示例演示了如何使用 describe-page。

## AWS CLI

向联系渠道列出页面的详细信息

以下describe-page示例列出了联系渠道页面的详细信息。该页面将包括所提供的主题和内容。

```
aws ssm-contacts describe-page \  

```



```
--page-id "arn:aws:ssm-contacts:us-east-2:111122223333:page/akuam/ad0052bd-  
e606-498a-861b-25726292eb93"
```

输出：

```
{  
  "PageArn": "arn:aws:ssm-contacts:us-east-2:111122223333:page/akuam/ad0052bd-  
e606-498a-861b-25726292eb93",  
  "EngagementArn": "arn:aws:ssm-contacts:us-east-2:111122223333:engagement/  
akuam/78a29753-3674-4ac5-9f83-0468563567f0",  
  "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/akuam",  
  "Sender": "cli",  
  "Subject": "cli-test",  
  "Content": "Testing engagements via CLI",  
  "PublicSubject": "cli-test",  
  "PublicContent": "Testing engagements va CLI",  
  "SentTime": "2021-05-18T18:43:29.301000+00:00",  
  "ReadTime": "2021-05-18T18:43:55.708000+00:00",  
  "DeliveryTime": "2021-05-18T18:43:55.265000+00:00"  
}
```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePage](#)中的。

## get-contact-channel

以下代码示例演示了如何使用 get-contact-channel。

### AWS CLI

列出联系渠道的详细信息

以下get-contact-channel示例列出了联系渠道的详细信息。

```
aws ssm-contacts get-contact-channel \  
  --contact-channel-id "arn:aws:ssm-contacts:us-east-2:111122223333:contact-  
channel/akuam/fc7405c4-46b2-48b7-87b2-93e2f225b90d"
```

输出：

```
{  
  "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/akuam",
```

```
"ContactChannelArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact-
channel/akuam/fc7405c4-46b2-48b7-87b2-93e2f225b90d",
  "Name": "akuas sms",
  "Type": "SMS",
  "DeliveryAddress": {
    "SimpleAddress": "+15005550199"
  },
  "ActivationStatus": "ACTIVATED"
}
```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetContactChannel](#)中的。

## get-contact-policy

以下代码示例演示了如何使用 get-contact-policy。

### AWS CLI

列出联系人的资源政策

以下get-contact-policy示例列出了与指定联系人关联的资源策略。

```
aws ssm-contacts get-contact-policy \
  --contact-arn "arn:aws:ssm-contacts:us-east-1:111122223333:contact/akuam"
```

输出：

```
{
  "ContactArn": "arn:aws:ssm-contacts:us-east-1:111122223333:contact/akuam",
  "Policy": "{\n\"Version\": \"2012-10-17\", \"Statement\": [\n{\n\"Sid\":\n\n\"SharePolicyForDocumentationDralia\", \"Effect\": \"Allow\", \"Principal\":\n\n{\n\"AWS\": \"222233334444\"}, \"Action\": [\n\"ssm-contacts:GetContact\", \"ssm-
contacts:StartEngagement\", \"ssm-contacts:DescribeEngagement\", \"ssm-
contacts:ListPagesByEngagement\", \"ssm-contacts:StopEngagement\"], \"Resource
\": [\n\"arn:aws:ssm-contacts:*:111122223333:contact/akuam\", \"arn:aws:ssm-
contacts:*:111122223333:engagement/akuam/*\"]\n\n]]}"
}
```

有关更多信息，请参阅《事件管理器用户指南》中的[使用共享联系人和响应计划](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetContactPolicy](#)中的。

## get-contact

以下代码示例演示了如何使用 get-contact。

### AWS CLI

#### 示例 1：描述联系计划

以下 get-contact 示例描述了联系人。

```
aws ssm-contacts get-contact \
  --contact-id "arn:aws:ssm-contacts:us-east-2:111122223333:contact/akuam"
```

输出：

```
{
  "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/akuam",
  "Alias": "akuam",
  "DisplayName": "Akua Mansa",
  "Type": "PERSONAL",
  "Plan": {
    "Stages": [
      {
        "DurationInMinutes": 5,
        "Targets": [
          {
            "ChannelTargetInfo": {
              "ContactChannelId": "arn:aws:ssm-contacts:us-
east-2:111122223333:contact-channel/akuam/beb25840-5ac8-4644-95cc-7a8de390fa65",
              "RetryIntervalInMinutes": 1
            }
          }
        ]
      },
      {
        "DurationInMinutes": 5,
        "Targets": [
          {
            "ChannelTargetInfo": {
              "ContactChannelId": "arn:aws:ssm-contacts:us-
east-2:111122223333:contact-channel/akuam/49f3c24d-5f9f-4638-ae25-3f49e04229ad",
              "RetryIntervalInMinutes": 1
            }
          }
        ]
      }
    ]
  }
}
```

```

        ]
      },
      {
        "DurationInMinutes": 5,
        "Targets": [
          {
            "ChannelTargetInfo": {
              "ContactChannelId": "arn:aws:ssm-contacts:us-
east-2:111122223333:contact-channel/akuam/77d4f447-f619-4954-afff-85551e369c2a",
              "RetryIntervalInMinutes": 1
            }
          }
        ]
      }
    ]
  }
}

```

## 示例 2：描述升级计划

以下 `get-contact` 示例描述了升级计划。

```

aws ssm-contacts get-contact \
--contact-id "arn:aws:ssm-contacts:us-east-2:111122223333:contact/
example_escalation"

```

输出：

```

{
  "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/
example_escalation",
  "Alias": "example_escalation",
  "DisplayName": "Example Escalation",
  "Type": "ESCALATION",
  "Plan": {
    "Stages": [
      {
        "DurationInMinutes": 5,
        "Targets": [
          {
            "ContactTargetInfo": {
              "ContactId": "arn:aws:ssm-contacts:us-
east-2:111122223333:contact/akuam",

```

```

        "IsEssential": true
      }
    ]
  },
  {
    "DurationInMinutes": 5,
    "Targets": [
      {
        "ContactTargetInfo": {
          "ContactId": "arn:aws:ssm-contacts:us-
east-2:111122223333:contact/alejr",
          "IsEssential": false
        }
      }
    ]
  },
  {
    "DurationInMinutes": 0,
    "Targets": [
      {
        "ContactTargetInfo": {
          "ContactId": "arn:aws:ssm-contacts:us-
east-2:111122223333:contact/anasi",
          "IsEssential": false
        }
      }
    ]
  }
]
}
}
}

```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetContact](#)中的。

## list-contact-channels

以下代码示例演示了如何使用 list-contact-channels。

### AWS CLI

列出联系人的联系渠道

以下 `list-contact-channels` 示例列出了指定联系人的可用联系渠道。

```
aws ssm-contacts list-contact-channels \  
  --contact-id "arn:aws:ssm-contacts:us-east-2:111122223333:contact/akuam"
```

输出：

```
{  
  [  
    {  
      "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/  
akuam",  
      "Name": "akuas email",  
      "Type": "EMAIL",  
      "DeliveryAddress": {  
        "SimpleAddress": "akuam@example.com"  
      },  
      "ActivationStatus": "NOT_ACTIVATED"  
    },  
    {  
      "ContactChannelArn": "arn:aws:ssm-contacts:us-  
east-2:111122223333:contact-channel/akuam/fc7405c4-46b2-48b7-87b2-93e2f225b90d",  
      "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/  
akuam",  
      "Name": "akuas sms",  
      "Type": "SMS",  
      "DeliveryAddress": {  
        "SimpleAddress": "+15005550100"  
      },  
      "ActivationStatus": "ACTIVATED"  
    }  
  ]  
}
```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListContactChannels](#) 中的。

## list-contacts

以下代码示例演示了如何使用 `list-contacts`。

## AWS CLI

列出所有升级计划和联系人

以下`list-contacts`示例列出了您账户中的联系人和升级计划。

```
aws ssm-contacts list-contacts
```

输出：

```
{
  "Contacts": [
    {
      "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/
akuam",
      "Alias": "akuam",
      "DisplayName": "Akua Mansa",
      "Type": "PERSONAL"
    },
    {
      "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/
alejrr",
      "Alias": "alejrr",
      "DisplayName": "Alejandro Rosalez",
      "Type": "PERSONAL"
    },
    {
      "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/
anasi",
      "Alias": "anasi",
      "DisplayName": "Ana Carolina Silva",
      "Type": "PERSONAL"
    },
    {
      "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/
example_escalation",
      "Alias": "example_escalation",
      "DisplayName": "Example Escalation",
      "Type": "ESCALATION"
    }
  ]
}
```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListContacts](#)中的。

## list-engagements

以下代码示例演示了如何使用 list-engagements。

### AWS CLI

列出所有参与情况

以下list-engagements示例列出了升级计划和联系人的参与情况。您还可以列出单个事件的参与度。

```
aws ssm-contacts list-engagements
```

输出：

```
{
  "Engagements": [
    {
      "EngagementArn": "arn:aws:ssm-contacts:us-east-2:111122223333:engagement/akuam/91792571-0b53-4821-9f73-d25d13d9e529",
      "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/akuam",
      "Sender": "cli",
      "StartTime": "2021-05-18T20:37:50.300000+00:00"
    },
    {
      "EngagementArn": "arn:aws:ssm-contacts:us-east-2:111122223333:engagement/akuam/78a29753-3674-4ac5-9f83-0468563567f0",
      "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/akuam",
      "Sender": "cli",
      "StartTime": "2021-05-18T18:40:26.666000+00:00"
    },
    {
      "EngagementArn": "arn:aws:ssm-contacts:us-east-2:111122223333:engagement/example_escalation/69e40ce1-8dbb-4d57-8962-5fbe7fc53356",
      "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/example_escalation",

```



```

        "Sender": "cli",
        "StartTime": "2021-05-18T18:25:41.151000+00:00"
    },
    {
        "EngagementArn": "arn:aws:ssm-contacts:us-
east-2:111122223333:engagement/akuam/607ced0e-e8fa-4ea7-8958-a237b8803f8f",
        "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/
akuam",
        "Sender": "cli",
        "StartTime": "2021-05-18T18:20:58.093000+00:00"
    }
]
}

```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListEngagements](#)中的。

## list-page-receipts

以下代码示例演示了如何使用 list-page-receipts。

### AWS CLI

#### 列出页面收据

以下command-name示例列出了联系人是否收到页面。

```

aws ssm-contacts list-page-receipts \
  --page-id "arn:aws:ssm-contacts:us-east-2:111122223333:page/
akuam/94ea0c7b-56d9-46c3-b84a-a37c8b067ad3"

```

输出：

```

{
  "Receipts": [
    {
      "ContactChannelArn": "arn:aws:ssm-contacts:us-
east-2:111122223333:contact-channel/akuam/fc7405c4-46b2-48b7-87b2-93e2f225b90d",
      "ReceiptType": "DELIVERED",
      "ReceiptInfo": "425440",
      "ReceiptTime": "2021-05-18T20:42:57.485000+00:00"
    },
  ],
}

```

```

    {
      "ContactChannelArn": "arn:aws:ssm-contacts:us-
east-2:111122223333:contact-channel/akuam/fc7405c4-46b2-48b7-87b2-93e2f225b90d",
      "ReceiptType": "READ",
      "ReceiptInfo": "425440",
      "ReceiptTime": "2021-05-18T20:42:57.907000+00:00"
    },
    {
      "ContactChannelArn": "arn:aws:ssm-contacts:us-
east-2:111122223333:contact-channel/akuam/fc7405c4-46b2-48b7-87b2-93e2f225b90d",
      "ReceiptType": "SENT",
      "ReceiptInfo": "SM6656c19132f1465f9c9c1123a5dde7c9",
      "ReceiptTime": "2021-05-18T20:40:52.962000+00:00"
    }
  ]
}

```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPageReceipts](#)中的。

## list-pages-by-contact

以下代码示例演示了如何使用 `list-pages-by-contact`。

### AWS CLI

按联系人列出页面

以下`list-pages-by-contact`示例列出了指定联系人的所有页面。

```
aws ssm-contacts list-pages-by-contact \
  --contact-id "arn:aws:ssm-contacts:us-east-2:111122223333:contact/akuam"
```

输出：

```

{
  "Pages": [
    {
      "PageArn": "arn:aws:ssm-contacts:us-east-2:111122223333:page/akuam/
ad0052bd-e606-498a-861b-25726292eb93",
      "EngagementArn": "arn:aws:ssm-contacts:us-
east-2:111122223333:engagement/akuam/78a29753-3674-4ac5-9f83-0468563567f0",

```

```

    "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/
akuam",
    "Sender": "cli",
    "SentTime": "2021-05-18T18:43:29.301000+00:00",
    "DeliveryTime": "2021-05-18T18:43:55.265000+00:00",
    "ReadTime": "2021-05-18T18:43:55.708000+00:00"
  }
]
}

```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPagesByContact](#)中的。

## list-pages-by-engagement

以下代码示例演示了如何使用 list-pages-by-engagement。

### AWS CLI

要列出要联系渠道的页面，请从互动开始。

以下list-pages-by-engagement示例列出了在参与已定义的互动计划时出现的页面。

```

aws ssm-contacts list-pages-by-engagement \
  --engagement-id "arn:aws:ssm-contacts:us-east-2:111122223333:engagement/
akuam/78a29753-3674-4ac5-9f83-0468563567f0"

```

输出：

```

{
  "Pages": [
    {
      "PageArn": "arn:aws:ssm-contacts:us-east-2:111122223333:page/akuam/
ad0052bd-e606-498a-861b-25726292eb93",
      "EngagementArn": "arn:aws:ssm-contacts:us-
east-2:111122223333:engagement/akuam/78a29753-3674-4ac5-9f83-0468563567f0",
      "ContactArn": "arn:aws:ssm-contacts:us-east-2:111122223333:contact/
akuam",
      "Sender": "cli",
      "SentTime": "2021-05-18T18:40:27.245000+00:00"
    }
  ]
}

```

```
]
}
```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPagesByEngagement](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出联系人的标签

以下`list-tags-for-resource`示例列出了指定联系人的标签。

```
aws ssm-contacts list-tags-for-resource \
  --resource-arn "arn:aws:ssm-contacts:us-east-1:111122223333:contact/akuam"
```

输出：

```
{
  "Tags": [
    {
      "Key": "group1",
      "Value": "1"
    }
  ]
}
```

有关更多信息，请参阅《事件管理器用户指南》中的[标记](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## put-contact-policy

以下代码示例演示了如何使用 `put-contact-policy`。

### AWS CLI

共享联系人和参与度

以下put-contact-policy示例向联系人 Akua 添加资源政策，该策略与委托人共享联系和相关互动。

```
aws ssm-contacts put-contact-policy \
  --contact-arn "arn:aws:ssm-contacts:us-east-1:111122223333:contact/akuam" \
  --policy "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
  \"ExampleResourcePolicy\",\"Action\":[\"ssm-contacts:GetContact\", \"ssm-
  contacts:StartEngagement\", \"ssm-contacts:DescribeEngagement\", \"ssm-
  contacts:ListPagesByEngagement\", \"ssm-contacts:StopEngagement\"],
  \"Principal\":{\"AWS\":\"222233334444\"}, \"Effect\":\"Allow\", \"Resource
  \": [\"arn:aws:ssm-contacts:*:111122223333:contact/akuam\", \"arn:aws:ssm-
  contacts:*:111122223333:engagement/akuam/*\"]}]}"
```

此命令不生成任何输出。

有关更多信息，请参阅《事件管理器用户指南》中的[使用共享联系人和响应计划](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutContactPolicy](#)中的。

## send-activation-code

以下代码示例演示了如何使用 send-activation-code。

### AWS CLI

#### 发送激活码

以下send-activation-code示例向指定的联系渠道发送激活码和消息。

```
aws ssm-contacts send-activation-code \
  --contact-channel-id "arn:aws:ssm-contacts:us-east-1:111122223333:contact-
  channel/akuam/8ddae2d1-12c8-4e45-b852-c8587266c400"
```

此命令不生成任何输出。

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SendActivationCode](#)中的。

## start-engagement

以下代码示例演示了如何使用 start-engagement。

## AWS CLI

### 示例 1：拨打联系人的联系渠道

以下start-engagement页面联系人的联系渠道。发件人、主题、公共主题和公共内容都没有字段。事件管理器将主题和内容发送到提供的语音或电子邮件联系渠道。事件管理器将公共主题和公共内容发送到提供的短信联系渠道。发送者用于跟踪谁开始了互动。

```
aws ssm-contacts start-engagement \  
  --contact-id "arn:aws:ssm-contacts:us-east-2:111122223333:contact/akuam" \  
  --sender "cli" \  
  --subject "cli-test" \  
  --content "Testing engagements via CLI" \  
  --public-subject "cli-test" \  
  --public-content "Testing engagements va CLI"
```

输出：

```
{  
  "EngagementArn": "arn:aws:ssm-contacts:us-east-2:111122223333:engagement/  
akuam/607ced0e-e8fa-4ea7-8958-a237b8803f8f"  
}
```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

### 示例 2：在提供的升级计划中呼叫联系人。

以下内容通过升级计划start-engagement吸引联系人。每位联系人都会根据他们的参与计划进行寻呼。

```
aws ssm-contacts start-engagement \  
  --contact-id "arn:aws:ssm-contacts:us-east-2:111122223333:contact/  
example_escalation" \  
  --sender "cli" \  
  --subject "cli-test" \  
  --content "Testing engagements via CLI" \  
  --public-subject "cli-test" \  
  --public-content "Testing engagements va CLI"
```

输出：

```
{
```

```
"EngagementArn": "arn:aws:ssm-contacts:us-east-2:111122223333:engagement/
example_escalation/69e40ce1-8dbb-4d57-8962-5fbe7fc53356"
}
```

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartEngagement](#)中的。

## stop-engagement

以下代码示例演示了如何使用 stop-engagement。

### AWS CLI

#### 停止互动

以下stop-engagement示例阻止互动向更多联系人和联系人渠道传呼。

```
aws ssm-contacts stop-engagement \
  --engagement-id "arn:aws:ssm-contacts:us-east-2:111122223333:engagement/
example_escalation/69e40ce1-8dbb-4d57-8962-5fbe7fc53356"
```

此命令不生成任何输出。

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopEngagement](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

#### 为联系人添加标签

以下tag-resource示例使用提供的标签键值对为指定的联系人添加标签。

```
aws ssm-contacts tag-resource \
  --resource-arn "arn:aws:ssm-contacts:us-east-1:111122223333:contact/akuam" \
  --tags '[{"Key":"group1","Value":"1"}]'
```

此命令不生成任何输出。

有关更多信息，请参阅《事件管理器用户指南》中的[标记](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从联系人中移除标签

以下untag-resource示例从指定联系人中删除 group1 标签。

```
aws ssm-contacts untag-resource \  
  --resource-arn "arn:aws:ssm-contacts:us-east-1:111122223333:contact/akuam" \  
  --tag-keys "group1"
```

此命令不生成任何输出。

有关更多信息，请参阅《事件管理器用户指南》中的[标记](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-contact-channel

以下代码示例演示了如何使用 update-contact-channel。

### AWS CLI

更新联系渠道

以下update-contact-channel示例更新了联系渠道的名称和收货地址。

```
aws ssm-contacts update-contact-channel \  
  --contact-channel-id "arn:aws:ssm-contacts:us-east-2:111122223333:contact-  
channel/akuam/49f3c24d-5f9f-4638-ae25-3f49e04229ad" \  
  --name "akuas voice channel" \  
  --delivery-address '{"SimpleAddress": "+15005550198"}'
```



此命令不生成任何输出。

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateContactChannel](#)中的。

## update-contact

以下代码示例演示了如何使用 update-contact。

### AWS CLI

#### 更新联系人参与计划

以下update-contact示例更新了联系人 Akua 的参与计划，使其包括三种类型的联系渠道。这是在为 Akua 创建联系渠道之后完成的。

```
aws ssm-contacts update-contact \  
  --contact-id "arn:aws:ssm-contacts:us-east-2:111122223333:contact/akuam" \  
  --plan '{"Stages": [{"DurationInMinutes": 5, "Targets": [{"ChannelTargetInfo":  
  {"ContactChannelId": "arn:aws:ssm-contacts:us-east-2:111122223333:contact-  
channel/akuam/beb25840-5ac8-4644-95cc-7a8de390fa65", "RetryIntervalInMinutes":  
1 }]}], {"DurationInMinutes": 5, "Targets": [{"ChannelTargetInfo":  
{"ContactChannelId": "arn:aws:ssm-contacts:us-east-2:111122223333:contact-channel/  
akuam/49f3c24d-5f9f-4638-ae25-3f49e04229ad", "RetryIntervalInMinutes": 1}]}}],  
{"DurationInMinutes": 5, "Targets": [{"ChannelTargetInfo": {"ContactChannelId":  
"arn:aws:ssm-contacts:us-east-2:111122223333:contact-channel/akuam/77d4f447-  
f619-4954-afff-85551e369c2a", "RetryIntervalInMinutes": 1 }]}]}'
```

此命令不生成任何输出。

有关更多信息，请参阅事件管理器用户指南中的[联系人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateContact](#)中的。

## Amazon Inspector 示例使用 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon Inspector 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## add-attributes-to-findings

以下代码示例演示了如何使用 `add-attributes-to-findings`。

AWS CLI

为调查结果添加属性

以下 `add-attribute-to-finding` 命令将键为 `Example` 和值为 `example` 的属性分配给 ARN `arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-811VIE0D/run/0-Z02cjjug/finding/0-T8yM9mEU` 的查找结果：

```
aws inspector add-attributes-to-findings --finding-arns arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-811VIE0D/run/0-Z02cjjug/finding/0-T8yM9mEU --attributes key=Example,value=example
```

输出：

```
{
  "failedItems": {}
}
```

有关更多信息，请参阅亚马逊 Inspector 指南中的 [Amazon Inspector 调查结果](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AddAttributesToFindings](#) 中的。

## create-assessment-target

以下代码示例演示了如何使用 `create-assessment-target`。

## AWS CLI

### 创建评估目标

以下 `create-assessment-target` 命令 `ExampleAssessmentTarget` 使用资源组创建名为 ARN 的评估目标：`arn:aws:inspector:us-west-2:123456789012:resourcegroup/0-AB6DMKnv`

```
aws inspector create-assessment-target --assessment-target-name
ExampleAssessmentTarget --resource-group-arn arn:aws:inspector:us-
west-2:123456789012:resourcegroup/0-AB6DMKnv
```

输出：

```
{
  "assessmentTargetArn": "arn:aws:inspector:us-west-2:123456789012:target/0-
nvgVhaxX"
}
```

有关更多信息，请参阅 Amazon Inspector 指南中的亚马逊检查员评估目标。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateAssessmentTarget](#) 中的。

## **create-assessment-template**

以下代码示例演示了如何使用 `create-assessment-template`。

## AWS CLI

### 创建评估模板

以下 `create-assessment-template` 命令为评估目标创建名为 `ExampleAssessmentTemplate` 的评估模板，ARN 为：`arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX`

```
aws inspector create-assessment-template --assessment-target-arn
arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX --assessment-template-
name ExampleAssessmentTemplate --duration-in-seconds 180 --rules-package-arns
arn:aws:inspector:us-west-2:758058086616:rulespackage/0-9hgA516p --user-attributes-
for-findings key=ExampleTag,value=examplevalue
```

输出：

```
{
  "assessmentTemplateArn": "arn:aws:inspector:us-west-2:123456789012:target/0-
nvgVhaxX/template/0-it5r2S4T"
}
```

有关更多信息，请参阅 Amazon Inspector 指南中的 Amazon Inspector 评估模板和评估运行。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateAssessmentTemplate](#) 中的。

## create-resource-group

以下代码示例演示了如何使用 create-resource-group。

### AWS CLI

创建资源组

以下 create-resource-group 命令使用标签键 Name 和价值创建资源组 example：

```
aws inspector create-resource-group --resource-group-tags key=Name,value=example
```

输出：

```
{
  "resourceGroupArn": "arn:aws:inspector:us-west-2:123456789012:resourcegroup/0-
AB6DMKnv"
}
```

有关更多信息，请参阅 Amazon Inspector 指南中的亚马逊检查员评估目标。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateResourceGroup](#) 中的。

## delete-assessment-run

以下代码示例演示了如何使用 delete-assessment-run。

### AWS CLI

要删除评估，请运行

以下 delete-assessment-run 命令删除运行的 ARN 为的评估：arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-it5r2S4T/run/0-11LMTAVe

```
aws inspector delete-assessment-run --assessment-run-arn arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-it5r2S4T/run/0-11LMTAVe
```

有关更多信息，请参阅 Amazon Inspector 指南中的 Amazon Inspector 评估模板和评估运行。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteAssessmentRun](#) 中的。

## delete-assessment-target

以下代码示例演示了如何使用 delete-assessment-target。

### AWS CLI

删除评估目标

以下 delete-assessment-target 命令删除 ARN 为的评估目标：arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq

```
aws inspector delete-assessment-target --assessment-target-arn arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq
```

有关更多信息，请参阅 Amazon Inspector 指南中的亚马逊检查员评估目标。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteAssessmentTarget](#) 中的。

## delete-assessment-template

以下代码示例演示了如何使用 delete-assessment-template。

### AWS CLI

删除评估模板

以下 delete-assessment-template 命令删除 ARN 为的评估模板：arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-it5r2S4T

```
aws inspector delete-assessment-template --assessment-template-arn arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-it5r2S4T
```

有关更多信息，请参阅 Amazon Inspector 指南中的 Amazon Inspector 评估模板和评估运行。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAssessmentTemplate](#)中的。

## describe-assessment-runs

以下代码示例演示了如何使用 describe-assessment-runs。

### AWS CLI

#### 描述评估运行

以下describe-assessment-run命令描述了在 ARN 为的情况下运行的评估：  
arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw/run/0-MKkpXXPE

```
aws inspector describe-assessment-runs --assessment-run-arns arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw/run/0-MKkpXXPE
```

#### 输出：

```
{
  "assessmentRuns": [
    {
      "arn": "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw/run/0-MKkpXXPE",
      "assessmentTemplateArn": "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw",
      "completedAt": 1458680301.4,
      "createdAt": 1458680170.035,
      "dataCollected": true,
      "durationInSeconds": 3600,
      "name": "Run 1 for ExampleAssessmentTemplate",
      "notifications": [],
      "rulesPackageArns": [
        "arn:aws:inspector:us-west-2:758058086616:rulespackage/0-X1KXtawP"
      ],
      "startedAt": 1458680170.161,
      "state": "COMPLETED",
      "stateChangedAt": 1458680301.4,
      "stateChanges": [
        {
          "state": "CREATED",
          "stateChangedAt": 1458680170.035
        }
      ]
    }
  ]
}
```

```
    },
    {
      "state": "START_DATA_COLLECTION_PENDING",
      "stateChangedAt": 1458680170.065
    },
    {
      "state": "START_DATA_COLLECTION_IN_PROGRESS",
      "stateChangedAt": 1458680170.096
    },
    {
      "state": "COLLECTING_DATA",
      "stateChangedAt": 1458680170.161
    },
    {
      "state": "STOP_DATA_COLLECTION_PENDING",
      "stateChangedAt": 1458680239.883
    },
    {
      "state": "DATA_COLLECTED",
      "stateChangedAt": 1458680299.847
    },
    {
      "state": "EVALUATING_RULES",
      "stateChangedAt": 1458680300.099
    },
    {
      "state": "COMPLETED",
      "stateChangedAt": 1458680301.4
    }
  ],
  "userAttributesForFindings": []
}
],
"failedItems": {}
}
```

有关更多信息，请参阅 Amazon Inspector 指南中的 Amazon Inspector 评估模板和评估运行。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAssessmentRuns](#) 中的。

## describe-assessment-targets

以下代码示例演示了如何使用 describe-assessment-targets。

## AWS CLI

### 描述评估目标

以下describe-assessment-targets命令描述了 ARN 为的评估目

标：arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq

```
aws inspector describe-assessment-targets --assessment-target-arns
arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq
```

输出：

```
{
  "assessmentTargets": [
    {
      "arn": "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq",
      "createdAt": 1458074191.459,
      "name": "ExampleAssessmentTarget",
      "resourceGroupArn": "arn:aws:inspector:us-
west-2:123456789012:resourcegroup/0-PyGXopAI",
      "updatedAt": 1458074191.459
    }
  ],
  "failedItems": {}
}
```

有关更多信息，请参阅 Amazon Inspector 指南中的亚马逊检查员评估目标。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAssessmentTargets](#)中的。

## describe-assessment-templates

以下代码示例演示了如何使用 describe-assessment-templates。

## AWS CLI

### 描述评估模板

以下describe-assessment-templates命令描述了 ARN 为的评估模

板：arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/  
template/0-4r1V2mAw



```
aws inspector describe-assessment-templates --assessment-template-arns
arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw
```

输出：

```
{
  "assessmentTemplates": [
    {
      "arn": "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/
template/0-4r1V2mAw",
      "assessmentTargetArn": "arn:aws:inspector:us-
west-2:123456789012:target/0-0kFIPusq",
      "createdAt": 1458074191.844,
      "durationInSeconds": 3600,
      "name": "ExampleAssessmentTemplate",
      "rulesPackageArns": [
        "arn:aws:inspector:us-west-2:758058086616:rulespackage/0-X1KXtawP"
      ],
      "userAttributesForFindings": []
    }
  ],
  "failedItems": {}
}
```

有关更多信息，请参阅 Amazon Inspector 指南中的 Amazon Inspector 评估模板和评估运行。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAssessmentTemplates](#) 中的。

## describe-cross-account-access-role

以下代码示例演示了如何使用 `describe-cross-account-access-role`。

### AWS CLI

描述跨账户访问角色

以下 `describe-cross-account-access-role` 命令描述了允许 Amazon Inspector 访问您的 AWS 账户的 IAM 角色：

```
aws inspector describe-cross-account-access-role
```

输出：

```
{
  "registeredAt": 1458069182.826,
  "roleArn": "arn:aws:iam::123456789012:role/inspector",
  "valid": true
}
```

有关更多信息，请参阅亚马逊 Inspector 指南中的设置 Amazon Inspector。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeCrossAccountAccessRole](#) 中的。

## describe-findings

以下代码示例演示了如何使用 describe-findings。

### AWS CLI

#### 描述调查结果

以下 describe-findings 命令描述了 ARN 为的结果：arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw/run/0-MKkpXXPE/finding/0-HwPnsDm4

```
aws inspector describe-findings --finding-arns arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw/run/0-MKkpXXPE/finding/0-HwPnsDm4
```

输出：

```
{
  "failedItems": {},
  "findings": [
    {
      "arn": "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw/run/0-MKkpXXPE/finding/0-HwPnsDm4",
      "assetAttributes": {
        "ipv4Addresses": [],
        "schemaVersion": 1
      },
      "assetType": "ec2-instance",
      "attributes": [],
      "confidence": 10,
      "createdAt": 1458680301.37,
```

```

        "description": "Amazon Inspector did not find any potential security
issues during this assessment.",
        "indicatorOfCompromise": false,
        "numericSeverity": 0,
        "recommendation": "No remediation needed.",
        "schemaVersion": 1,
        "service": "Inspector",
        "serviceAttributes": {
            "assessmentRunArn": "arn:aws:inspector:us-
west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw/run/0-MKkpXXPE",
            "rulesPackageArn": "arn:aws:inspector:us-
west-2:758058086616:rulespackage/0-X1KXtawP",
            "schemaVersion": 1
        },
        "severity": "Informational",
        "title": "No potential security issues found",
        "updatedAt": 1458680301.37,
        "userAttributes": []
    }
]
}

```

有关更多信息，请参阅亚马逊 Inspector 指南中的 [Amazon Inspector 调查结果](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeFindings](#) 中的。

## describe-resource-groups

以下代码示例演示了如何使用 `describe-resource-groups`。

### AWS CLI

#### 描述资源组

以下 `describe-resource-groups` 命令描述了 ARN 为的资源组：`arn:aws:inspector:us-west-2:123456789012:resourcegroup/0-PyGXopAI`

```
aws inspector describe-resource-groups --resource-group-arns arn:aws:inspector:us-
west-2:123456789012:resourcegroup/0-PyGXopAI
```

输出：

```
{
```

```

    "failedItems": {},
    "resourceGroups": [
      {
        "arn": "arn:aws:inspector:us-west-2:123456789012:resourcegroup/0-
PyGXopAI",
        "createdAt": 1458074191.098,
        "tags": [
          {
            "key": "Name",
            "value": "example"
          }
        ]
      }
    ]
  }
}

```

有关更多信息，请参阅 Amazon Inspector 指南中的亚马逊检查员评估目标。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeResourceGroups](#) 中的。

## describe-rules-packages

以下代码示例演示了如何使用 describe-rules-packages。

### AWS CLI

#### 描述规则包

以下 describe-rules-packages 命令描述了 ARN 为的规则包：arn:aws:inspector:us-west-2:758058086616:rulespackage/0-9hgA516p

```
aws inspector describe-rules-packages --rules-package-arns arn:aws:inspector:us-west-2:758058086616:rulespackage/0-9hgA516p
```

#### 输出：

```

{
  "failedItems": {},
  "rulesPackages": [
    {
      "arn": "arn:aws:inspector:us-
west-2:758058086616:rulespackage/0-9hgA516p",

```

```

        "description": "The rules in this package help verify whether the EC2
        instances in your application are exposed to Common Vulnerabilities and
        Exposures (CVEs). Attacks can exploit unpatched vulnerabilities to
        compromise the confidentiality, integrity, or availability of your service
        or data. The CVE system provides a reference for publicly known
        information security vulnerabilities and exposures. For more information, see
        [https://cve.mitre.org/](https://cve.mitre.org/). If a particular CVE
        appears in one of the produced Findings at the end of a completed
        Inspector assessment, you can search [https://cve.mitre.org/](https://
        cve.mitre.org/) using the CVE's ID (for example, \"CVE-2009-0021\") to
        find detailed information about this CVE, its severity, and how to
        mitigate it. ",
        "name": "Common Vulnerabilities and Exposures",
        "provider": "Amazon Web Services, Inc.",
        "version": "1.1"
    }
]
}

```

有关更多信息，请参阅 Amazon Inspector 指南中的 [A mazon Inspec tor 规则包和规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeRulesPackages](#) 中的。

## get-telemetry-metadata

以下代码示例演示了如何使用 get-telemetry-metadata。

### AWS CLI

#### 获取遥测元数据

以下 get-telemetry-metadata 命令生成有关为评估运行而收集的数据的信息，ARN 为：arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw/run/0-MKkpXXPE

```
aws inspector get-telemetry-metadata --assessment-run-arn arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw/run/0-MKkpXXPE
```

输出：

```
{
  "telemetryMetadata": [
    {
```

```
    "count": 2,  
    "dataSize": 345,  
    "messageType": "InspectorDuplicateProcess"  
  },  
  {  
    "count": 3,  
    "dataSize": 255,  
    "messageType": "InspectorTimeEventMsg"  
  },  
  {  
    "count": 4,  
    "dataSize": 1082,  
    "messageType": "InspectorNetworkInterface"  
  },  
  {  
    "count": 2,  
    "dataSize": 349,  
    "messageType": "InspectorDnsEntry"  
  },  
  {  
    "count": 11,  
    "dataSize": 2514,  
    "messageType": "InspectorDirectoryInfoMsg"  
  },  
  {  
    "count": 1,  
    "dataSize": 179,  
    "messageType": "InspectorTcpV6ListeningPort"  
  },  
  {  
    "count": 101,  
    "dataSize": 10949,  
    "messageType": "InspectorTerminal"  
  },  
  {  
    "count": 26,  
    "dataSize": 5916,  
    "messageType": "InspectorUser"  
  },  
  {  
    "count": 282,  
    "dataSize": 32148,  
    "messageType": "InspectorDynamicallyLoadedCodeModule"  
  },  
}
```

```
{
  "count": 18,
  "dataSize": 10172,
  "messageType": "InspectorCreateProcess"
},
{
  "count": 3,
  "dataSize": 8001,
  "messageType": "InspectorProcessPerformance"
},
{
  "count": 1,
  "dataSize": 360,
  "messageType": "InspectorOperatingSystem"
},
{
  "count": 6,
  "dataSize": 546,
  "messageType": "InspectorStopProcess"
},
{
  "count": 1,
  "dataSize": 1553,
  "messageType": "InspectorInstanceMetaData"
},
{
  "count": 2,
  "dataSize": 434,
  "messageType": "InspectorTcpV4Connection"
},
{
  "count": 474,
  "dataSize": 2960322,
  "messageType": "InspectorPackageInfo"
},
{
  "count": 3,
  "dataSize": 2235,
  "messageType": "InspectorSystemPerformance"
},
{
  "count": 105,
  "dataSize": 46048,
  "messageType": "InspectorCodeModule"
}
```

```
  },
  {
    "count": 1,
    "dataSize": 182,
    "messageType": "InspectorUdpV6ListeningPort"
  },
  {
    "count": 2,
    "dataSize": 371,
    "messageType": "InspectorUdpV4ListeningPort"
  },
  {
    "count": 18,
    "dataSize": 8362,
    "messageType": "InspectorKernelModule"
  },
  {
    "count": 29,
    "dataSize": 48788,
    "messageType": "InspectorConfigurationInfo"
  },
  {
    "count": 1,
    "dataSize": 79,
    "messageType": "InspectorMonitoringStart"
  },
  {
    "count": 5,
    "dataSize": 0,
    "messageType": "InspectorSplitMsgBegin"
  },
  {
    "count": 51,
    "dataSize": 4593,
    "messageType": "InspectorGroup"
  },
  {
    "count": 1,
    "dataSize": 184,
    "messageType": "InspectorTcpV4ListeningPort"
  },
  {
    "count": 1159,
    "dataSize": 3146579,
```



```

        "messageType": "Total"
      },
      {
        "count": 5,
        "dataSize": 0,
        "messageType": "InspectorSplitMsgEnd"
      },
      {
        "count": 1,
        "dataSize": 612,
        "messageType": "InspectorLoadImageInProgress"
      }
    ]
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTelemetryMetadata](#)中的。

## list-assessment-run-agents

以下代码示例演示了如何使用 list-assessment-run-agents。

### AWS CLI

要列出评估，请运行代理

以下list-assessment-run-agents命令列出了使用指定 ARN 运行的评估代理。

```

aws inspector list-assessment-run-agents \
  --assessment-run-arn arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/
  template/0-4r1V2mAw/run/0-MKkpXXPE

```

输出：

```

{
  "assessmentRunAgents": [
    {
      "agentHealth": "HEALTHY",
      "agentHealthCode": "HEALTHY",
      "agentId": "i-49113b93",
      "assessmentRunArn": "arn:aws:inspector:us-
west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw/run/0-MKkpXXPE",
      "telemetryMetadata": [
        {

```

```
    "count": 2,  
    "dataSize": 345,  
    "messageType": "InspectorDuplicateProcess"  
  },  
  {  
    "count": 3,  
    "dataSize": 255,  
    "messageType": "InspectorTimeEventMsg"  
  },  
  {  
    "count": 4,  
    "dataSize": 1082,  
    "messageType": "InspectorNetworkInterface"  
  },  
  {  
    "count": 2,  
    "dataSize": 349,  
    "messageType": "InspectorDnsEntry"  
  },  
  {  
    "count": 11,  
    "dataSize": 2514,  
    "messageType": "InspectorDirectoryInfoMsg"  
  },  
  {  
    "count": 1,  
    "dataSize": 179,  
    "messageType": "InspectorTcpV6ListeningPort"  
  },  
  {  
    "count": 101,  
    "dataSize": 10949,  
    "messageType": "InspectorTerminal"  
  },  
  {  
    "count": 26,  
    "dataSize": 5916,  
    "messageType": "InspectorUser"  
  },  
  {  
    "count": 282,  
    "dataSize": 32148,  
    "messageType": "InspectorDynamicallyLoadedCodeModule"  
  },  
}
```

```
{
  "count": 18,
  "dataSize": 10172,
  "messageType": "InspectorCreateProcess"
},
{
  "count": 3,
  "dataSize": 8001,
  "messageType": "InspectorProcessPerformance"
},
{
  "count": 1,
  "dataSize": 360,
  "messageType": "InspectorOperatingSystem"
},
{
  "count": 6,
  "dataSize": 546,
  "messageType": "InspectorStopProcess"
},
{
  "count": 1,
  "dataSize": 1553,
  "messageType": "InspectorInstanceMetaData"
},
{
  "count": 2,
  "dataSize": 434,
  "messageType": "InspectorTcpV4Connection"
},
{
  "count": 474,
  "dataSize": 2960322,
  "messageType": "InspectorPackageInfo"
},
{
  "count": 3,
  "dataSize": 2235,
  "messageType": "InspectorSystemPerformance"
},
{
  "count": 105,
  "dataSize": 46048,
  "messageType": "InspectorCodeModule"
}
```

```
    },
    {
      "count": 1,
      "dataSize": 182,
      "messageType": "InspectorUdpV6ListeningPort"
    },
    {
      "count": 2,
      "dataSize": 371,
      "messageType": "InspectorUdpV4ListeningPort"
    },
    {
      "count": 18,
      "dataSize": 8362,
      "messageType": "InspectorKernelModule"
    },
    {
      "count": 29,
      "dataSize": 48788,
      "messageType": "InspectorConfigurationInfo"
    },
    {
      "count": 1,
      "dataSize": 79,
      "messageType": "InspectorMonitoringStart"
    },
    {
      "count": 5,
      "dataSize": 0,
      "messageType": "InspectorSplitMsgBegin"
    },
    {
      "count": 51,
      "dataSize": 4593,
      "messageType": "InspectorGroup"
    },
    {
      "count": 1,
      "dataSize": 184,
      "messageType": "InspectorTcpV4ListeningPort"
    },
    {
      "count": 1159,
      "dataSize": 3146579,
```

```
        "messageType": "Total"
      },
      {
        "count": 5,
        "dataSize": 0,
        "messageType": "InspectorSplitMsgEnd"
      },
      {
        "count": 1,
        "dataSize": 612,
        "messageType": "InspectorLoadImageInProgress"
      }
    ]
  }
]
```

有关更多信息，请参阅 Amazon Inspector 用户指南中的[AWS 代理](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAssessmentRunAgents](#) 中的。

## list-assessment-runs

以下代码示例演示了如何使用 `list-assessment-runs`。

### AWS CLI

列出评估运行情况

以下 `list-assessment-runs` 命令列出了所有现有的评估运行。

```
aws inspector list-assessment-runs
```

输出：

```
{
  "assessmentRunArns": [
    "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/
template/0-4r1V2mAw/run/0-MKkpXXPE",
    "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/
template/0-4r1V2mAw/run/0-v5D6fI3v"
  ]
}
```

```
}
```

有关更多信息，请参阅 [Amazon Inspector 用户指南中的 Amazon Inspector 评估模板和评估运行](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAssessmentRuns](#) 中的。

## list-assessment-targets

以下代码示例演示了如何使用 `list-assessment-targets`。

### AWS CLI

列出评估目标

以下 `list-assessment-targets` 命令列出了所有现有的评估目标：

```
aws inspector list-assessment-targets
```

输出：

```
{
  "assessmentTargetArns": [
    "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq"
  ]
}
```

有关更多信息，请参阅 Amazon Inspector 指南中的亚马逊检查员评估目标。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAssessmentTargets](#) 中的。

## list-assessment-templates

以下代码示例演示了如何使用 `list-assessment-templates`。

### AWS CLI

列出评估模板

以下 `list-assessment-templates` 命令列出了所有现有的评估模板：

```
aws inspector list-assessment-templates
```

输出：

```
{
  "assessmentTemplateArns": [
    "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/
template/0-4r1V2mAw",
    "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-
Uza6ihLh"
  ]
}
```

有关更多信息，请参阅 Amazon Inspector 指南中的 Amazon Inspector 评估模板和评估运行。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAssessmentTemplates](#) 中的。

## list-event-subscriptions

以下代码示例演示了如何使用 list-event-subscriptions。

AWS CLI

列出活动订阅

以下 list-event-subscriptions 命令列出了评估模板的所有事件订阅，ARN 为：arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-7sbz2Kz0

```
aws inspector list-event-subscriptions --resource-arn arn:aws:inspector:us-
west-2:123456789012:target/0-nvgVhaxX/template/0-7sbz2Kz0
```

输出：

```
{
  "subscriptions": [
    {
      "eventSubscriptions": [
        {
          "event": "ASSESSMENT_RUN_COMPLETED",
          "subscribedAt": 1459455440.867
        }
      ],
    }
  ],
}
```

```
        "resourceArn": "arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-7sbz2Kz0",
        "topicArn": "arn:aws:sns:us-west-2:123456789012:exampletopic"
    }
]
}
```

有关更多信息，请参阅 Amazon Inspector 指南中的 Amazon Inspector 评估模板和评估运行。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListEventSubscriptions](#) 中的。

## list-findings

以下代码示例演示了如何使用 list-findings。

### AWS CLI

列出调查结果

以下 list-findings 命令列出了所有生成的调查结果：

```
aws inspector list-findings
```

输出：

```
{
  "findingArns": [
    "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw/run/0-MKkpXXPE/finding/0-HwPnsDm4",
    "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-4r1V2mAw/run/0-v5D6fI3v/finding/0-tyvmqBLY"
  ]
}
```

有关更多信息，请参阅亚马逊 Inspector 指南中的 Amazon Inspector 调查结果。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListFindings](#) 中的。

## list-rules-packages

以下代码示例演示了如何使用 list-rules-packages。



## AWS CLI

### 列出规则包

以下 `list-rules-packages` 命令列出了所有可用的 Inspector 规则包：

```
aws inspector list-rules-packages
```

输出：

```
{
  "rulesPackageArns": [
    "arn:aws:inspector:us-west-2:758058086616:rulespackage/0-9hgA516p",
    "arn:aws:inspector:us-west-2:758058086616:rulespackage/0-H5hpSawc",
    "arn:aws:inspector:us-west-2:758058086616:rulespackage/0-JJ0tZiqQ",
    "arn:aws:inspector:us-west-2:758058086616:rulespackage/0-vg5GGHSD"
  ]
}
```

有关更多信息，请参阅 Amazon Inspector 指南中的 [Amazon Inspector 规则包和规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListRulesPackages](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

#### 列出资源的标签

以下 `list-tags-for-resource` 命令列出了与评估模板关联的所有标签，ARN 为：`arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-gcwFliYu`

```
aws inspector list-tags-for-resource --resource-arn arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-gcwFliYu
```

输出：

```
{
  "tags": [
```

```
{
  "key": "Name",
  "value": "Example"
}
]
```

有关更多信息，请参阅 Amazon Inspector 指南中的 Amazon Inspector 评估模板和评估运行。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## preview-agents

以下代码示例演示了如何使用 `preview-agents`。

### AWS CLI

#### 预览代理

以下 `preview-agents` 命令预览安装在属于评估目标的 EC2 实例上的代理，ARN 为 `arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusqN` 为：

```
aws inspector preview-agents --preview-agents-arn arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusqN
```

输出：

```
{
  "agentPreviews": [
    {
      "agentId": "i-49113b93"
    }
  ]
}
```

有关更多信息，请参阅 Amazon Inspector 指南中的亚马逊检查员评估目标。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PreviewAgents](#) 中的。

## register-cross-account-access-role

以下代码示例演示了如何使用 `register-cross-account-access-role`。

## AWS CLI

### 注册跨账户访问角色

以下 `register-cross-account-access-role` 命令使用 Amazon Inspector 的 ARN 注册 IAM 角色 `arn:aws:iam::123456789012:role/inspector`，当你调用 `preview-agents` 命令时，评估运行开始时，Amazon Inspector 使用该角色列出你的 EC2 实例：

```
aws inspector register-cross-account-access-role --role-arn
arn:aws:iam::123456789012:role/inspector
```

有关更多信息，请参阅亚马逊 Inspector 指南中的设置 Amazon Inspector。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterCrossAccountAccessRole](#) 中的。

## remove-attributes-from-findings

以下代码示例演示了如何使用 `remove-attributes-from-findings`。

### AWS CLI

#### 从调查结果中移除属性

以下 `remove-attributes-from-finding` 命令 `example` 从查找结果中移除键为 `Example` 和值为 `example` 的属性，ARN 为 `arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/template/0-811VIE0D/run/0-Z02cjjug/finding/0-T8yM9mEU`

```
aws inspector remove-attributes-from-findings --finding-arns arn:aws:inspector:us-
west-2:123456789012:target/0-0kFIPusq/template/0-811VIE0D/run/0-Z02cjjug/finding/0-
T8yM9mEU --attribute-keys key=Example,value=example
```

输出：

```
{
  "failedItems": {}
}
```

有关更多信息，请参阅亚马逊 Inspector 指南中的 Amazon Inspector 调查结果。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RemoveAttributesFromFindings](#) 中的。

## set-tags-for-resource

以下代码示例演示了如何使用 `set-tags-for-resource`。

### AWS CLI

为资源设置标签

以下 `set-tags-for-resource` 命令将键为 `Example` 和值为 `example` 的标签设置 `example` 为评估模板，ARN 为 `arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-7sbz2Kz0`

```
aws inspector set-tags-for-resource --resource-arn arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-7sbz2Kz0 --tags key=Example,value=example
```

有关更多信息，请参阅 Amazon Inspector 指南中的 Amazon Inspector 评估模板和评估运行。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SetTagsForResource](#) 中的。

## start-assessment-run

以下代码示例演示了如何使用 `start-assessment-run`。

### AWS CLI

开始评估运行

以下 `start-assessment-run` 命令开始 `examplerun` 使用评估模板命名的评估运行，ARN 为 `arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-it5r2S4T`

```
aws inspector start-assessment-run --assessment-run-name exemplarun --assessment-template-arn arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-it5r2S4T
```

输出：

```
{
  "assessmentRunArn": "arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-it5r2S4T/run/0-j0oroxyY"
}
```

有关更多信息，请参阅 Amazon Inspector 指南中的 [Amazon Inspector 评估模板和评估运行](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartAssessmentRun](#) 中的。

## stop-assessment-run

以下代码示例演示了如何使用 stop-assessment-run。

### AWS CLI

#### 停止评估运行

以下 stop-assessment-run 命令停止评估运行，ARN 为：arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-it5r2S4T/run/0-j0oroxyY

```
aws inspector stop-assessment-run --assessment-run-arn arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-it5r2S4T/run/0-j0oroxyY
```

有关更多信息，请参阅 Amazon Inspector 指南中的 [Amazon Inspector 评估模板和评估运行](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StopAssessmentRun](#) 中的。

## subscribe-to-event

以下代码示例演示了如何使用 subscribe-to-event。

### AWS CLI

#### 订阅活动

以下示例启用了向主题发送有关该 ASSESSMENT\_RUN\_COMPLETED 事件的 Amazon SNS 通知的流程，ARN 为 arn:aws:sns:us-west-2:123456789012:exampletopic

```
aws inspector subscribe-to-event \
  --event ASSESSMENT_RUN_COMPLETED \
  --resource-arn arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/
template/0-7sbz2Kz0 \
  --topic-arn arn:aws:sns:us-west-2:123456789012:exampletopic
```

此命令不生成任何输出。

有关更多信息，请参阅 [Amazon Inspector 指南中的 Amazon Inspector 评估模板和评估运行](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SubscribeToEvent](#)中的。

## unsubscribe-from-event

以下代码示例演示了如何使用 `unsubscribe-from-event`。

### AWS CLI

#### 取消订阅活动

以下`unsubscribe-from-event`命令禁止向主题发送有关该事件的 Amazon SNS 通知ASSESSMENT\_RUN\_COMPLETED的过程，ARN 为：`arn:aws:sns:us-west-2:123456789012:exampletopic`

```
aws inspector unsubscribe-from-event --event ASSESSMENT_RUN_COMPLETED --resource-arn
arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX/template/0-7sbz2Kz0 --
topic arn:aws:sns:us-west-2:123456789012:exampletopic
```

有关更多信息，请参阅 Amazon Inspector 指南中的 Amazon Inspector 评估模板和评估运行。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UnsubscribeFromEvent](#)中的。

## update-assessment-target

以下代码示例演示了如何使用 `update-assessment-target`。

### AWS CLI

#### 更新评估目标

以下`update-assessment-target`命令使用的 ARN `arn:aws:inspector:us-west-2:123456789012:target/0-nvgVhaxX` 和名称更新评估目标Example，使用 ARN 更新资源组：`arn:aws:inspector:us-west-2:123456789012:resourcegroup/0-yNbgL5Pt`

```
aws inspector update-assessment-target --assessment-target-arn arn:aws:inspector:us-
west-2:123456789012:target/0-nvgVhaxX --assessment-target-name Example --resource-
group-arn arn:aws:inspector:us-west-2:123456789012:resourcegroup/0-yNbgL5Pt
```

有关更多信息，请参阅 Amazon Inspector 指南中的亚马逊检查员评估目标。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAssessmentTarget](#)中的。

## AWS IoT 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS IoT。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过在同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **accept-certificate-transfer**

以下代码示例演示了如何使用 accept-certificate-transfer。

AWS CLI

接受从其他 AWS 账户转移的设备证书

以下accept-certificate-transfer示例接受从其他 AWS 账户转移的设备证书。证书由其 ID 标识。

```
aws iot accept-certificate-transfer \  
  --certificate-id  
  488b6a7f2acdeb00a77384e63c4e40b18bEXAMPLEEe57b7272ba44c45e3448142
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Core 开发者指南》中的将[证书转移到其他账户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AcceptCertificateTransfer](#)中的。

### **add-thing-to-billing-group**

以下代码示例演示了如何使用 add-thing-to-billing-group。

## AWS CLI

### 示例 1：按名称向账单组添加事物

以下add-thing-to-billing-group示例将名为的事物MyLightBulb添加到名为的账单组GroupOne。

```
aws iot add-thing-to-billing-group \  
  --billing-group-name GroupOne \  
  --thing-name MyLightBulb
```

此命令不生成任何输出。

### 示例 2：通过 ARN 将事物添加到账单组

以下add-thing-to-billing-group示例将具有指定 ARN 的事物添加到具有指定 ARN 的账单组。如果您使用多个 AWS 地区或账户，则指定 ARN 会很有帮助。它可以帮助确保您添加到正确的地区和账户。

```
aws iot add-thing-to-thing-group \  
  --billing-group-arn "arn:aws:iot:us-west-2:123456789012:billinggroup/GroupOne" \  
  --thing-arn "arn:aws:iot:us-west-2:123456789012:thing/MyOtherLightBulb"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[账单组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddThingToBillingGroup](#)中的。

## add-thing-to-thing-group

以下代码示例演示了如何使用 add-thing-to-thing-group。

## AWS CLI

### 向群组添加事物

以下add-thing-to-thing-group示例将指定的事物添加到指定的事物组。

```
aws iot add-thing-to-thing-group \  
  --thing-name MyLightBulb \  
  --thing-group-name LightBulbs
```



此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[事物组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddThingToThingGroup](#)中的。

## associate-targets-with-job

以下代码示例演示了如何使用 `associate-targets-with-job`。

### AWS CLI

将事物组与连续作业关联

以下`associate-targets-with-job`示例将指定的事物组与指定的连续任务相关联。

```
aws iot associate-targets-with-job \  
  --targets "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs" \  
  --job-id "example-job-04"
```

输出：

```
{  
  "jobArn": "arn:aws:iot:us-west-2:123456789012:job/example-job-04",  
  "jobId": "example-job-04",  
  "description": "example continuous job"  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[创建和管理作业 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateTargetsWithJob](#)中的。

## attach-policy

以下代码示例演示了如何使用 `attach-policy`。

### AWS CLI

示例 1：将策略附加到事物组

以下`attach-policy`示例将指定的策略附加到由其 ARN 标识的事物组。

```
aws iot attach-policy \  
  --policy-name "example-policy"
```

```
--target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs" \  
--policy-name "UpdateDeviceCertPolicy"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[事物组](#)。

示例 2：将策略附加到证书

以下attach-policy示例将策略附加UpdateDeviceCertPolicy到证书指定的委托人。

```
aws iot attach-policy \  
  --policy-name UpdateDeviceCertPolicy \  
  --target "arn:aws:iot:us-  
west-2:123456789012:cert/4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的“[将AWS 物联网策略附加到设备证书](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachPolicy](#)中的。

## attach-security-profile

以下代码示例演示了如何使用 attach-security-profile。

### AWS CLI

将安全配置文件与所有未注册的设备关联

以下attach-security-profile示例将名为的 AWS IoT Device Defender 安全配置文件Testprofile与该 AWS 账户us-west-2在该地区的所有未注册设备相关联。

```
aws iot attach-security-profile \  
  --security-profile-name Testprofile \  
  --security-profile-target-arn "arn:aws:iot:us-west-2:123456789012:all/  
unregistered-things"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachSecurityProfile](#)中的。

## attach-thing-principal

以下代码示例演示了如何使用 `attach-thing-principal`。

### AWS CLI

为你的东西附加证书

以下 `attach-thing-principal` 示例将证书附加到 `MyTemperatureSensor` 事物。该证书由 ARN 标识。您可以在物 AWS 联网控制台中找到证书的 ARN。

```
aws iot attach-thing-principal \  
  --thing-name MyTemperatureSensor \  
  --principal arn:aws:iot:us-  
west-2:123456789012:cert/2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT 开发人员指南》中的 [如何使用注册表管理事物](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AttachThingPrincipal](#) 中的。

## cancel-audit-mitigation-actions-task

以下代码示例演示了如何使用 `cancel-audit-mitigation-actions-task`。

### AWS CLI

取消审计缓解措施任务

以下 `cancel-audit-mitigation-actions-task` 示例取消了对指定任务应用的缓解操作。您无法取消已经完成的任務。

```
aws iot cancel-audit-mitigation-actions-task  
  --task-id "myActionsTaskId"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [CancelAuditMitigationActionsTask \(缓解操作命令\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CancelAuditMitigationActionsTask](#) 中的。

## cancel-audit-task

以下代码示例演示了如何使用 `cancel-audit-task`。

### AWS CLI

#### 取消审计任务

以下`cancel-audit-task`示例取消了具有指定任务 ID 的审计任务。您无法取消已完成的任务。

```
aws iot cancel-audit-task \  
  --task-id a3aea009955e501a31b764abe1bebd3d
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[审计命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelAuditTask](#)中的。

## cancel-certificate-transfer

以下代码示例演示了如何使用 `cancel-certificate-transfer`。

### AWS CLI

#### 要取消将证书转移到其他 AWS 账户

以下`cancel-certificate-transfer`示例取消了指定证书转移的转移。证书由证书 ID 标识。您可以在 AWS 物联网控制台中找到证书的 ID。

```
aws iot cancel-certificate-transfer \  
  --certificate-id  
  f0f33678c7c9a046e5cc87b2b1a58dfa0beec26db78add5e605d630e05c7fc8
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Core 开发者指南》中的将[证书转移到其他账户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelCertificateTransfer](#)中的。

## cancel-job-execution

以下代码示例演示了如何使用 `cancel-job-execution`。

## AWS CLI

### 取消在设备上执行的任务

以下`cancel-job-execution`示例取消了在设备上执行指定作业。如果作业未处于QUEUED状态，则必须添加`--force`参数。

```
aws iot cancel-job-execution \  
  --job-id "example-job-03" \  
  --thing-name "MyRPi"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[创建和管理作业 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelJobExecution](#)中的。

## cancel-job

以下代码示例演示了如何使用 `cancel-job`。

## AWS CLI

### 取消任务

以下`cancel-job`示例取消了指定的作业。

```
aws iot cancel-job \  
  --job-id "example-job-03"
```

输出：

```
{  
  "jobArn": "arn:aws:iot:us-west-2:123456789012:job/example-job-03",  
  "jobId": "example-job-03",  
  "description": "example job test"  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[创建和管理作业 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelJob](#)中的。

## clear-default-authorizer

以下代码示例演示了如何使用 `clear-default-authorizer`。

### AWS CLI

#### 清除默认授权方

以下 `clear-default-authorizer` 示例清除当前配置的默认自定义授权方。运行此命令后，就没有默认授权者了。使用自定义授权方时，必须在 HTTP 请求标头中按名称进行指定。

```
aws iot clear-default-authorizer
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT API 参考 [ClearDefaultAuthorizer](#) 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ClearDefaultAuthorizer](#) 中的。

## confirm-topic-rule-destination

以下代码示例演示了如何使用 `confirm-topic-rule-destination`。

### AWS CLI

#### 确认主题规则目的地

以下 `confirm-topic-rule-destination` 示例使用在 HTTP 端点收到的确认令牌来确认主题规则目标。

```
aws iot confirm-topic-rule-destination \  
  --confirmation-token "AYADeIcmtq-  
ZkxfpiWlQqHWM5ucAXwABABVhd3MtY3J5cHRvLXB1YmxpYy1rZXkAREFyY1E0UmlGeDg0V21BZWZ1VjZtZWFRVUJJUkt  
aywpPqg8YEsa11D4B40aJ2s1wEHKMybiF1Ro0ZzYisI0IvslzQY5UmCkqq3tV-3f7-  
nKfosgIAAAAAADAAAEEAAAAAAAAAAAAAAAAAAAAA9RMgy-  
V19V9m6Iw2xfbw_____wAAAAEAAAAAAAAAAAAAAAAAAEAAAB1hw4SokgUcxiJ3gT06n50NLJVpzyQR1UmPIj5sShqXEQGcC  
iufgrzTePl8RZY0Wr006Aj9DiVzJZx-1iD6Pu-  
G6PUw1ka07Knzs2B4AD0qfrHUF4pYRTvyUgBnMGUCMQC8ZRmhKqntd_c6Kgrow3bMUDbvNqo2qZr8Z8Jm2rzgseR01An  
PIetJ803Z4I1I1F8xX1cdPGP-PV1d0XFemyL8g"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [确认主题规则目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ConfirmTopicRuleDestination](#)中的。

## create-audit-suppression

以下代码示例演示了如何使用 create-audit-suppression。

### AWS CLI

#### 创建审计结果抑制功能

以下create-audit-suppression示例为被标记为过于宽松的名为“virtualMachinePolicy”的策略创建了审计结果抑制功能。

```
aws iot create-audit-suppression \  
  --check-name IOT_POLICY_OVERLY_PERMISSIVE_CHECK \  
  --resource-identifier  
  policyVersionIdentifier={"policyName"="virtualMachinePolicy","policyVersionId"="1"} \  
 \  
  --no-suppress-indefinitely \  
  --expiration-date 2020-10-20
```

此命令不生成任何输出。

有关更多信息，请参阅《物AWS 联网开发人员指南》中的[“审计发现抑制”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAuditSuppression](#)中的。

## create-authorizer

以下代码示例演示了如何使用 create-authorizer。

### AWS CLI

#### 创建自定义授权方

以下create-authorizer示例创建了一个使用指定的 Lambda 函数作为自定义身份验证服务一部分的自定义授权方。

```
aws iot create-authorizer \  
  --authorizer-name "CustomAuthorizer" \  
  --authorizer-function-arn "arn:aws:lambda:us-  
west-2:123456789012:function:CustomAuthorizerFunction" \  
 \  
  --no-suppress-indefinitely
```

```

--token-key-name "MyAuthToken" \
--status ACTIVE \
--token-signing-public-keys FIRST_KEY="-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAA0CAQ8AMIIBCgKCAQEA1uJ0B4lQPgG/lM6ZfIwo
Z+7ENxAio9q6QD4FFqjGZsvjtYwjoe1RKK0U8Eq9xb503kRSmyIwTzwzm/f4Gf0Y
ZUloJ+t3PUUwHrmbYTAgrCUgRFygfjfgVwGCPs5ZAX4Eyqt5cr+AIHIiUDbxSa7p
zwOBKPeic0asNJpqT8PkBbRaKylEJh5oo81NDHHmVtbBm5A5YiJjqYXLaVAowKzZ
+GqsNvAQ9Jy1wI2VrEa10fL8fLDB/BJLm7zjpfP0HDJQgID0XnZwAlNnZc0hCwIx
50g2LW20y9R/dmqtDmJiVP97Z4GykxPvw1YHrUXY0iW1R3AR/Ac1NhCTGZMwVDB1
LQIDAQAB
-----END PUBLIC KEY-----"

```

输出：

```

{
  "authorizerName": "CustomAuthorizer",
  "authorizerArn": "arn:aws:iot:us-west-2:123456789012:authorizer/
CustomAuthorizer2"
}

```

有关更多信息，请参阅 AWS IoT API 参考[CreateAuthorizer](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateAuthorizer](#)中的。

## create-billing-group

以下代码示例演示了如何使用 create-billing-group。

AWS CLI

创建账单组

以下 create-billing-group 示例创建了一个名为的简单账单组 GroupOne。

```

aws iot create-billing-group \
--billing-group-name GroupOne

```

输出：

```

{
  "billingGroupName": "GroupOne",
  "billingGroupArn": "arn:aws:iot:us-west-2:123456789012:billinggroup/GroupOne",
}

```



```
"billingGroupId": "103de383-114b-4f51-8266-18f209ef5562"
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[账单组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateBillingGroup](#)中的。

## create-certificate-from-csr

以下代码示例演示了如何使用 `create-certificate-from-csr`。

### AWS CLI

通过证书签名请求 (CSR) 创建设备证书

以下`create-certificate-from-csr`示例从 CSR 创建设备证书。您可以使用`openssl`命令创建 CSR。

```
aws iot create-certificate-from-csr \
  --certificate-signing-request=file://certificate.csr
```

输出：

```
{
  "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
c0c57bbc8baaf4631a9a0345c957657f5e710473e3ddbbee1428d216d54d53ac9",
  "certificateId":
  "c0c57bbc8baaf4631a9a0345c957657f5e710473e3ddbbee1428d216d54d53ac9",
  "certificatePem": "<certificate-text>"
}
```

有关更多信息，请参阅《AWS 物联网 API 参考》中的 [CreateCertificateFromCSR](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCertificateFromCsr](#)中的。

## create-custom-metric

以下代码示例演示了如何使用 `create-custom-metric`。

### AWS CLI

创建由您的设备发布到 Device Defender 的自定义指标

以下`create-custom-metric`示例创建了一个用于衡量电池电量百分比的自定义指标。

```
aws iot create-custom-metric \  
  --metric-name "batteryPercentage" \  
  --metric-type "number" \  
  --display-name "Remaining battery percentage." \  
  --region us-east-1 \  
  --client-request-token "02ccb92b-33e8-4dfa-a0c1-35b181ed26b0"
```

输出：

```
{  
  "metricName": "batteryPercentage",  
  "metricArn": "arn:aws:iot:us-east-1:1234564789012:custommetric/  
batteryPercentage"  
}
```

有关更多信息，请参阅 AWS IoT Core 开发者指南中的[自定义指标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCustomMetric](#)中的。

## create-dimension

以下代码示例演示了如何使用 `create-dimension`。

### AWS CLI

#### 创建维度

以下内容使用名为的单个主题筛选器`create-dimension`创建维度`TopicFilterForAuthMessages`。

```
aws iot create-dimension \  
  --name TopicFilterForAuthMessages \  
  --type TOPIC_FILTER \  
  --string-values device/+/auth
```

输出：

```
{  
  "name": "TopicFilterForAuthMessages",  
  "arn": "arn:aws:iot:eu-west-2:123456789012:dimension/TopicFilterForAuthMessages"
```

```
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDimension](#)中的。

## create-domain-configuration

以下代码示例演示了如何使用 create-domain-configuration。

### AWS CLI

#### 创建域配置

以下create-domain-configuration示例创建了一个服务类型为的 AWS托管域配置。DATA

```
aws iot create-domain-configuration \  
  --domain-configuration-name "additionalDataDomain" \  
  --service-type "DATA"
```

输出：

```
{  
  "domainConfigurationName": "additionalDataDomain",  
  "domainConfigurationArn": "arn:aws:iot:us-  
west-2:123456789012:domainconfiguration/additionalDataDomain/dikMh"  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[可配置终端节点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDomainConfiguration](#)中的。

## create-dynamic-thing-group

以下代码示例演示了如何使用 create-dynamic-thing-group。

### AWS CLI

#### 创建动态事物组

以下create-dynamic-thing-group示例创建了一个动态事物组，其中包含温度属性大于 60 度的任何事物。必须先启用 AWS IoT 队列索引，然后才能使用动态事物组。

```
aws iot create-dynamic-thing-group \  
  --thing-group-name "RoomTooWarm" \  
  --query-string "attributes.temperature>60"
```

输出：

```
{  
  "thingGroupName": "RoomTooWarm",  
  "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RoomTooWarm",  
  "thingGroupId": "9d52492a-fc87-43f4-b6e2-e571d2ffcad1",  
  "indexName": "AWS_Things",  
  "queryString": "attributes.temperature>60",  
  "queryVersion": "2017-09-30"  
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[动态事物组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDynamicThingGroup](#)中的。

## create-job

以下代码示例演示了如何使用 create-job。

### AWS CLI

#### 示例 1：创建作业

以下create-job示例创建了一个向MyRaspberryPi设备发送 JSON 文档的简单 AWS 物联网作业。

```
aws iot create-job \  
  --job-id "example-job-01" \  
  --targets "arn:aws:iot:us-west-2:123456789012:thing/MyRaspberryPi" \  
  --document file://example-job.json \  
  --description "example job test" \  
  --target-selection SNAPSHOT
```

输出：

```
{  
  "jobArn": "arn:aws:iot:us-west-2:123456789012:job/example-job-01",
```

```
"jobId": "example-job-01",
"description": "example job test"
}
```

## 示例 2：创建连续作业

以下 `create-job` 示例创建了一个作业，该作业将在指定为目标的任务完成后继续运行。在此示例中，目标是一个事物组，因此，当向该组中添加新设备时，连续作业将在这些新事物上运行。

```
aws iot create-job --job-id "example-job-04" --目标 "arn:aws:iot:us-west-2:123456789012:thinggroup/" --document file://example-job.json --描述 "示例连续作业" --目标选择连续 DeadBulbs
```

输出：

```
{
  "jobArn": "arn:aws:iot:us-west-2:123456789012:job/example-job-04",
  "jobId": "example-job-04",
  "description": "example continuous job"
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [创建和管理作业 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateJob](#) 中的。

## create-keys-and-certificate

以下代码示例演示了如何使用 `create-keys-and-certificate`。

### AWS CLI

#### 创建 RSA 密钥对并颁发 X.509 证书

以下内容 `create-keys-and-certificate` 创建 2048 位 RSA 密钥对，并使用已颁发的公钥颁发 X.509 证书。由于这是 AWS 物联网唯一一次为此证书提供私钥，因此请务必将其保存在安全的地方。

```
aws iot create-keys-and-certificate \
  --certificate-pem-outfile "myTest.cert.pem" \
  --public-key-outfile "myTest.public.key" \
  --private-key-outfile "myTest.private.key"
```

输出：

```
{
  "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
  "certificateId":
  "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
  "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAKGA1UEBhMC
VVMxCzAJBgNVBAgEXAMPLEAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAStC01BTSEXAMPLE2x1MRIwEAYDVQQDEw1UZXR0Q21sYW1xHmAd
BgkqhkiG9w0BCQEWEG5vb251QGftYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAKGA1UEBhMCEXAMPLEJBgNVBAgTAldBMRawDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDAAEXAMPLEsTC01BTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYW1xHmAdBgkqhkiG9w0BCQEXAMPLE251QGft
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZEXAMPLEELG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzSzwY6786m86gpE
Ibb30hjZncvcQAEXAMPLEEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUHVvXyUntneD9+h8Mg9qEXAMPLEEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFbjvSfpJI1J00zbhNYS5f6GuoEDEXAMPLEBHjJnyp3780D8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
  "keyPair": {
    "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h\nMMEXAMPLEEuuN/
dMAS3fyce8DW/4+EXAMPLEYjmoF/YVF/gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y+jikqX0gHh/
xJTtwo
+sGpWEXAMPLEDz18x0d2ka4tCzuWEXAMPLEEahJbYkCPUBSU8opVkr7qkEXAMPLE1DR6sx2Hocli00Ltu6Fkw91swQWEX
\nGB3ZPrNh0PzQYvjUSTzEcycyNCx2EXAMPLEEv9mQ0UXP6p1fgxwKRX2fEXAMPLEDa
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEcW+LyFhI5mgFR188eGdsAEXAMPLE1nI9EesG\nfQIDAQAB\n-----
END PUBLIC KEY-----\n",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omitted for security
reasons\n-----END RSA PRIVATE KEY-----\n"
  }
}
```

有关更多信息，请参阅 [《物 AWS 联网开发者指南》](#) 中的 [创建和注册物 AWS 联网设备证书](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateKeysAndCertificate](#) 中的。

## create-mitigation-action

以下代码示例演示了如何使用 create-mitigation-action。

## AWS CLI

### 创建缓解措施

以下create-mitigation-action示例定义了一个名为的缓解操作，应用AddThingsToQuarantineGroup1Action该操作后，会将事物移动到名为的事物组中QuarantineGroup1。此操作将覆盖动态事物组。

```
aws iot create-mitigation-action --cli-input-json file::params.json
```

params.json 的内容：

```
{
  "actionName": "AddThingsToQuarantineGroup1Action",
  "actionParams": {
    "addThingsToThingGroupParams": {
      "thingGroupNames": [
        "QuarantineGroup1"
      ],
      "overrideDynamicGroups": true
    }
  },
  "roleArn": "arn:aws:iam::123456789012:role/service-role/MoveThingsToQuarantineGroupRole"
}
```

输出：

```
{
  "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/AddThingsToQuarantineGroup1Action",
  "actionId": "992e9a63-a899-439a-aa50-4e20c52367e1"
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [CreateMitigationAction \(缓解操作命令\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateMitigationAction](#)中的。

### create-ota-update

以下代码示例演示了如何使用 create-ota-update。

## AWS CLI

### 创建用于亚马逊 FreeRTOS 的 OTA 更新

以下 `create-ota-update` 示例在目标 AWS 物组或群组上创建 IoT OtaUpdate。这是 Amazon FreeRTOS over-the-air 更新的一部分，通过该更新，您可以将新的固件映像部署到单个设备或一组设备上。

```
aws iot create-ota-update \  
  --cli-input-json file://create-ota-update.json
```

`create-ota-update.json` 的内容：

```
{  
  "otaUpdateId": "ota12345",  
  "description": "A critical update needed right away.",  
  "targets": [  
    "device1",  
    "device2",  
    "device3",  
    "device4"  
  ],  
  "targetSelection": "SNAPSHOT",  
  "awsJobExecutionsRolloutConfig": {  
    "maximumPerMinute": 10  
  },  
  "files": [  
    {  
      "fileName": "firmware.bin",  
      "fileLocation": {  
        "stream": {  
          "streamId": "004",  
          "fileId": 123  
        }  
      },  
      "codeSigning": {  
        "awsSignerJobId": "48c67f3c-63bb-4f92-a98a-4ee0fbc2bef6"  
      }  
    }  
  ]  
  "roleArn": "arn:aws:iam:123456789012:role/service-role/my_ota_role"  
}
```



输出：

```
{
  "otaUpdateId": "ota12345",
  "awsIotJobId": "job54321",
  "otaUpdateArn": "arn:aws:iot:us-west-2:123456789012:otaupdate/itsaupdate",
  "awsIotJobArn": "arn:aws:iot:us-west-2:123456789012:job/itsajob",
  "otaUpdateStatus": "CREATE_IN_PROGRESS"
}
```

有关更多信息，请参阅《物联网 API 参考》中的 [createOtaUpdate](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateOtaUpdate](#)中的。

## create-policy-version

以下代码示例演示了如何使用 create-policy-version。

### AWS CLI

#### 使用新版本更新政策

以下create-policy-version示例更新了策略定义，创建了新的策略版本。此示例还将新版本设为默认版本。

```
aws iot create-policy-version \
  --policy-name UpdateDeviceCertPolicy \
  --policy-document file://policy.json \
  --set-as-default
```

policy.json 的内容：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:UpdateCertificate",
      "Resource": "*"
    }
  ]
}
```

输出：

```
{
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/UpdateDeviceCertPolicy",
  "policyDocument": "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Effect\": \"Allow\", \"Action\": \"iot:UpdateCertificate\", \"Resource\": \"*\" } ] }",
  "policyVersionId": "2",
  "isDefaultVersion": true
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》中的AWS IoT 政策](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePolicyVersion](#)中的。

## create-policy

以下代码示例演示了如何使用 create-policy。

### AWS CLI

#### 创建 AWS 物联网策略

以下create-policy示例创建了一个名为的 AWS IoT 策略 TemperatureSensorPolicy。该policy.json文件包含允许 AWS IoT 策略操作的语句。

```
aws iot create-policy \
  --policy-name TemperatureSensorPolicy \
  --policy-document file://policy.json
```

policy.json 的内容：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:123456789012:topic/topic_1",

```

```

        "arn:aws:iot:us-west-2:123456789012:topic/topic_2"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "iot:Subscribe"
    ],
    "Resource": [
        "arn:aws:iot:us-west-2:123456789012:topicfilter/topic_1",
        "arn:aws:iot:us-west-2:123456789012:topicfilter/topic_2"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "iot:Connect"
    ],
    "Resource": [
        "arn:aws:iot:us-west-2:123456789012:client/basicPubSub"
    ]
}
]
}
}

```

输出：

```

{
  "policyName": "TemperatureSensorPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
TemperatureSensorPolicy",
  "policyDocument": "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
      {
        \"Effect\": \"Allow\",
        \"Action\": [
          \"iot:Publish\",
          \"iot:Receive\"
        ],
        \"Resource\": [
          \"arn:aws:iot:us-west-2:123456789012:topic/topic_1\",
          \"arn:aws:iot:us-west-2:123456789012:topic/topic_2\"
        ]
      }
    ]
  }"
}

```

```

    ]
  },
  {
    \"Effect\": \"Allow\",
    \"Action\": [
      \"iot:Subscribe\"
    ],
    \"Resource\": [
      \"arn:aws:iot:us-west-2:123456789012:topicfilter/topic_1\",
      \"arn:aws:iot:us-west-2:123456789012:topicfilter/topic_2\"
    ]
  },
  {
    \"Effect\": \"Allow\",
    \"Action\": [
      \"iot:Connect\"
    ],
    \"Resource\": [
      \"arn:aws:iot:us-west-2:123456789012:client/basicPubSub\"
    ]
  }
]
}],
\"policyVersionId\": \"1\"
}

```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》中的AWS IoT 政策](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePolicy](#)中的。

## create-provisioning-claim

以下代码示例演示了如何使用 create-provisioning-claim。

### AWS CLI

#### 创建预配声明

以下create-provisioning-claim示例根据配置模板创建置备声明。

```
aws iot create-provisioning-claim \
  --template-name MyTestProvisioningTemplate
```

输出：

```
{
  "certificateId":
    "78de02184b2ce80cf8fb709bda59e62b19fb83513590483eb0434589476ab09f",
  "certificatePem": "-----BEGIN CERTIFICATE-----\nMIIDdzCCA1
+gAwIBAgIUXSZHEBLztMLZ2fHG
14gV0NymYY0wDQYJKoZIhvcNAQEL
\nBQAwfjELMAkGA1UEBhMCVVMxEzARBgNVBAgMC1dhc2hpbmd0b24xEDA0Bg
VBAcM\nB1NlYXR0bGUxGDAWBgNVBAoMD0FtYXpvcvi5jb20gSW5jLjEgMB4GA1UECwwXQW1h
\nem9uIElVVCBQcm9
2aXNpb25pbmcxDDAKBgNVBAUTAzEuMDAeFw0yMDA3Mjg0NjQ0\nMDZaFw0yMDA3Mjg0NjUxMDZaMEsxBHBBGVB
AMMQDFhNDEyM2VkNmIxYjU3MzE3\nZTgzMTJmY2MzN2FiNTdhY2MzYTZkZGVjOGQ5OGY3NzUwMWR1Mjc0YjhmYTQ
xN2Iw\nnggEiMA0GCSqGSIb3EXAMPLEAA4IBDwAwggEKAoIBAQBDBhKI94ktKLqTwnj+ay0q1\nTAJt/
N6s6IJDZv1
rYjkC0E7wzaeY3TprWk03S29vUzVuE0XHXQXZbihgpg2m6fza\nkwm9/
wpjzE9ny5+xkPGVH4Wnwz7yK5m8S0agL
T96cRBSWnWmon0WdY0GKVzni0CA\n+iyGudgrFKm7Eae/
v18oXrf82Kt0AG04xG0KE2WKYHsT1fx3c9xZh1XP/eX
Lhv00\n+1Gp0WVw9PbhKfrxliKJ5q6sL5nVUaUHq6h1QPYwsATe0vAp3u0ak5zgTyL0fg7Y
\nPyKk6VYwLW62r+V
YBSForEM0Ahkq3LsP/rjxpEKmi2W41PVS6oFZRKcD+H1Kyil5\nAgMBAAGjIDAeMAwGA1UdEwEB/
wQMAAwDgYDV
R0PAQH/BAQDAgeAMA0GCSqGSIb3\nDQEBcWUAA4IBAQAQgix2k6nVqbZFKq97/fZBzLGS0dyz5rT/
E41cDIRX+1j
EPW41\nnw0D+2sXheCZLZZnSkvIiP74IToNeXDrdcaodeGFVHIElRjhMIq+4ZebPbRLtidF
\nRc2hfcTAlqq9Z6v
5Vk6BeM1tu0RqH1wPoVUccLPya8EjNCbnJZUmGd0frN/Y9pho\n5ikV+HPeZhG/k6dhE2GsQJyKFVHL/
uBgKSily
1bRyWU1r6qcpWBNBHjUoD7Hg0wD
\nnzMh4XRb2FQDsqFalKCSYmeL8IVC49sgPD90typ5uteGMTy62usAAUQdq/f
ZvrWg\n0kFpwMVnGKVKT7Kq0kK0LzKw0BB2Jm4/gmrJ\n-----END CERTIFICATE-----\n",
  "keyPair": {
    "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCg
KCAQEAWSiPeJLSi6k8J4/msjq
\nUwCbfzer0iCQ2b5a2I5AtB08M2nmN06a1pNN0tvb1M1bhDlx10F2W4oYKYN
pun8\n2pFpVf8KY8xPZ8ufsZDx1R+FP8M+8iuZvEtGoC0/enEQUl1pqJzlnWNBilc54tA
\nngPoshrnYKxSpuxGn
v79fKF63/NirTgBjuMRtChNlimEXAMPLE3PcWYZVz/3ly4b9\nNPPRqdf1cPT24Sn68ZYiieaurC
+Z1VG1B6uoZU
D2MLAE3jrwKd7tGp0c4E8i9H40\n2D8ip0lWMC1utq/
lWAUhaKxDDgIZKty7D/648aRCpotluJT1UuqBWUSnA/h9
Ssop\nneQIDAQAB\n-----END PUBLIC KEY-----\n",
```

```

    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIIEowIBAAKCAQEAWYSiPeJLSi6k8J4/
msjqtUwCbfzer0iCQ2b5a2I5AtB08M2n
\nmN06a1pNN0tvb1M1bhDlx10F2W4oYKYNpun82pFpvf8KY8xPZ8ufsz
Dx1R+Fp8M+\n8iuZvEtGoC0/enEQUl1pqJz1nWNBilc54tAgPoshrnYKxSpuxGnv79fKF63/Nir
\nTgBjuMRtCh
NlimB7E9X8d3PcWYZVz/3ly4b9NPPRqdFlcPT24Sn68ZYiieaurC+Z
\n1VGLB6uoZUD2MLAE3jrwKd7tGp0c4E8i
9H402D8ip0lWMC1utq/1WAUhaKxDDgIZ\nKty7D/648aRCpotluJT1UuqBWUSnA/
h9SsopeQIDAQABAoIBAEAybn
QUtx9T2/nK\nntZT2pA4iugecxI4dz+DmT0XVXs5VJmrx/
nBSq6ejXExEpSIM04RY7LE3ZdJcnd56\nF7tQkkY7yR
VzfxHeXFU1kr0IPuxWebN0rRoPZr+1RSer+ww2aBC525+88pVuR6tM
\nm3pgkrR2ycCj9Fd0UoQxdjHBHaM5PDMj
9aSxCKdg3nReepeGwsR2TQA+m2vVxWk7\nnou0+91eTOP+/QfP7P8Zj0Ik02Xiv1RcVDyN/
E4QXPKuIkM/8vS8VK+
E9pATQ0MtB\n21w8R/YU5AJd6j1EXAMPLEGU2UzRzInNWiltkPPPqgqXXhx0f+mxByjcMa1VJk0L
\nh0G2R0UCgY
EA+R0cHNHy/XbsP7Fih0hEh+6Q2QxQ2ncBUPYbBazrR8Hn+7SCICQK
\nVyYfd8Ajfq3e7RsKVL5S1MBp7S1idxak
bIn28fKfPn62DaemGCIOyDgLf+eUxBx
\nngzbCiBZga8brfurza43UZjKZLpg3hq721+FeAiXi1Nma4Yr9YWEHEN
8CgYEAxuwT\nnpzdWwmsiFzfsAw0sy9ySDA/xr5WRWzJyAqUsjsks6rxNzWebpufnYHcmtW7pLdqM
\nkboHwN2pXa
kmZvrk2nKkEMq5brBYGDxuxDe+V369Bianx8aZFyIsckA70wXW1w1h
\nngRC5rQ4X0gp3+Jmw7eA08LRYDjaN846+
Qbt02KcCgYAWS0UL51bijQR0ZwI0dz27\nnFQVuCAYsp748aurcRTACCj8jbnK/
QbqTNlxWsaH7ssBjZko2D5sAqY
BRtASW0Dab\naHXsDhVm2Jye+ESLoHMaCLoyCkT3118yqXicEDStM07f01Ryag164EiJvSiRmfny\nnNL/
fXVjCSH
/udCxdzPt+7QKBgQC+LAD7rxdr4J9538hTqpc4XK9vxRbrMXEH55XH
\nHbMa2x0NZXpmeTgEQBukyohCVceyRhK9
i0e6irZTjVXgh0eoTpC8VXkzcnzouTiQ
\nnFQQSGfnp7Ioe6UIz23715pKduszSnkMSKrG924ktv7CyDBF1gBQI5g
aDoHnddJBJ\nnPRtIZQKBgA8MASxtTxQntRwXXzR92U0vAighiuRkB/mx9jQpUcK1qiqHbkAMqgNF
\nPFCBYIUbFT
iYKKKeJNbyJQvjfsJcKAnaFJ+RnTxk0Q6Wjm20peJ/ii4QiDdnigoE\nnvd1c5cFQewWb4/
zqAtPdinkPLN94ileI
79XQdc7R1J0jpgTimL+V\n-----END RSA PRIVATE KEY-----\n"
    },
    "expiration": 1595955066.0
}

```

有关更多信息，请参阅《AWS IoT Core 开发人员指南》中的[可信用户配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateProvisioningClaim](#) 中的。

## create-provisioning-template-version

以下代码示例演示了如何使用 create-provisioning-template-version。

### AWS CLI

#### 创建预配模板版本

以下示例为指定的配置模板创建了一个版本。文件中提供了新版本的正文 template.json。

```
aws iot create-provisioning-template-version \  
  --template-name widget-template \  
  --template-body file://template.json
```

template.json 的内容：

```
{  
  "Parameters" : {  
    "DeviceLocation": {  
      "Type": "String"  
    }  
  },  
  "Mappings": {  
    "LocationTable": {  
      "Seattle": {  
        "LocationUrl": "https://example.aws"  
      }  
    }  
  },  
  "Resources" : {  
    "thing" : {  
      "Type" : "AWS::IoT::Thing",  
      "Properties" : {  
        "AttributePayload" : {  
          "version" : "v1",  
          "serialNumber" : "serialNumber"  
        },  
        "ThingName" : {"Fn::Join":["",["ThingPrefix_",  
{"Ref":"SerialNumber"}]]},
```

```

        "ThingTypeName" : {"Fn::Join":["",["ThingTypePrefix_",
{"Ref":"SerialNumber"}]]},
        "ThingGroups" : ["widgets", "WA"],
        "BillingGroup": "BillingGroup"
    },
    "OverrideSettings" : {
        "AttributePayload" : "MERGE",
        "ThingTypeName" : "REPLACE",
        "ThingGroups" : "DO_NOTHING"
    }
},
"certificate" : {
    "Type" : "AWS::IoT::Certificate",
    "Properties" : {
        "CertificateId": {"Ref": "AWS::IoT::Certificate::Id"},
        "Status" : "Active"
    }
},
"policy" : {
    "Type" : "AWS::IoT::Policy",
    "Properties" : {
        "PolicyDocument" : {
            "Version": "2012-10-17",
            "Statement": [{
                "Effect": "Allow",
                "Action":["iot:Publish"],
                "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/foo/
bar"]
            }]
        }
    }
},
"DeviceConfiguration": {
    "FallbackUrl": "https://www.example.com/test-site",
    "LocationUrl": {
        "Fn::FindInMap": ["LocationTable",{"Ref": "DeviceLocation"},
"LocationUrl"]}
    }
}
}

```

输出：



```
{
  "templateArn": "arn:aws:iot:us-east-1:123456789012:provisioningtemplate/widget-
template",
  "templateName": "widget-template",
  "versionId": 2,
  "isDefaultVersion": false
}
```

有关更多信息，请参阅《[AWS 物联网核心开发者指南](#)》中的 [Io AWS T 安全隧道](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateProvisioningTemplateVersion](#) 中的。

## create-provisioning-template

以下代码示例演示了如何使用 create-provisioning-template。

### AWS CLI

#### 创建配置模板

以下 create-provisioning-template 示例创建了文件定义的配置模板 template.json。

```
aws iot create-provisioning-template \
  --template-name widget-template \
  --description "A provisioning template for widgets" \
  --provisioning-role-arn arn:aws:iam::123456789012:role/Provision_role \
  --template-body file://template.json
```

template.json 的内容：

```
{
  "Parameters" : {
    "DeviceLocation": {
      "Type": "String"
    }
  },
  "Mappings": {
    "LocationTable": {
      "Seattle": {
        "LocationUrl": "https://example.aws"
      }
    }
  },
}
```

```

"Resources" : {
  "thing" : {
    "Type" : "AWS::IoT::Thing",
    "Properties" : {
      "AttributePayload" : {
        "version" : "v1",
        "serialNumber" : "serialNumber"
      },
      "ThingName" : {"Fn::Join":["",["ThingPrefix_",
{"Ref":"SerialNumber"}]]},
      "ThingTypeName" : {"Fn::Join":["",["ThingTypePrefix_",
{"Ref":"SerialNumber"}]]},
      "ThingGroups" : ["widgets", "WA"],
      "BillingGroup": "BillingGroup"
    },
    "OverrideSettings" : {
      "AttributePayload" : "MERGE",
      "ThingTypeName" : "REPLACE",
      "ThingGroups" : "DO_NOTHING"
    }
  },
  "certificate" : {
    "Type" : "AWS::IoT::Certificate",
    "Properties" : {
      "CertificateId": {"Ref": "AWS::IoT::Certificate::Id"},
      "Status" : "Active"
    }
  },
  "policy" : {
    "Type" : "AWS::IoT::Policy",
    "Properties" : {
      "PolicyDocument" : {
        "Version": "2012-10-17",
        "Statement": [{
          "Effect": "Allow",
          "Action":["iot:Publish"],
          "Resource": ["arn:aws:iot:us-east-1:504350838278:topic/foo/
bar"]
        }]
      }
    }
  },
  "DeviceConfiguration": {

```

```

    "FallbackUrl": "https://www.example.com/test-site",
    "LocationUrl": {
      "Fn::FindInMap": ["LocationTable", {"Ref": "DeviceLocation"}],
      "LocationUrl": []
    }
  }
}

```

输出：

```

{
  "templateArn": "arn:aws:iot:us-east-1:123456789012:provisioningtemplate/widget-template",
  "templateName": "widget-template",
  "defaultVersionId": 1
}

```

有关更多信息，请参阅 [《AWS 物联网核心开发者指南》中的 Io AWS T 安全隧道](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateProvisioningTemplate](#) 中的。

## create-role-alias

以下代码示例演示了如何使用 create-role-alias。

### AWS CLI

#### 创建角色别名

以下 create-role-alias 示例为指定角色创建名 LightBulbRole 为的角色别名。

```

aws iot create-role-alias \
  --role-alias LightBulbRole \
  --role-arn arn:aws:iam::123456789012:role/lightbulbrole-001

```

输出：

```

{
  "roleAlias": "LightBulbRole",
  "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/LightBulbRole"
}

```

有关更多信息，请参阅 AWS IoT API 参考[CreateRoleAlias](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRoleAlias](#)中的。

## create-scheduled-audit

以下代码示例演示了如何使用 create-scheduled-audit。

### AWS CLI

#### 创建计划审计

以下create-scheduled-audit示例创建了每周星期三运行的定期审计，以检查 CA 证书或设备证书是否即将过期。

```
aws iot create-scheduled-audit \  
  --scheduled-audit-name WednesdayCertCheck \  
  --frequency WEEKLY \  
  --day-of-week WED \  
  --target-check-names CA_CERTIFICATE_EXPIRING_CHECK  
  DEVICE_CERTIFICATE_EXPIRING_CHECK
```

输出：

```
{  
  "scheduledAuditArn": "arn:aws:iot:us-west-2:123456789012:scheduledaudit/  
  WednesdayCertCheck"  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[审计命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateScheduledAudit](#)中的。

## create-security-profile

以下代码示例演示了如何使用 create-security-profile。

### AWS CLI

#### 创建安全配置文件

以下create-security-profile示例创建了一个安全配置文件，用于检查蜂窝带宽是否超过阈值，或者在五分钟内是否出现超过 10 次授权失败。

```
aws iot create-security-profile \  
  --security-profile-name PossibleIssue \  
  --security-profile-description "Check to see if authorization fails 10 times in  
  5 minutes or if cellular bandwidth exceeds 128" \  
  --behaviors "[{"name":"CellularBandwidth","metric":"aws:message-byte-size  
","criteria":{"comparisonOperator":"greater-than","value":{"count":128},  
"consecutiveDatapointsToAlarm":1,"consecutiveDatapointsToClear":1}},{"name  
":"Authorization","metric":"aws:num-authorization-failures","criteria":  
{"comparisonOperator":"less-than","value":{"count":10},"durationSeconds  
":300,"consecutiveDatapointsToAlarm":1,"consecutiveDatapointsToClear":1}]"]
```

输出：

```
{  
  "securityProfileName": "PossibleIssue",  
  "securityProfileArn": "arn:aws:iot:us-west-2:123456789012:securityprofile/  
PossibleIssue"  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateSecurityProfile](#) 中的。

## create-stream

以下代码示例演示了如何使用 create-stream。

### AWS CLI

创建用于通过 MQTT 分块传送一个或多个大文件的流

以下 create-stream 示例创建了一个流，用于通过 MQTT 分块传送一个或多个大文件。流将来自 S3 之类的源的数据字节打包为 MQTT 消息，以块或数据块的形式传输。您可以将一个或多个文件与一个流关联。

```
aws iot create-stream \  
  --cli-input-json file://create-stream.json
```

create-stream.json 的内容：

```
{  
  "streamId": "stream12345",
```

```
"description": "This stream is used for Amazon FreeRTOS OTA Update 12345.",
"files": [
  {
    "fileId": 123,
    "s3Location": {
      "bucket": "codesign-ota-bucket",
      "key": "48c67f3c-63bb-4f92-a98a-4ee0fbc2bef6"
    }
  }
],
"roleArn": "arn:aws:iam:123456789012:role/service-role/my_ota_stream_role"
}
```

输出：

```
{
  "streamId": "stream12345",
  "streamArn": "arn:aws:iot:us-west-2:123456789012:stream/stream12345",
  "description": "This stream is used for Amazon FreeRTOS OTA Update 12345.",
  "streamVersion": "1"
}
```

有关更多信息，请参阅 AWS IoT API 参考[CreateStream](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateStream](#)中的。

## create-thing-group

以下代码示例演示了如何使用 create-thing-group。

### AWS CLI

#### 示例 1：创建事物组

以下 create-thing-group 示例创建了一个名为的事物组，LightBulbs 其中包含一个描述和两个属性。

```
aws iot create-thing-group \
  --thing-group-name LightBulbs \
  --thing-group-properties "thingGroupDescription=\"Generic bulb group\",
  attributePayload={attributes={Manufacturer=AnyCompany,wattage=60}}"
```

输出：

```
{
  "thingGroupName": "LightBulbs",
  "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs",
  "thingGroupId": "9198bf9f-1e76-4a88-8e8c-e7140142c331"
}
```

## 示例 2：创建属于父组的事物组

以下内容 `create-thing-group` 创建一个名为 `HalogenBulbs` 的事物组，其父事物组名为 `LightBulbs`。

```
aws iot create-thing-group \
  --thing-group-name HalogenBulbs \
  --parent-group-name LightBulbs
```

输出：

```
{
  "thingGroupName": "HalogenBulbs",
  "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/HalogenBulbs",
  "thingGroupId": "f4ec6b84-b42b-499d-9ce1-4dbd4d4f6f6e"
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[事物组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateThingGroup](#) 中的。

## create-thing-type

以下代码示例演示了如何使用 `create-thing-type`。

### AWS CLI

#### 定义事物类型

以下 `create-thing-type` 示例定义了事物类型和关联的属性。

```
aws iot create-thing-type \
  --thing-type-name "LightBulb" \
  --thing-type-properties "thingTypeDescription=light bulb type,
  searchableAttributes=wattage,model"
```

输出：

```
{
  "thingTypeName": "LightBulb",
  "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb",
  "thingTypeId": "ce3573b0-0a3c-45a7-ac93-4e0ce14cd190"
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的事物[类型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateThingType](#)中的。

## create-thing

以下代码示例演示了如何使用 create-thing。

### AWS CLI

示例 1：在注册表中创建事物记录

以下create-thing示例在 AWS IoT 事物注册表中为设备创建条目。

```
aws iot create-thing \
  --thing-name SampleIoTThing
```

输出：

```
{
  "thingName": "SampleIoTThing",
  "thingArn": "arn:aws:iot:us-west-2: 123456789012:thing/SampleIoTThing",
  "thingId": " EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE "
}
```

示例 2：定义与事物类型关联的事物

以下create-thing示例创建一个具有指定事物类型及其属性的事物。

```
aws iot create-thing \
  --thing-name "MyLightBulb" \
  --thing-type-name "LightBulb" \
  --attribute-payload '{"attributes": {"wattage": "75", "model": "123"}}'
```



输出：

```
{
  "thingName": "MyLightBulb",
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
  "thingId": "40da2e73-c6af-406e-b415-15acae538797"
}
```

有关更多信息，请参阅 [《物AWS 联网开发人员指南》](#) 中的“[如何使用注册表和事物类型管理事物](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateThing](#) 中的。

## create-topic-rule-destination

以下代码示例演示了如何使用 create-topic-rule-destination。

### AWS CLI

创建主题规则目标

以下 create-topic-rule-destination 示例为 HTTP 终端节点创建主题规则目标。

```
aws iot create-topic-rule-destination \
  --destination-configuration httpUrlConfiguration={confirmationUrl=https://
example.com}
```

输出：

```
{
  "topicRuleDestination": {
    "arn": "arn:aws:iot:us-west-2:123456789012:ruledestination/http/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "status": "IN_PROGRESS",
    "statusReason": "Awaiting confirmation. Confirmation message sent on
2020-07-09T22:47:54.154Z; no response received from the endpoint.",
    "httpUrlProperties": {
      "confirmationUrl": "https://example.com"
    }
  }
}
```

有关更多信息，请参阅 [《AWS 物联网开发者指南》](#) 中的 [创建主题规则目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTopicRuleDestination](#)中的。

## create-topic-rule

以下代码示例演示了如何使用 create-topic-rule。

### AWS CLI

#### 创建发送 Amazon SNS 警报的规则

以下create-topic-rule示例创建了一条规则，该规则用于在设备影子中发现的土壤湿度读数较低时发送 Amazon SNS 消息。

```
aws iot create-topic-rule \  
  --rule-name "LowMoistureRule" \  
  --topic-rule-payload file://plant-rule.json
```

该示例要求将以下 JSON 代码保存到名为的文件中plant-rule.json：

```
{  
  "sql": "SELECT * FROM '$aws/things/MyRPi/shadow/update/accepted' WHERE  
state.reported.moisture = 'low'\n",  
  "description": "Sends an alert whenever soil moisture level readings are too  
low.",  
  "ruleDisabled": false,  
  "awsIotSqlVersion": "2016-03-23",  
  "actions": [{  
    "sns": {  
      "targetArn": "arn:aws:sns:us-  
west-2:123456789012:MyRPiLowMoistureTopic",  
      "roleArn": "arn:aws:iam::123456789012:role/service-role/  
MyRPiLowMoistureTopicRole",  
      "messageFormat": "RAW"  
    }  
  ]  
}
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的创建AWS 物联网规则。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTopicRule](#)中的。

## delete-account-audit-configuration

以下代码示例演示了如何使用 delete-account-audit-configuration。

### AWS CLI

禁用 AWS 账户的所有审计检查

以下delete-account-audit-configuration示例恢复该账户的 AWS IoT Device Defender 的默认设置，禁用所有审计检查并清除配置数据。它还会删除对此账户的所有计划审计。请谨慎使用此命令。

```
aws iot delete-account-audit-configuration \  
  --delete-scheduled-audits
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[审计命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAccountAuditConfiguration](#)中的。

## delete-audit-suppression

以下代码示例演示了如何使用 delete-audit-suppression。

### AWS CLI

删除审计结果隐藏

以下delete-audit-suppression示例删除了  
DEVICE\_CERTIFICATE\_EXPIRICATE\_EXPIRING\_CECK

```
aws iot delete-audit-suppression \  
  --check-name DEVICE_CERTIFICATE_EXPIRING_CHECK \  
  --resource-identifier deviceCertificateId="c7691e<shortened>"
```

此命令不生成任何输出。

有关更多信息，请参阅《物AWS 联网开发人员指南》中的[“审计发现抑制”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAuditSuppression](#)中的。

## delete-authorizer

以下代码示例演示了如何使用 delete-authorizer。

### AWS CLI

#### 删除自定义授权方

以下delete-authorizer示例删除名为CustomAuthorizer的授权者。自定义授权方必须处于INACTIVE状态，然后才能将其删除。

```
aws iot delete-authorizer \  
  --authorizer-name CustomAuthorizer
```

此命令不生成任何输出。

有关更多信息，请参阅[DeleteAuthorizer](#) 《AWS 物联网开发人员指南》。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAuthorizer](#)中的。

## delete-billing-group

以下代码示例演示了如何使用 delete-billing-group。

### AWS CLI

#### 删除账单组

以下delete-billing-group示例删除了指定的账单组。即使账单组包含一项或多项内容，您也可以将其删除。

```
aws iot delete-billing-group \  
  --billing-group-name BillingGroupTwo
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[账单组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBillingGroup](#)中的。

## delete-ca-certificate

以下代码示例演示了如何使用 delete-ca-certificate。

## AWS CLI

### 删除 CA 证书

以下delete-ca-certificate示例删除具有指定证书 ID 的 CA 证书。

```
aws iot delete-ca-certificate \  
  --certificate-id  
  f4efed62c0142f16af278166f61962501165c4f0536295207426460058cd1467
```

此命令不生成任何输出。

有关更多信息，请参阅《物联网 [API 参考](#)》中的“删除证书”。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCaCertificate](#)中的。

## delete-certificate

以下代码示例演示了如何使用 delete-certificate。

## AWS CLI

### 删除设备证书

以下delete-certificate示例删除具有指定 ID 的设备证书。

```
aws iot delete-certificate \  
  --certificate-id  
  c0c57bbc8baaf4631a9a0345c957657f5e710473e3ddbbee1428d216d54d53ac9
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT API 参考[DeleteCertificate](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCertificate](#)中的。

## delete-custom-metric

以下代码示例演示了如何使用 delete-custom-metric。

## AWS CLI

### 删除自定义指标

以下delete-custom-metric示例删除了自定义指标。

```
aws iot delete-custom-metric \  
  --metric-name batteryPercentage \  
  --region us-east-1
```

输出：

```
HTTP 200
```

有关更多信息，请参阅 AWS IoT Core 开发者指南中的[自定义指标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCustomMetric](#)中的。

## delete-dimension

以下代码示例演示了如何使用 delete-dimension。

AWS CLI

删除维度

以下delete-dimension示例删除名为的维度TopicFilterForAuthMessages。

```
aws iot delete-dimension \  
  --name TopicFilterForAuthMessages
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDimension](#)中的。

## delete-domain-configuration

以下代码示例演示了如何使用 delete-domain-configuration。

AWS CLI

删除域配置

以下delete-domain-configuration示例additionalDataDomain从您的 AWS 账户中删除名为的域配置。

```
aws iot delete-domain-configuration \  
  --domain-configuration-name "additionalDataDomain" \  
  --domain-configuration-status "OK"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[可配置终端节点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDomainConfiguration](#)中的。

## delete-dynamic-thing-group

以下代码示例演示了如何使用 delete-dynamic-thing-group。

### AWS CLI

#### 删除动态事物组

以下delete-dynamic-thing-group示例删除了指定的动态事物组。

```
aws iot delete-dynamic-thing-group \  
  --thing-group-name "RoomTooWarm"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[动态事物组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDynamicThingGroup](#)中的。

## delete-job-execution

以下代码示例演示了如何使用 delete-job-execution。

### AWS CLI

#### 删除任务执行

以下delete-job-execution示例删除了设备上指定任务的执行情况。describe-job-execution用于获取执行编号。

```
aws iot delete-job-execution
  --job-id "example-job-02"
  --thing-name "MyRaspberryPi"
  --execution-number 1
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[创建和管理作业 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteJobExecution](#)中的。

## delete-job

以下代码示例演示了如何使用 delete-job。

### AWS CLI

#### 删除任务

以下delete-job示例删除了指定的作业。通过指定该--force选项，即使状态为，作业也会被删除IN\_PROGRESS。

```
aws iot delete-job \
  --job-id "example-job-04" \
  --force
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[创建和管理作业 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteJob](#)中的。

## delete-mitigation-action

以下代码示例演示了如何使用 delete-mitigation-action。

### AWS CLI

#### 删除缓解措施

以下delete-mitigation-action示例删除了指定的缓解操作。

```
aws iot delete-mitigation-action \
```



```
--action-name AddThingsToQuarantineGroup1Action
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [DeleteMitigationAction](#)（[缓解操作命令](#)）。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteMitigationAction](#) 中的。

## delete-ota-update

以下代码示例演示了如何使用 delete-ota-update。

### AWS CLI

删除 OTA 更新

以下 delete-ota-update 示例删除指定的 OTA 更新。

```
aws iot delete-ota-update \  
  --ota-update-id ota12345 \  
  --delete-stream \  
  --force-delete-aws-job
```

此命令不生成任何输出。

有关更多信息，请参阅《物联网 API 参考》中的 [DeleteUpdate](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteOtaUpdate](#) 中的。

## delete-policy-version

以下代码示例演示了如何使用 delete-policy-version。

### AWS CLI

删除策略的某个版本

以下 delete-policy-version 示例从您的 AWS 账户中删除指定策略的版本 2。

```
aws iot delete-policy-version \  
  --policy-name UpdateDeviceCertPolicy \  
  --policy-version-id 2
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS 物联网开发者指南》中的AWS IoT 策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePolicyVersion](#)中的。

## delete-policy

以下代码示例演示了如何使用 delete-policy。

### AWS CLI

#### 删除策略

以下delete-policy示例将从您的 AWS 账户中删除指定的政策。

```
aws iot delete-policy --policy-name UpdateDeviceCertPolicy
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS 物联网开发人员指南》中的AWS IoT 政策](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePolicy](#)中的。

## delete-provisioning-template-version

以下代码示例演示了如何使用 delete-provisioning-template-version。

### AWS CLI

#### 删除预配模板版本

以下delete-provisioning-template-version示例删除了指定配置模板的版本 2。

```
aws iot delete-provisioning-template-version \  
  --version-id 2 \  
  --template-name "widget-template"
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS 物联网核心开发者指南》中的Io AWS T 安全隧道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteProvisioningTemplateVersion](#)中的。

## delete-provisioning-template

以下代码示例演示了如何使用 delete-provisioning-template。

### AWS CLI

#### 删除预配模板

以下delete-provisioning-template示例删除了指定的配置模板。

```
aws iot delete-provisioning-template \  
  --template-name widget-template
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网核心开发者指南](#)》中的 [Io AWS T 安全隧道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteProvisioningTemplate](#)中的。

## delete-registration-code

以下代码示例演示了如何使用 delete-registration-code。

### AWS CLI

#### 删除您的注册码

以下delete-registration-code示例删除了特定于 I AWS oT 账户的注册码。

```
aws iot delete-registration-code
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网开发者指南](#)》中的[使用自己的证书](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRegistrationCode](#)中的。

## delete-role-alias

以下代码示例演示了如何使用 delete-role-alias。

### AWS CLI

#### 删除 Io AWS T 角色别名

以下delete-role-alias示例删除名为的 AWS IoT 角色别名LightBulbRole。

```
aws iot delete-role-alias \  
  --role-alias LightBulbRole
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[授权直接调用 AWS 服务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRoleAlias](#)中的。

## delete-scheduled-audit

以下代码示例演示了如何使用 delete-scheduled-audit。

### AWS CLI

删除计划审计

以下delete-scheduled-audit示例删除名为的 AWS IoT Device Defender 计划审计AWSIoTDeviceDefenderDailyAudit。

```
aws iot delete-scheduled-audit \  
  --scheduled-audit-name AWSIoTDeviceDefenderDailyAudit
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[审计命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteScheduledAudit](#)中的。

## delete-security-profile

以下代码示例演示了如何使用 delete-security-profile。

### AWS CLI

删除安全配置文件

以下delete-security-profile示例删除名为的安全配置文件PossibleIssue。

```
aws iot delete-security-profile \  
  --security-profile-name PossibleIssue
```

```
--security-profile-name PossibleIssue
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSecurityProfile](#)中的。

## delete-stream

以下代码示例演示了如何使用 delete-stream。

### AWS CLI

#### 删除直播

以下delete-stream示例删除了指定的直播。

```
aws iot delete-stream \  
  --stream-id stream12345
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT API 参考[DeleteStream](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteStream](#)中的。

## delete-thing-group

以下代码示例演示了如何使用 delete-thing-group。

### AWS CLI

#### 删除事物组

以下delete-thing-group示例删除了指定的事物组。如果事物组包含子事物组，则无法将其删除。

```
aws iot delete-thing-group \  
  --thing-group-name DefectiveBulbs
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[事物组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteThingGroup](#)中的。

## delete-thing-type

以下代码示例演示了如何使用 delete-thing-type。

### AWS CLI

示例 1：删除事物类型

以下delete-thing-type示例删除了已弃用的事物类型。

```
aws iot delete-thing-type \  
  --thing-type-name "obsoleteThingType"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[事物类型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteThingType](#)中的。

## delete-thing

以下代码示例演示了如何使用 delete-thing。

### AWS CLI

显示有关事物的详细信息

以下delete-thing示例从您 AWS 账户的 AWS IoT 注册表中删除一个事物。

```
aws iot 删除-thing-thing-name "" FourthBulb
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT 开发人员指南》中的[如何使用注册表管理事物](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteThing](#)中的。

## delete-topic-rule-destination

以下代码示例演示了如何使用 delete-topic-rule-destination。

## AWS CLI

### 删除主题规则目标

以下delete-topic-rule-destination示例删除了指定的主题规则目标。

```
aws iot delete-topic-rule-destination \  
  --arn "arn:aws:iot:us-west-2:123456789012:ruledestination/http/  
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[删除主题规则目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTopicRuleDestination](#)中的。

## delete-topic-rule

以下代码示例演示了如何使用 delete-topic-rule。

## AWS CLI

### 删除规则

以下delete-topic-rule示例删除了指定的规则。

```
aws iot delete-topic-rule \  
  --rule-name "LowMoistureRule"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[删除规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTopicRule](#)中的。

## delete-v2-logging-level

以下代码示例演示了如何使用 delete-v2-logging-level。

## AWS CLI

### 删除事物组的日志记录级别

以下delete-v2-logging-level示例删除了指定事物组的日志级别。

```
aws iot delete-v2-logging-level \  
  --target-type THING_GROUP \  
  --target-name LightBulbs
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》LoggingLevel 中的 [deleteV2](#)。

## deprecate-thing-type

以下代码示例演示了如何使用 deprecate-thing-type。

### AWS CLI

#### 示例 1：弃用事物类型

以下 deprecate-thing-type 示例弃用了某一事物类型，因此用户无法将任何新事物与其相关联。

```
aws iot deprecate-thing-type \  
  --thing-type-name "obsoleteThingType"
```

此命令不生成任何输出。

#### 示例 2：撤消对某一事物的弃用

以下 deprecate-thing-type 示例撤消了对某一事物的弃用，这样用户就可以再次将新事物与其关联起来。

```
aws iot deprecate-thing-type \  
  --thing-type-name "obsoleteThingType" \  
  --undo-deprecate
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发人员指南》中的事物[类型](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeprecateThingType](#) 中的。

## describe-account-audit-configuration

以下代码示例演示了如何使用 describe-account-audit-configuration。



## AWS CLI

### 查看当前的审计配置设置

以下describe-account-audit-configuration示例列出了您的 AWS IoT Device Defender 审核配置的当前设置。

```
aws iot describe-account-audit-configuration
```

输出：

```
{
  "roleArn": "arn:aws:iam::123456789012:role/service-role/
AWSIoTDeviceDefenderAudit_1551201085996",
  "auditNotificationTargetConfigurations": {
    "SNS": {
      "targetArn": "arn:aws:sns:us-west-2:123456789012:ddaudits",
      "roleArn": "arn:aws:iam::123456789012:role/service-role/
AWSIoTDeviceDefenderAudit",
      "enabled": true
    }
  },
  "auditCheckConfigurations": {
    "AUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK": {
      "enabled": true
    },
    "CA_CERTIFICATE_EXPIRING_CHECK": {
      "enabled": true
    },
    "CONFLICTING_CLIENT_IDS_CHECK": {
      "enabled": true
    },
    "DEVICE_CERTIFICATE_EXPIRING_CHECK": {
      "enabled": true
    },
    "DEVICE_CERTIFICATE_SHARED_CHECK": {
      "enabled": true
    },
    "IOT_POLICY_OVERLY_PERMISSIVE_CHECK": {
      "enabled": true
    },
    "LOGGING_DISABLED_CHECK": {
      "enabled": true
    }
  }
}
```

```

    },
    "REVOKED_CA_CERTIFICATE_STILL_ACTIVE_CHECK": {
      "enabled": true
    },
    "REVOKED_DEVICE_CERTIFICATE_STILL_ACTIVE_CHECK": {
      "enabled": true
    },
    "UNAUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK": {
      "enabled": true
    }
  }
}

```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[审计命令](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeAccountAuditConfiguration](#)中的。

## describe-audit-finding

以下代码示例演示了如何使用 describe-audit-finding。

### AWS CLI

列出审计结果的详细信息

以下 describe-audit-finding 示例列出了指定 AWS IoT Device Defender 审计结果的详细信息。一次审计可以得出多个调查结果。使用 list-audit-findings 命令获取审计结果列表以获取 findingId。

```

aws iot describe-audit-finding \
  --finding-id "ef4826b8-e55a-44b9-b460-5c485355371b"

```

输出：

```

{
  "finding": {
    "findingId": "ef4826b8-e55a-44b9-b460-5c485355371b",
    "taskId": "873ed69c74a9ec8fa9b8e88e9abc4661",
    "checkName": "IOT_POLICY_OVERLY_PERMISSIVE_CHECK",
    "taskStartTime": 1576012045.745,
    "findingTime": 1576012046.168,
  }
}

```

```

    "severity": "CRITICAL",
    "nonCompliantResource": {
      "resourceType": "IOT_POLICY",
      "resourceIdentifier": {
        "policyVersionIdentifier": {
          "policyName": "smp-ggrass-group_Core-policy",
          "policyVersionId": "1"
        }
      }
    },
    "reasonForNonCompliance": "Policy allows broad access to IoT data plane
actions: [iot:Subscribe, iot:Connect, iot:GetThingShadow, iot>DeleteThingShadow,
iot:UpdateThingShadow, iot:Publish].",
    "reasonForNonComplianceCode":
"ALLOWS_BROAD_ACCESS_TO_IOT_DATA_PLANE_ACTIONS"
  }
}

```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检查审计结果（审计命令）](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeAuditFinding](#)中的。

## describe-audit-mitigation-actions-task

以下代码示例演示了如何使用 describe-audit-mitigation-actions-task。

### AWS CLI

显示审计缓解措施任务的详细信息

以下 describe-audit-mitigation-actions-task 示例显示了指定任务的详细信息，其中应用 ResetPolicyVersionAction 于查找结果。结果包括任务的开始和结束时间、针对的发现数量（以及结果），以及作为该任务一部分应用的操作的定义。

```

aws iot describe-audit-mitigation-actions-task \
  --task-id ResetPolicyTask01

```

输出：

```

{
  "taskStatus": "COMPLETED",
  "startTime": "2019-12-10T15:13:19.457000-08:00",

```

```

    "endTime": "2019-12-10T15:13:19.947000-08:00",
    "taskStatistics": {
      "IOT_POLICY_OVERLY_PERMISSIVE_CHECK": {
        "totalFindingsCount": 1,
        "failedFindingsCount": 0,
        "succeededFindingsCount": 1,
        "skippedFindingsCount": 0,
        "canceledFindingsCount": 0
      }
    },
    "target": {
      "findingIds": [
        "ef4826b8-e55a-44b9-b460-5c485355371b"
      ]
    },
    "auditCheckToActionsMapping": {
      "IOT_POLICY_OVERLY_PERMISSIVE_CHECK": [
        "ResetPolicyVersionAction"
      ]
    },
    "actionsDefinition": [
      {
        "name": "ResetPolicyVersionAction",
        "id": "1ea0b415-bef1-4a01-bd13-72fb63c59afb",
        "roleArn": "arn:aws:iam::123456789012:role/service-role/
ReplacePolicyVersionRole",
        "actionParams": {
          "replaceDefaultPolicyVersionParams": {
            "templateName": "BLANK_POLICY"
          }
        }
      }
    ]
  }
}

```

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [DescribeAuditMitigationActionsTask](#)（[缓解操作命令](#)）。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAuditMitigationActionsTask](#) 中的。

## describe-audit-suppression

以下代码示例演示了如何使用 describe-audit-suppression。

## AWS CLI

获取有关审计结果抑制的详细信息

以下describe-audit-suppression示例列出了有关禁止审计结果的详细信息。

```
aws iot describe-audit-task \  
  --task-id "787ed873b69cb4d6cdbae6ddd06996c5"
```

输出：

```
{  
  "taskStatus": "COMPLETED",  
  "taskType": "SCHEDULED_AUDIT_TASK",  
  "taskStartTime": 1596168096.157,  
  "taskStatistics": {  
    "totalChecks": 1,  
    "inProgressChecks": 0,  
    "waitingForDataCollectionChecks": 0,  
    "compliantChecks": 0,  
    "nonCompliantChecks": 1,  
    "failedChecks": 0,  
    "canceledChecks": 0  
  },  
  "scheduledAuditName": "AWSIoTDeviceDefenderDailyAudit",  
  "auditDetails": {  
    "DEVICE_CERTIFICATE_EXPIRING_CHECK": {  
      "checkRunStatus": "COMPLETED_NON_COMPLIANT",  
      "checkCompliant": false,  
      "totalResourcesCount": 195,  
      "nonCompliantResourcesCount": 2  
    }  
  }  
}
```

有关更多信息，请参阅《物AWS 联网开发人员指南》中的“[审计发现抑制](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAuditSuppression](#)中的。

### describe-audit-task

以下代码示例演示了如何使用 describe-audit-task。

## AWS CLI

### 获取有关审计实例的信息

以下describe-audit-task示例获取有关 AWS IoT Device Defender 审计实例的信息。如果审计完成，结果中将包括运行的汇总统计信息。

```
aws iot describe-audit-task \  
  --task-id a3aea009955e501a31b764abe1bebd3d
```

输出：

```
{  
  "taskStatus": "COMPLETED",  
  "taskType": "ON_DEMAND_AUDIT_TASK",  
  "taskStartTime": 1560356923.434,  
  "taskStatistics": {  
    "totalChecks": 3,  
    "InProgressChecks": 0,  
    "waitingForDataCollectionChecks": 0,  
    "compliantChecks": 3,  
    "nonCompliantChecks": 0,  
    "failedChecks": 0,  
    "canceledChecks": 0  
  },  
  "auditDetails": {  
    "CA_CERTIFICATE_EXPIRING_CHECK": {  
      "checkRunStatus": "COMPLETED_COMPLIANT",  
      "checkCompliant": true,  
      "totalResourcesCount": 0,  
      "nonCompliantResourcesCount": 0  
    },  
    "DEVICE_CERTIFICATE_EXPIRING_CHECK": {  
      "checkRunStatus": "COMPLETED_COMPLIANT",  
      "checkCompliant": true,  
      "totalResourcesCount": 6,  
      "nonCompliantResourcesCount": 0  
    },  
    "REVOKED_CA_CERTIFICATE_STILL_ACTIVE_CHECK": {  
      "checkRunStatus": "COMPLETED_COMPLIANT",  
      "checkCompliant": true,  
      "totalResourcesCount": 0,  
      "nonCompliantResourcesCount": 0  
    }  
  }  
}
```

```

    }
  }
}

```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[审计命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAuditTask](#)中的。

## describe-authorizer

以下代码示例演示了如何使用 describe-authorizer。

### AWS CLI

获取有关自定义授权方的信息

以下describe-authorizer示例显示了指定自定义授权方的详细信息。

```

aws iot describe-authorizer \
  --authorizer-name CustomAuthorizer

```

输出：

```

{
  "authorizerDescription": {
    "authorizerName": "CustomAuthorizer",
    "authorizerArn": "arn:aws:iot:us-west-2:123456789012:authorizer/
CustomAuthorizer",
    "authorizerFunctionArn": "arn:aws:lambda:us-
west-2:123456789012:function:CustomAuthorizerFunction",
    "tokenKeyName": "MyAuthToken",
    "tokenSigningPublicKeys": {
      "FIRST_KEY": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAA0CAQ8AMIIBCgKCAQEA1uJ0B4lQPgG/1M6ZfIwo
\nZ+7ENxAio9q6QD4FFqjGZsvjtYwjoe1RKK0U8Eq9xb503kRSmyIwTzwzm/f4Gf0Y
\nZUloJ+t3PUUwHrmbYTAGTrCUgRFyggfVwGCPs5ZAX4Eyqt5cr+AIHIiUDbxSa7p
\nzw0BKPeic0asNJpqT8PkBbRaKylEJh5oo81NDHHmVtbBm5A5YiJjqYXLaVAowKzZ\n
+GqsNvAQ9Jy1wI2VrEa10fL8f1DB/BJLm7zjpfP0HDJQgID0XnZwAlNnZc0hCwIx\n50g2LW20y9R/
dmqtDmJiVP97Z4GykxPvw1YHrUXY0iW1R3AR/Ac1NhCTGZMwVDB1\nlQIDAQAB\n-----END PUBLIC
KEY-----"
    },
    "status": "ACTIVE",
  }
}

```

```
    "creationDate": 1571245658.069,  
    "lastModifiedDate": 1571245658.069  
  }  
}
```

有关更多信息，请参阅 AWS IoT API 参考[DescribeAuthorizer](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeAuthorizer](#)中的。

## describe-billing-group

以下代码示例演示了如何使用 describe-billing-group。

### AWS CLI

获取账单组的相关信息

以下 describe-billing-group 示例获取指定账单组的信息。

```
aws iot describe-billing-group --billing-group-name GroupOne
```

输出：

```
{  
  "billingGroupName": "GroupOne",  
  "billingGroupId": "103de383-114b-4f51-8266-18f209ef5562",  
  "billingGroupArn": "arn:aws:iot:us-west-2:123456789012:billinggroup/GroupOne",  
  "version": 1,  
  "billingGroupProperties": {},  
  "billingGroupMetadata": {  
    "creationDate": 1560199355.378  
  }  
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[账单组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeBillingGroup](#)中的。

## describe-ca-certificate

以下代码示例演示了如何使用 describe-ca-certificate。



## AWS CLI

获取有关 CA 证书的详细信息

以下describe-ca-certificate示例显示了指定 CA 证书的详细信息。

```
aws iot describe-ca-certificate \
  --certificate-id
  f4efed62c0142f16af278166f61962501165c4f0536295207426460058cd1467
```

输出：

```
{
  "certificateDescription": {
    "certificateArn": "arn:aws:iot:us-west-2:123456789012:cacert/
f4efed62c0142f16af278166f61962501165c4f0536295207426460058cd1467",
    "certificateId":
    "f4efed62c0142f16af278166f61962501165c4f0536295207426460058cd1467",
    "status": "INACTIVE",
    "certificatePem": "-----BEGIN CERTIFICATE-----
\nMIICzzCCAbegEXAMPLEJANVEPWX18taPMA0GCSqSIB3DQEBBQUAMB4xCzAJBgNV
\nBAYTA1VMTMQ8wDQYDVQQKDAZBbWF6b24wHhcNMtkwOTI0MjEzMTU1WhcNMjkwOTIx
\nMjEzMTU1WjAeMQswCQYDVQQGEwJVUzEPMA0GA1UECgwGQW1hem9uMIIIBIjANBgkq
\nhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAZd3R3ioalCS0MhFWfBrVGR036EK07Uaf
\nVdz9EXAMPLE1VczICbADnATK522kEIB51/18Vz1FtAhQL5V5eybXKnB7QebNer5m
\n4Yibx7shR5oqNzFsrXWxuugN5+w5gEfqNMaw0jhF4Lscu1KG49yuqjcDU19/13ua
\n3B2gxs1Pe7TiWWvUskzxb01F2WCshbEjvqY8fIWtGYCjTeJAgQ9hvZx/69XhKen
\nwV9LJw0QxrsUS0Ty8IHwbB8fRy72VM3u7fJoaU+n04jD5cqaoEPtzoefUEXAMPLE
\nyVAJpqHwgbYbcUfn7V+AB6yh1+0Fa1rEQGuZDPGyJs1xwr5vh8nRewIDAQABoxAw
\nDjAMBgNVHRMEBTADAQH/MA0GCSqSIB3DQEBBQUAA4IBAQA+3a5CV3Ijg0nd0AgI
\nBgVMtmYzTvqAngx26aG9/spvCjXckh2SBF+EcB1CFwH1yakwjJL1dR4yarnrfxgI
\nEqP4A0YVimAVoQ5FBwnloHe16+3qtDib1U9DeXBUctS55EcfrEXAMPLEYtXdqU5C
\nU9ia4KAjV0dxW1+EFYmWx5eGeb0gDTNHBylV6B/f0SZiQAwDYp4x3B+gAP+a/bWB
\nu1um0qtBdWe6L6/83L+JhaTByqV25iVJ4c/UZUnG8926wU1DM9zQvEXuEVvzZ7+m\n4PSNqst/
nV0vnLpoG4e0WgcJgANuB33CSWtjWSuYsbhmQQRknGhREXAMPLEZT4fm\nfo0e\n-----END
CERTIFICATE-----\n",
    "ownedBy": "123456789012",
    "creationDate": 1569365372.053,
    "autoRegistrationStatus": "DISABLE",
    "lastModifiedDate": 1569365372.053,
    "customerVersion": 1,
    "generationId": "c5c2eb95-140b-4f49-9393-6aaac85b2a90",
    "validity": {
```

```

        "notBefore": 1569360675.0,
        "notAfter": 1884720675.0
    }
}
}

```

有关更多信息，请参阅《[物联网 API 参考](#)》中的“[描述证书](#)”。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeCaCertificate](#) 中的。

## describe-certificate

以下代码示例演示了如何使用 describe-certificate。

### AWS CLI

获取有关证书的信息

以下 describe-certificate 示例显示了指定证书的详细信息。

```

aws iot describe-certificate \
  --certificate-id
  "4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e"

```

输出：

```

{
  "certificateDescription": {
    "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e",
    "certificateId":
    "4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e",
    "status": "ACTIVE",
    "certificatePem": "-----BEGIN CERTIFICATE-----
MIICiTEXAMPLEQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBEXAMPLEMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC01BTSBDEXAMPLE1MRIwEAYDVQQDEwLUZXN0Q21sYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5EXAMPLEcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNEXAMPLEdBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC01BEXAMPLEz
b2xEXAMPLEYDVQQDEwLUZXN0Q21sYWMxHzAdBgkqhkiG9w0BCQEWEG5vb251QGft
YXpvbi5jb20wgZ8EXAMPLEZIHvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySwTC2XADZ4nB+BLYEXAMPLEpiwsZ3G93vUEI03IyNoH/f0wYK8m9T

```

```

rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7EXAMPLEGBzZswY6786m86gpE
Ibb30hjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFEXAMPLEAtCu4
nUhVVxYUnEXAMPLE8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GEXAMPLE10ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----",
    "ownedBy": "123456789012",
    "creationDate": 1541022751.983,
    "lastModifiedDate": 1541022751.983,
    "customerVersion": 1,
    "transferData": {},
    "generationId": "6974fbed-2e61-4114-bc5e-4204cc79b045",
    "validity": {
        "notBefore": 1541022631.0,
        "notAfter": 2524607999.0
    }
}
}
}

```

有关更多信息，请参阅 AWS IoT API 参考[DescribeCertificate](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeCertificate](#)中的。

## describe-custom-metric

以下代码示例演示了如何使用 describe-custom-metric。

### AWS CLI

获取有关 Device Defender 自定义指标的信息

以下 describe-custom-metric 示例获取有关名为 myCustomMetric 的自定义指标的信息。

```

aws iot describe-custom-metric \
  --metric-name myCustomMetric

```

输出：

```

{
  "metricName": "myCustomMetric",
  "metricArn": "arn:aws:iot:us-east-1:1234564789012:custommetric/myCustomMetric",
  "metricType": "number",
  "displayName": "My custom metric",
}

```

```
"creationDate": 2020-11-17T23:02:12.879000-09:00,  
"lastModifiedDate": 2020-11-17T23:02:12.879000-09:00  
}
```

有关更多信息，请参阅 AWS IoT Core 开发者指南中的[自定义指标](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeCustomMetric](#)中的。

## describe-default-authorizer

以下代码示例演示了如何使用 describe-default-authorizer。

### AWS CLI

获取有关默认自定义授权方的信息

以下 describe-default-authorizer 示例显示了默认自定义授权方的详细信息。

```
aws iot describe-default-authorizer
```

输出：

```
{  
  "authorizerName": "CustomAuthorizer",  
  "authorizerArn": "arn:aws:iot:us-west-2:123456789012:authorizer/  
CustomAuthorizer"  
}
```

有关更多信息，请参阅 AWS IoT API 参考[DescribeDefaultAuthorizer](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeDefaultAuthorizer](#)中的。

## describe-dimension

以下代码示例演示了如何使用 describe-dimension。

### AWS CLI

获取有关维度的信息

以下 describe-dimension 示例获取有关名为的维度的信息 TopicFilterForAuthMessages。

```
aws iot describe-dimension \  
  --name TopicFilterForAuthMessages
```

输出：

```
{  
  "name": "TopicFilterForAuthMessages",  
  "arn": "arn:aws:iot:eu-west-2:123456789012:dimension/  
TopicFilterForAuthMessages",  
  "type": "TOPIC_FILTER",  
  "stringValues": [  
    "device/+/auth"  
  ],  
  "creationDate": 1578620223.255,  
  "lastModifiedDate": 1578620223.255  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDimension](#)中的。

## describe-domain-configuration

以下代码示例演示了如何使用 describe-domain-configuration。

AWS CLI

描述域配置

以下describe-domain-configuration示例显示有关指定域配置的详细信息。

```
aws iot describe-domain-configuration \  
  --domain-configuration-name "additionalDataDomain"
```

输出：

```
{  
  "domainConfigurationName": "additionalDataDomain",  
  "domainConfigurationArn": "arn:aws:iot:us-  
east-1:758EXAMPLE143:domainconfiguration/additionalDataDomain/norpw",  
  "domainName": "d055exampleed74y71zfd-ats.beta.us-east-1.iot.amazonaws.com",
```

```
"serverCertificates": [],
"domainConfigurationStatus": "ENABLED",
"serviceType": "DATA",
"domainType": "AWS_MANAGED",
"lastStatusChangeDate": 1601923783.774
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[可配置终端节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDomainConfiguration](#) 中的。

## describe-endpoint

以下代码示例演示了如何使用 describe-endpoint。

### AWS CLI

示例 1：获取您当前的 AWS 终端节点

以下 describe-endpoint 示例检索应用所有命令的默认 AWS 端点。

```
aws iot describe-endpoint
```

输出：

```
{
  "endpointAddress": "abc123defghijk.iot.us-west-2.amazonaws.com"
}
```

有关更多信息，请参阅 [DescribeEndpoint](#) 《AWS 物联网开发人员指南》。

示例 2：获取您的 ATS 端点

以下 describe-endpoint 示例检索 Amazon Trust Services (ATS) 端点。

```
aws iot describe-endpoint \
  --endpoint-type iot:Data-ATS
```

输出：

```
{
  "endpointAddress": "abc123defghijk-ats.iot.us-west-2.amazonaws.com"
}
```

```
}
```

有关更多信息，请参阅《物联网开发者指南》中的 [X.509 证书和物 AWSAWS 联网](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEndpoint](#) 中的。

## describe-event-configurations

以下代码示例演示了如何使用 `describe-event-configurations`。

### AWS CLI

显示已发布的事件类型

以下 `describe-event-configurations` 示例列出了控制添加、更新或删除内容时生成哪些事件的配置。

```
aws iot describe-event-configurations
```

输出：

```
{
  "eventConfigurations": {
    "CA_CERTIFICATE": {
      "Enabled": false
    },
    "CERTIFICATE": {
      "Enabled": false
    },
    "JOB": {
      "Enabled": false
    },
    "JOB_EXECUTION": {
      "Enabled": false
    },
    "POLICY": {
      "Enabled": false
    },
    "THING": {
      "Enabled": false
    },
    "THING_GROUP": {
      "Enabled": false
    }
  }
}
```

```
    },
    "THING_GROUP_HIERARCHY": {
      "Enabled": false
    },
    "THING_GROUP_MEMBERSHIP": {
      "Enabled": false
    },
    "THING_TYPE": {
      "Enabled": false
    },
    "THING_TYPE_ASSOCIATION": {
      "Enabled": false
    }
  }
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[事件消息](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEventConfigurations](#) 中的。

## describe-index

以下代码示例演示了如何使用 describe-index。

### AWS CLI

检索事物索引的当前状态

以下 describe-index 示例检索事物索引的当前状态。

```
aws iot describe-index \
  --index-name "AWS_Things"
```

输出：

```
{
  "indexName": "AWS_Things",
  "indexStatus": "ACTIVE",
  "schema": "REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS"
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的管理事物[索引](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeIndex](#)中的。

## describe-job-execution

以下代码示例演示了如何使用 describe-job-execution。

### AWS CLI

在设备上获取任务的执行细节

以下describe-job-execution示例获取指定任务的执行详细信息。

```
aws iot describe-job-execution \  
  --job-id "example-job-01" \  
  --thing-name "MyRaspberryPi"
```

输出：

```
{  
  "execution": {  
    "jobId": "example-job-01",  
    "status": "QUEUED",  
    "statusDetails": {},  
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyRaspberryPi",  
    "queuedAt": 1560787023.636,  
    "lastUpdatedAt": 1560787023.636,  
    "executionNumber": 1,  
    "versionNumber": 1  
  }  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[创建和管理作业 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeJobExecution](#)中的。

## describe-job

以下代码示例演示了如何使用 describe-job。

### AWS CLI

获取任务的详细状态

以下describe-job示例获取 ID 为的任务的详细状态example-job-01。

```
aws iot describe-job \  
  --job-id "example-job-01"
```

输出：

```
{  
  "job": {  
    "jobArn": "arn:aws:iot:us-west-2:123456789012:job/example-job-01",  
    "jobId": "example-job-01",  
    "targetSelection": "SNAPSHOT",  
    "status": "IN_PROGRESS",  
    "targets": [  
      "arn:aws:iot:us-west-2:123456789012:thing/MyRaspberryPi"  
    ],  
    "description": "example job test",  
    "presignedUrlConfig": {},  
    "jobExecutionsRolloutConfig": {},  
    "createdAt": 1560787022.733,  
    "lastUpdatedAt": 1560787026.294,  
    "jobProcessDetails": {  
      "numberOfCanceledThings": 0,  
      "numberOfSucceededThings": 0,  
      "numberOfFailedThings": 0,  
      "numberOfRejectedThings": 0,  
      "numberOfQueuedThings": 1,  
      "numberOfInProgressThings": 0,  
      "numberOfRemovedThings": 0,  
      "numberOfTimedOutThings": 0  
    },  
    "timeoutConfig": {}  
  }  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[创建和管理作业 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeJob](#)中的。

## describe-mitigation-action

以下代码示例演示了如何使用 describe-mitigation-action。

## AWS CLI

查看已定义的缓解措施的详细信息

以下describe-mitigation-action示例显示了指定缓解措施的详细信息。

```
aws iot describe-mitigation-action \  
  --action-name AddThingsToQuarantineGroupAction
```

输出：

```
{  
  "actionName": "AddThingsToQuarantineGroupAction",  
  "actionType": "ADD_THINGS_TO_THING_GROUP",  
  "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/  
AddThingsToQuarantineGroupAction",  
  "actionId": "2fd2726d-98e1-4abf-b10f-09465ccd6bfa",  
  "roleArn": "arn:aws:iam::123456789012:role/service-role/  
MoveThingsToQuarantineGroupRole",  
  "actionParams": {  
    "addThingsToThingGroupParams": {  
      "thingGroupNames": [  
        "QuarantineGroup1"  
      ],  
      "overrideDynamicGroups": true  
    }  
  },  
  "creationDate": "2019-12-10T11:09:35.999000-08:00",  
  "lastModifiedDate": "2019-12-10T11:09:35.999000-08:00"  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [DescribeMitigationAction \(缓解操作命令\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeMitigationAction](#)中的。

## describe-provisioning-template-version

以下代码示例演示了如何使用 describe-provisioning-template-version。

## AWS CLI

描述配置模板版本

以下describe-provisioning-template-version示例描述了配置模板版本。

```
aws iot describe-provisioning-template-version \  
  --template-name MyTestProvisioningTemplate \  
  --version-id 1
```

输出：

```
{  
  "versionId": 1,  
  "creationDate": 1589308310.574,  
  "templateBody": "{  
    \"Parameters\":{  
      \"SerialNumber\":{  
        \"Type\": \"String\"  
      },  
      \"AWS::IoT::Certificate::Id\":{  
        \"Type\": \"String\"  
      }  
    },  
    \"Resources\":{  
      \"certificate\":{  
        \"Properties\":{  
          \"CertificateId\":{  
            \"Ref\": \"AWS::IoT::Certificate::Id\"  
          },  
          \"Status\": \"Active\"  
        },  
        \"Type\": \"AWS::IoT::Certificate\"  
      },  
      \"policy\":{  
        \"Properties\":{  
          \"PolicyName\": \"MyIotPolicy\"  
        },  
        \"Type\": \"AWS::IoT::Policy\"  
      },  
      \"thing\":{  
        \"OverrideSettings\":{  
          \"AttributePayload\": \"MERGE\",  
          \"ThingGroups\": \"DO_NOTHING\",  
          \"ThingTypeName\": \"REPLACE\"  
        },  
        \"Properties\":{
```

```

        \AttributePayload\:{},
        \ThingGroups\:[],
        \ThingName\:{
            \Fn::Join\:[
                \",",
                [
                    \DemoGroup_\",
                    {\Ref\:\SerialNumber\}
                ]
            ]
        },
        \ThingTypeName\:\VirtualThings\
    },
    \Type\:\AWS::IoT::Thing\
}
}
}],
"isDefaultVersion": true
}

```

有关更多信息，请参阅《AWS 物联网核心开发人员指南》中的[使用队列配置配置没有设备证书的设备](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeProvisioningTemplateVersion](#)中的。

## describe-provisioning-template

以下代码示例演示了如何使用 describe-provisioning-template。

### AWS CLI

#### 描述配置模板

以下 describe-provisioning-template 示例描述了配置模板。

```
aws iot describe-provisioning-template \
  --template-name MyTestProvisioningTemplate
```

输出：

```
{
  "templateArn": "arn:aws:iot:us-west-2:57EXAMPLE833:provisioningtemplate/
  MyTestProvisioningTemplate",

```

```

"templateName": "MyTestProvisioningTemplate",
"creationDate": 1589308310.574,
"lastModifiedDate": 1589308345.539,
"defaultVersionId": 1,
"templateBody": "{
  \"Parameters\":{
    \"SerialNumber\":{
      \"Type\": \"String\"
    },
    \"AWS::IoT::Certificate::Id\":{
      \"Type\": \"String\"
    }
  },
  \"Resources\":{
    \"certificate\":{
      \"Properties\":{
        \"CertificateId\":{
          \"Ref\": \"AWS::IoT::Certificate::Id\"
        },
        \"Status\": \"Active\"
      },
      \"Type\": \"AWS::IoT::Certificate\"
    },
    \"policy\":{
      \"Properties\":{
        \"PolicyName\": \"MyIotPolicy\"
      },
      \"Type\": \"AWS::IoT::Policy\"
    },
    \"thing\":{
      \"OverrideSettings\":{
        \"AttributePayload\": \"MERGE\",
        \"ThingGroups\": \"DO_NOTHING\",
        \"ThingTypeName\": \"REPLACE\"
      },
      \"Properties\":{
        \"AttributePayload\": {},
        \"ThingGroups\": [],
        \"ThingName\":{
          \"Fn::Join\": [
            \"\",
            [
              \"DemoGroup_\",
              {\"Ref\": \"SerialNumber\"}
            ]
          ]
        }
      }
    }
  }
}

```

```

        ]
      ],
    },
    \"ThingTypeName\": \"VirtualThings\"
  },
  \"Type\": \"AWS::IoT::Thing\"
}
}
}],
"enabled": true,
"provisioningRoleArn": "arn:aws:iam::571032923833:role/service-role/IoT_access"
}

```

有关更多信息，请参阅《AWS 物联网核心开发人员指南》中的[使用队列配置配置没有设备证书的设备](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeProvisioningTemplate](#)中的。

## describe-role-alias

以下代码示例演示了如何使用 `describe-role-alias`。

### AWS CLI

获取有关 AWS IoT 角色别名的信息

以下 `describe-role-alias` 示例显示了指定角色别名的详细信息。

```
aws iot describe-role-alias \
  --role-alias LightBulbRole
```

输出：

```
{
  "roleAliasDescription": {
    "roleAlias": "LightBulbRole",
    "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/
LightBulbRole",
    "roleArn": "arn:aws:iam::123456789012:role/light_bulb_role_001",
    "owner": "123456789012",
    "credentialDurationSeconds": 3600,
    "creationDate": 1570558643.221,
  }
}
```

```
    "lastModifiedDate": 1570558643.221
  }
}
```

有关更多信息，请参阅 AWS IoT API 参考[DescribeRoleAlias](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeRoleAlias](#)中的。

## describe-scheduled-audit

以下代码示例演示了如何使用 describe-scheduled-audit。

### AWS CLI

获取有关计划审计的信息

以下describe-scheduled-audit示例获取有关名为的 AWS IOT Device Defender 计划审核的详细信息AWSIoTDeviceDefenderDailyAudit。

```
aws iot describe-scheduled-audit \
  --scheduled-audit-name AWSIoTDeviceDefenderDailyAudit
```

输出：

```
{
  "frequency": "DAILY",
  "targetCheckNames": [
    "AUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK",
    "CONFLICTING_CLIENT_IDS_CHECK",
    "DEVICE_CERTIFICATE_SHARED_CHECK",
    "IOT_POLICY_OVERLY_PERMISSIVE_CHECK",
    "REVOKED_CA_CERTIFICATE_STILL_ACTIVE_CHECK",
    "UNAUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK"
  ],
  "scheduledAuditName": "AWSIoTDeviceDefenderDailyAudit",
  "scheduledAuditArn": "arn:aws:iot:us-west-2:123456789012:scheduledaudit/AWSIoTDeviceDefenderDailyAudit"
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[审计命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeScheduledAudit](#)中的。



## describe-security-profile

以下代码示例演示了如何使用 describe-security-profile。

### AWS CLI

获取有关安全配置文件的信息

以下describe-security-profile示例获取有关名为 AWS IoT Device Defender 安全配置文件的信息 PossibleIssue。

```
aws iot describe-security-profile \  
  --security-profile-name PossibleIssue
```

输出：

```
{  
  "securityProfileName": "PossibleIssue",  
  "securityProfileArn": "arn:aws:iot:us-west-2:123456789012:securityprofile/  
PossibleIssue",  
  "securityProfileDescription": "check to see if authorization fails 10 times in 5  
minutes or if cellular bandwidth exceeds 128",  
  "behaviors": [  
    {  
      "name": "CellularBandwidth",  
      "metric": "aws:message-byte-size",  
      "criteria": {  
        "comparisonOperator": "greater-than",  
        "value": {  
          "count": 128  
        },  
        "consecutiveDatapointsToAlarm": 1,  
        "consecutiveDatapointsToClear": 1  
      },  
    },  
    {  
      "name": "Authorization",  
      "metric": "aws:num-authorization-failures",  
      "criteria": {  
        "comparisonOperator": "greater-than",  
        "value": {  
          "count": 10  
        },  
      },  
    }  
  ]  
}
```

```
        "durationSeconds": 300,
        "consecutiveDatapointsToAlarm": 1,
        "consecutiveDatapointsToClear": 1
    }
}
],
"version": 1,
"creationDate": 1560278102.528,
"lastModifiedDate": 1560278102.528
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeSecurityProfile](#) 中的。

## describe-stream

以下代码示例演示了如何使用 describe-stream。

### AWS CLI

获取有关直播的信息

以下 describe-stream 示例显示有关指定直播的详细信息。

```
aws iot describe-stream \
  --stream-id stream12345
```

输出：

```
{
  "streamInfo": {
    "streamId": "stream12345",
    "streamArn": "arn:aws:iot:us-west-2:123456789012:stream/stream12345",
    "streamVersion": 1,
    "description": "This stream is used for Amazon FreeRTOS OTA Update 12345.",
    "files": [
      {
        "fileId": "123",
        "s3Location": {
          "bucket": "codesign-ota-bucket",
          "key": "48c67f3c-63bb-4f92-a98a-4ee0fbc2bef6"
        }
      }
    ]
  }
}
```

```

    }
  ],
  "createdAt": 1557863215.995,
  "lastUpdatedAt": 1557863215.995,
  "roleArn": "arn:aws:iam:123456789012:role/service-role/my_ota_stream_role"
}
}

```

有关更多信息，请参阅 AWS IoT API 参考[DescribeStream](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeStream](#)中的。

## describe-thing-group

以下代码示例演示了如何使用 describe-thing-group。

### AWS CLI

获取有关事物组的信息

以下 describe-thing-group 示例获取有关名为的事物组的信息 HalogenBulbs。

```
aws iot describe-thing-group \
  --thing-group-name HalogenBulbs
```

输出：

```

{
  "thingGroupName": "HalogenBulbs",
  "thingGroupId": "f4ec6b84-b42b-499d-9ce1-4dbd4d4f6f6e",
  "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/HalogenBulbs",
  "version": 1,
  "thingGroupProperties": {},
  "thingGroupMetadata": {
    "parentGroupName": "LightBulbs",
    "rootToParentThingGroups": [
      {
        "groupName": "LightBulbs",
        "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
LightBulbs"
      }
    ]
  },
  "creationDate": 1559927609.897
}

```

```
}  
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的事物[组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeThingGroup](#)中的。

## describe-thing-type

以下代码示例演示了如何使用 describe-thing-type。

### AWS CLI

获取有关事物类型的信息

以下describe-thing-type示例显示有关您的 AWS 账户中定义的指定事物类型的信息。

```
aws iot describe-thing-type \  
  --thing-type-name "LightBulb"
```

输出：

```
{  
  "thingTypeName": "LightBulb",  
  "thingTypeId": "ce3573b0-0a3c-45a7-ac93-4e0ce14cd190",  
  "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb",  
  "thingTypeProperties": {  
    "thingTypeDescription": "light bulb type",  
    "searchableAttributes": [  
      "model",  
      "wattage"  
    ]  
  },  
  "thingTypeMetadata": {  
    "deprecated": false,  
    "creationDate": 1559772562.498  
  }  
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的事物[类型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeThingType](#)中的。

## describe-thing

以下代码示例演示了如何使用 describe-thing。

### AWS CLI

显示有关事物的详细信息

以下describe-thing示例显示有关在您的 AWS 账户的 AWS IoT 注册表中定义的事物（设备）的信息。

```
aws iot 描述-thing-thing-name "" MyLightBulb
```

输出：

```
{
  "defaultClientId": "MyLightBulb",
  "thingName": "MyLightBulb",
  "thingId": "40da2e73-c6af-406e-b415-15acae538797",
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
  "thingTypeName": "LightBulb",
  "attributes": {
    "model": "123",
    "wattage": "75"
  },
  "version": 1
}
```

有关更多信息，请参阅《AWS IoT 开发人员指南》中的[如何使用注册表管理事物](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeThing](#)中的。

## detach-policy

以下代码示例演示了如何使用 detach-policy。

### AWS CLI

示例 1：将 AWS IoT 策略与事物组分离

以下detach-policy示例将指定的策略从事物组中分离出来，进而从该组中的所有事物以及该组的任何子组中分离出来。

```
aws iot detach-policy \  
  --target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs" \  
  --policy-name "MyFirstGroup_Core-policy"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[事物组](#)。

示例 2：将 AWS 物联网策略与设备证书分离

以下detach-policy示例将 TemperatureSensorPolicy 策略与 ARN 标识的设备证书分离。

```
aws iot detach-policy \  
  --policy-name TemperatureSensorPolicy \  
  --target arn:aws:iot:us-  
west-2:123456789012:cert/488b6a7f2acdeb00a77384e63c4e40b18b1b3caaae57b7272ba44c45e3448142
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachPolicy](#)中的。

## detach-security-profile

以下代码示例演示了如何使用 detach-security-profile。

### AWS CLI

取消安全配置文件与目标的关联

以下detach-security-profile示例删除了名为的 AWS IoT Device Defender 安全配置文件Testprofile与所有已注册事物目标之间的关联。

```
aws iot detach-security-profile \  
  --security-profile-name Testprofile \  
  --security-profile-target-arn "arn:aws:iot:us-west-2:123456789012:all/  
registered-things"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachSecurityProfile](#)中的。

## detach-thing-principal

以下代码示例演示了如何使用 detach-thing-principal。

### AWS CLI

将证书/委托人与事物分离

以下detach-thing-principal示例从指定事物中删除代表委托人的证书。

```
aws iot detach-thing-principal \  
  --thing-name "MyLightBulb" \  
  --principal "arn:aws:iot:us-  
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT 开发人员指南》中的[如何使用注册表管理事物](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachThingPrincipal](#)中的。

## disable-topic-rule

以下代码示例演示了如何使用 disable-topic-rule。

### AWS CLI

禁用主题规则

以下disable-topic-rule示例禁用了指定的主题规则。

```
aws iot disable-topic-rule \  
  --rule-name "MyPlantPiMoistureAlertRule"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[查看您的规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableTopicRule](#)中的。

## enable-topic-rule

以下代码示例演示了如何使用 `enable-topic-rule`。

### AWS CLI

启用主题规则

以下 `enable-topic-rule` 示例启用 ( 或重新启用 ) 指定的主题规则。

```
aws iot enable-topic-rule \  
  --rule-name "MyPlantPiMoistureAlertRule"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[查看您的规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [EnableTopicRule](#) 中的。

## get-behavior-model-training-summaries

以下代码示例演示了如何使用 `get-behavior-model-training-summaries`。

### AWS CLI

列出 Device Defender 的 ML Detect 安全配置文件训练模型的状态

以下 `get-behavior-model-training-summaries` 示例列出了所选安全配置文件中已配置行为的模型训练状态。对于每种行为，都会列出名称、模型状态和收集的数据点百分比。

```
aws iot get-behavior-model-training-summaries \  
  --security-profile-name MySecuirtyProfileName
```

输出：

```
{  
  "summaries": [  
    {  
      "securityProfileName": "MySecuirtyProfileName",  
      "behaviorName": "Messages_sent_ML_behavior",  
      "modelStatus": "PENDING_BUILD",
```



```

        "datapointsCollectionPercentage": 0.0
    },
    {
        "securityProfileName": "MySecuirtyProfileName",
        "behaviorName": "Messages_received_ML_behavior",
        "modelStatus": "PENDING_BUILD",
        "datapointsCollectionPercentage": 0.0
    },
    {
        "securityProfileName": "MySecuirtyProfileName",
        "behaviorName": "Authorization_failures_ML_behavior",
        "modelStatus": "PENDING_BUILD",
        "datapointsCollectionPercentage": 0.0
    },
    {
        "securityProfileName": "MySecuirtyProfileName",
        "behaviorName": "Message_size_ML_behavior",
        "modelStatus": "PENDING_BUILD",
        "datapointsCollectionPercentage": 0.0
    },
    {
        "securityProfileName": "MySecuirtyProfileName",
        "behaviorName": "Connection_attempts_ML_behavior",
        "modelStatus": "PENDING_BUILD",
        "datapointsCollectionPercentage": 0.0
    },
    {
        "securityProfileName": "MySPNoALerts",
        "behaviorName": "Disconnects_ML_behavior",
        "modelStatus": "PENDING_BUILD",
        "datapointsCollectionPercentage": 0.0
    }
]
}

```

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [GetBehaviorModelTrainingSummaries \(检测命令\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetBehaviorModelTrainingSummaries](#) 中的。

## get-cardinality

以下代码示例演示了如何使用 get-cardinality。

## AWS CLI

### 返回与查询匹配的唯一值的大致计数

您可以使用以下设置脚本创建 10 个代表 10 个温度传感器的东西。每个新事物都有 3 个属性。

```
# Bash script. If in other shells, type `bash` before running
Temperatures=(70 71 72 73 74 75 47 97 98 99)
Racks=(Rack1 Rack1 Rack2 Rack2 Rack3 Rack4 Rack5 Rack6 Rack6 Rack6)
IsNormal=(true true true true true true false false false false)
for ((i=0; i<10 ; i++))
do
  thing=$(aws iot create-thing --thing-name "TempSensor$i" --attribute-payload
  attributes="{temperature=${Temperatures[i]},rackId=${Racks[i]},stateNormal=
  ${IsNormal[i]}}")
  aws iot describe-thing --thing-name "TempSensor$i"
done
```

安装脚本的输出示例：

```
{
  "version": 1,
  "thingName": "TempSensor0",
  "defaultClientId": "TempSensor0",
  "attributes": {
    "rackId": "Rack1",
    "stateNormal": "true",
    "temperature": "70"
  },
  "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/TempSensor0",
  "thingId": "example1-90ab-cdef-fedc-ba987example"
}
```

以下 `get-cardinality` 示例查询设置脚本创建的 10 个传感器，并返回具有报告异常温度值的温度传感器的机架数量。如果温度值低于 60 或高于 80，则温度传感器处于异常状态。

```
aws iot get-cardinality \
  --aggregation-field "attributes.rackId" \
  --query-string "thingName:TempSensor* AND attributes.stateNormal:false"
```

输出：

```
{
  "cardinality": 2
}
```

有关更多信息，请参阅 < <https://docs.aws.amazon.com/iot/latest/developerguide/index-aggregate.html> > 《AWS 物联网开发者指南》中的查询聚合数据。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetCardinality](#) 中的。

## get-effective-policies

以下代码示例演示了如何使用 `get-effective-policies`。

### AWS CLI

列出影响某件事的策略

以下 `get-effective-policies` 示例列出了影响指定事物的策略，包括附加到其所属任何群组的策略。

```
aws iot get-effective-policies \
  --thing-name TemperatureSensor-001 \
  --principal arn:aws:iot:us-
west-2:123456789012:cert/488b6a7f2acdeb00a77384e63c4e40b18b1b3caaae57b7272ba44c45e3448142
```

输出：

```
{
  "effectivePolicies": [
    {
      "policyName": "TemperatureSensorPolicy",
      "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
TemperatureSensorPolicy",
      "policyDocument": "{
        \"Version\": \"2012-10-17\",
        \"Statement\": [
          {
            \"Effect\": \"Allow\",
            \"Action\": [
              \"iot:Publish\",
              \"iot:Receive\"
            ],
          }
        ]
      }
```

```

        \"Resource\": [
            \"arn:aws:iot:us-west-2:123456789012:topic/topic_1\",
            \"arn:aws:iot:us-west-2:123456789012:topic/topic_2\"
        ]
    },
    {
        \"Effect\": \"Allow\",
        \"Action\": [
            \"iot:Subscribe\"
        ],
        \"Resource\": [
            \"arn:aws:iot:us-west-2:123456789012:topicfilter/
topic_1\",
            \"arn:aws:iot:us-west-2:123456789012:topicfilter/
topic_2\"
        ]
    },
    {
        \"Effect\": \"Allow\",
        \"Action\": [
            \"iot:Connect\"
        ],
        \"Resource\": [
            \"arn:aws:iot:us-west-2:123456789012:client/basicPubSub
\"
        ]
    }
]
}
}
]
}

```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[获取事物的有效策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetEffectivePolicies](#)中的。

## get-indexing-configuration

以下代码示例演示了如何使用 get-indexing-configuration。

### AWS CLI

#### 获取事物索引配置

以下`get-indexing-configuration`示例获取 AWS 物联网队列索引的当前配置数据。

```
aws iot get-indexing-configuration
```

输出：

```
{
  "thingIndexingConfiguration": {
    "thingIndexingMode": "OFF",
    "thingConnectivityIndexingMode": "OFF"
  },
  "thingGroupIndexingConfiguration": {
    "thingGroupIndexingMode": "OFF"
  }
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的管理事物[索引](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetIndexingConfiguration](#)中的。

## get-job-document

以下代码示例演示了如何使用 `get-job-document`。

### AWS CLI

检索作业的文档

以下`get-job-document`示例显示有关 ID 为的任务的文档的详细信息`example-job-01`。

```
aws iot get-job-document \
  --job-id "example-job-01"
```

输出：

```
{
  "document": "\n{\n  \"operation\": \"customJob\", \n  \"otherInfo\":\n  \"someValue\"\n}\n"
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[创建和管理作业 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetJobDocument](#)中的。

## get-logging-options

以下代码示例演示了如何使用 `get-logging-options`。

### AWS CLI

获取日志选项

以下`get-logging-options`示例获取您 AWS 账户的当前日志选项。

```
aws iot get-logging-options
```

输出：

```
{
  "roleArn": "arn:aws:iam::123456789012:role/service-role/iotLoggingRole",
  "logLevel": "ERROR"
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的标题。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLoggingOptions](#)中的。

## get-ota-update

以下代码示例演示了如何使用 `get-ota-update`。

### AWS CLI

检索有关 OTA 更新的信息

以下`get-ota-update`示例显示有关指定 OTA 更新的详细信息。

```
aws iot get-ota-update \
  --ota-update-id ota12345
```

输出：

```
{
  "otaUpdateInfo": {
    "otaUpdateId": "ota12345",
    "otaUpdateArn": "arn:aws:iot:us-west-2:123456789012:otaupdate/itsaupdate",
    "creationDate": 1557863215.995,
  }
}
```

```
"lastModifiedDate": 1557863215.995,
"description": "A critical update needed right away.",
"targets": [
  "device1",
  "device2",
  "device3",
  "device4"
],
"targetSelection": "SNAPSHOT",
"protocols": ["HTTP"],
"awsJobExecutionsRolloutConfig": {
  "maximumPerMinute": 10
},
"otaUpdateFiles": [
  {
    "fileName": "firmware.bin",
    "fileLocation": {
      "stream": {
        "streamId": "004",
        "fileId": 123
      }
    },
    "codeSigning": {
      "awsSignerJobId": "48c67f3c-63bb-4f92-a98a-4ee0fbc2bef6"
    }
  }
],
"roleArn": "arn:aws:iam:123456789012:role/service-role/my_ota_role"
"otaUpdateStatus": "CREATE_COMPLETE",
"awsIotJobId": "job54321",
"awsIotJobArn": "arn:aws:iot:us-west-2:123456789012:job/job54321",
"errorInfo": {
}
}
}
```

有关更多信息，请参阅《物联网 API 参考》中的 [GetotaUpdate](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetOtaUpdate](#)中的。

## get-percentiles

以下代码示例演示了如何使用 `get-percentiles`。

## AWS CLI

### 将与查询匹配的聚合值分组为百分位分组

您可以使用以下设置脚本创建 10 个代表 10 个温度传感器的东西。每个新事物都有 1 个属性。

```
# Bash script. If in other shells, type `bash` before running
Temperatures=(70 71 72 73 74 75 47 97 98 99)
for ((i=0; i<10 ; i++))
do
    thing=$(aws iot create-thing --thing-name "TempSensor$i" --attribute-payload
attributes="{temperature=${Temperatures[i]}}")
    aws iot describe-thing --thing-name "TempSensor$i"
done
```

安装脚本的输出示例：

```
{
  "version": 1,
  "thingName": "TempSensor0",
  "defaultClientId": "TempSensor0",
  "attributes": {
    "temperature": "70"
  },
  "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/TempSensor0",
  "thingId": "example1-90ab-cdef-fedc-ba987example"
}
```

以下 `get-percentiles` 示例查询安装脚本创建的 10 个传感器，并为指定的每个百分位数组返回一个值。百分位数组“10”包含聚合的字段值，该值出现在与查询匹配的值的约 10% 中。在以下输出中，`{“百分比”：10.0，“值”：67.7}` 表示大约 10.0% 的温度值低于 67.7。

```
aws iot get-percentiles \
  --aggregation-field "attributes.temperature" \
  --query-string "thingName:TempSensor*" \
  --percents 10 25 50 75 90
```

输出：

```
{
  "percentiles": [
```



```
{
  "percent": 10.0,
  "value": 67.7
},
{
  "percent": 25.0,
  "value": 71.25
},
{
  "percent": 50.0,
  "value": 73.5
},
{
  "percent": 75.0,
  "value": 91.5
},
{
  "percent": 90.0,
  "value": 98.1
}
]
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[查询聚合数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPercentiles](#)中的。

## get-policy-version

以下代码示例演示了如何使用 get-policy-version。

### AWS CLI

获取有关策略特定版本的信息

以下get-policy-version示例获取有关指定策略的第一个版本的信息。

```
aws iot get-policy \
  --policy-name UpdateDeviceCertPolicy
  --policy-version-id "1"
```

输出：

```
{
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/UpdateDeviceCertPolicy",
  "policyName": "UpdateDeviceCertPolicy",
  "policyDocument": "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Effect\": \"Allow\", \"Action\": \"iot:UpdateCertificate\", \"Resource\": \"*\" } ] }",
  "policyVersionId": "1",
  "isDefaultVersion": false,
  "creationDate": 1559925941.924,
  "lastModifiedDate": 1559926175.458,
  "generationId":
  "5066f1b6712ce9d2a1e56399771649a272d6a921762fead080e24fe52f24e042"
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》中的AWS IoT 政策](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPolicyVersion](#)中的。

## get-policy

以下代码示例演示了如何使用 get-policy。

### AWS CLI

获取有关策略默认版本的信息

以下get-policy示例检索有关指定策略的默认版本的信息。

```
aws iot get-policy \
  --policy-name UpdateDeviceCertPolicy
```

输出：

```
{
  "policyName": "UpdateDeviceCertPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/UpdateDeviceCertPolicy",
  "policyDocument": "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Effect\": \"Allow\", \"Action\": \"iot:UpdateCertificate\", \"Resource\": \"*\" } ] }",
  "defaultVersionId": "2",
  "creationDate": 1559925941.924,
  "lastModifiedDate": 1559925941.924,
  "generationId":
  "5066f1b6712ce9d2a1e56399771649a272d6a921762fead080e24fe52f24e042"
```

```
}
```

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的[AWS IoT 政策](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPolicy](#)中的。

## get-registration-code

以下代码示例演示了如何使用 get-registration-code。

### AWS CLI

获取您的 AWS 账户专用注册码

以下get-registration-code示例检索您的 AWS 账户专用注册码。

```
aws iot get-registration-code
```

输出：

```
{
  "registrationCode":
  "15c51ae5e36ba59ba77042df1115862076bea4bd15841c838fcb68d5010a614c"
}
```

有关更多信息，请参阅《[AWS 物联网开发者指南](#)》中的[使用自己的证书](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRegistrationCode](#)中的。

## get-statistics

以下代码示例演示了如何使用 get-statistics。

### AWS CLI

在设备索引中搜索聚合数据

以下get-statistics示例返回设备影子中名为connectivity.connected为的属性设置为的事物的数量false（即未连接的设备数量）。

```
aws iot get-statistics \
```

```
--index-name AWS_Things \  
--query-string "connectivity.connected:false"
```

输出：

```
{  
  "statistics": {  
    "count": 6  
  }  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的“[获取有关您的设备队列的统计信息](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetStatistics](#)中的。

## get-topic-rule-destination

以下代码示例演示了如何使用 get-topic-rule-destination。

AWS CLI

获取主题规则目的地

以下get-topic-rule-destination示例获取有关主题规则目标的信息。

```
aws iot get-topic-rule-destination \  
  --arn "arn:aws:iot:us-west-2:123456789012:ruledestination/http/  
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
```

输出：

```
{  
  "topicRuleDestination": {  
    "arn": "arn:aws:iot:us-west-2:123456789012:ruledestination/http/  
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
    "status": "DISABLED",  
    "httpUrlProperties": {  
      "confirmationUrl": "https://example.com"  
    }  
  }  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[使用主题规则目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTopicRuleDestination](#)中的。

## get-topic-rule

以下代码示例演示了如何使用 get-topic-rule。

### AWS CLI

获取有关规则的信息

以下get-topic-rule示例获取有关指定规则的信息。

```
aws iot get-topic-rule \  
  --rule-name MyRPiLowMoistureAlertRule
```

输出：

```
{  
  "ruleArn": "arn:aws:iot:us-west-2:123456789012:rule/MyRPiLowMoistureAlertRule",  
  "rule": {  
    "ruleName": "MyRPiLowMoistureAlertRule",  
    "sql": "SELECT * FROM '$aws/things/MyRPi/shadow/update/accepted' WHERE  
state.reported.moisture = 'low'\n          ",  
    "description": "Sends an alert whenever soil moisture level readings are too  
low.",  
    "createdAt": 1558624363.0,  
    "actions": [  
      {  
        "sns": {  
          "targetArn": "arn:aws:sns:us-  
west-2:123456789012:MyRPiLowMoistureTopic",  
          "roleArn": "arn:aws:iam::123456789012:role/service-role/  
MyRPiLowMoistureTopicRole",  
          "messageFormat": "RAW"  
        }  
      }  
    ],  
    "ruleDisabled": false,  
    "awsIotSqlVersion": "2016-03-23"  
  }  
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[查看您的规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTopicRule](#)中的。

## get-v2-logging-options

以下代码示例演示了如何使用 get-v2-logging-options。

### AWS CLI

列出当前的日志记录选项

以下get-v2-logging-options示例列出了 I AWS oT 的当前日志选项。

```
aws iot get-v2-logging-options
```

输出：

```
{
  "roleArn": "arn:aws:iam::094249569039:role/service-role/iotLoggingRole",
  "defaultLogLevel": "WARN",
  "disableAllLogs": false
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的标题。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考LoggingOptions》中的 [GetV2](#)。

## list-active-violations

以下代码示例演示了如何使用 list-active-violations。

### AWS CLI

列出活跃的违规行为

以下list-active-violations示例列出了指定安全配置文件的所有违规行为。

```
aws iot list-active-violations \
  --security-profile-name Testprofile
```

输出：

```
{
  "activeViolations": [
    {
      "violationId": "174db59167fa474c80a652ad1583fd44",
      "thingName": "iotconsole-1560269126751-1",
      "securityProfileName": "Testprofile",
      "behavior": {
        "name": "Authorization",
        "metric": "aws:num-authorization-failures",
        "criteria": {
          "comparisonOperator": "greater-than",
          "value": {
            "count": 10
          },
          "durationSeconds": 300,
          "consecutiveDatapointsToAlarm": 1,
          "consecutiveDatapointsToClear": 1
        }
      },
      "lastViolationValue": {
        "count": 0
      },
      "lastViolationTime": 1560293700.0,
      "violationStartTime": 1560279000.0
    },
    {
      "violationId": "c8a9466a093d3b7b35cd44ca58bdbeab",
      "thingName": "TvnQoEoU",
      "securityProfileName": "Testprofile",
      "behavior": {
        "name": "CellularBandwidth",
        "metric": "aws:message-byte-size",
        "criteria": {
          "comparisonOperator": "greater-than",
          "value": {
            "count": 128
          },
          "consecutiveDatapointsToAlarm": 1,
          "consecutiveDatapointsToClear": 1
        }
      },
      "lastViolationValue": {
```

```
        "count": 110
      },
      "lastViolationTime": 1560369000.0,
      "violationStartTime": 1560276600.0
    },
    {
      "violationId": "74aa393adea02e6648f3ac362beed55e",
      "thingName": "iotconsole-1560269232412-2",
      "securityProfileName": "Testprofile",
      "behavior": {
        "name": "Authorization",
        "metric": "aws:num-authorization-failures",
        "criteria": {
          "comparisonOperator": "greater-than",
          "value": {
            "count": 10
          },
          "durationSeconds": 300,
          "consecutiveDatapointsToAlarm": 1,
          "consecutiveDatapointsToClear": 1
        }
      },
      "lastViolationValue": {
        "count": 0
      },
      "lastViolationTime": 1560276600.0,
      "violationStartTime": 1560276600.0
    },
    {
      "violationId": "1e6ab5f7cf39a1466fcd154e1377e406",
      "thingName": "TvnQoEoU",
      "securityProfileName": "Testprofile",
      "behavior": {
        "name": "Authorization",
        "metric": "aws:num-authorization-failures",
        "criteria": {
          "comparisonOperator": "greater-than",
          "value": {
            "count": 10
          },
          "durationSeconds": 300,
          "consecutiveDatapointsToAlarm": 1,
          "consecutiveDatapointsToClear": 1
        }
      }
    }
  ]
}
```



```
    },
    "lastViolationValue": {
      "count": 0
    },
    "lastViolationTime": 1560369000.0,
    "violationStartTime": 1560276600.0
  }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListActiveViolations](#)中的。

## list-attached-policies

以下代码示例演示了如何使用 list-attached-policies。

### AWS CLI

示例 1：列出附加到群组的策略

以下list-attached-policies示例列出了附加到指定组的策略。

```
aws iot list-attached-policies \
  --target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
```

输出：

```
{
  "policies": [
    {
      "policyName": "UpdateDeviceCertPolicy",
      "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
UpdateDeviceCertPolicy"
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的事物[组](#)。

示例 2：列出附加到设备证书的策略

以下 `list-attached-policies` 示例列出了附加到设备证书的 AWS IoT 策略。证书由其 ARN 标识。

```
aws iot list-attached-policies \  
  --target arn:aws:iot:us-  
west-2:123456789012:cert/488b6a7f2acdeb00a77384e63c4e40b18b1b3caaae57b7272ba44c45e3448142
```

输出：

```
{  
  "policies": [  
    {  
      "policyName": "TemperatureSensorPolicy",  
      "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/  
TemperatureSensorPolicy"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的事物组。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAttachedPolicies](#) 中的。

## list-audit-findings

以下代码示例演示了如何使用 `list-audit-findings`。

### AWS CLI

示例 1：列出审计的所有结果

以下 `list-audit-findings` 示例列出了具有指定任务 ID 的 AWS IoT Device Defender 审计的所有结果。

```
aws iot list-audit-findings \  
  --task-id a3aea009955e501a31b764abe1bebd3d
```

输出：

```
{  
  "findings": []
```

```
}
```

## 示例 2：列出审计检查类型的结果

以下list-audit-findings示例显示了 2019 年 6 月 5 日至 2019 年 6 月 19 日期间进行的 AWS IoT Device Defender 审计的结果，其中设备共享设备证书。指定支票名称时，必须提供开始和结束时间。

```
aws iot list-audit-findings \  
  --check-name DEVICE_CERTIFICATE_SHARED_CHECK \  
  --start-time 1559747125 \  
  --end-time 1560962028
```

输出：

```
{  
  "findings": [  
    {  
      "taskId": "eef61068b0eb03c456d746c5a26ee04",  
      "checkName": "DEVICE_CERTIFICATE_SHARED_CHECK",  
      "taskStartTime": 1560161017.172,  
      "findingTime": 1560161017.592,  
      "severity": "CRITICAL",  
      "nonCompliantResource": {  
        "resourceType": "DEVICE_CERTIFICATE",  
        "resourceIdentifier": {  
          "deviceCertificateId":  
"b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b"  
        }  
      },  
      "relatedResources": [  
        {  
          "resourceType": "CLIENT_ID",  
          "resourceIdentifier": {  
            "clientId": "ZipxgAI1"  
          },  
          "additionalInfo": {  
            "CONNECTION_TIME": "1560086374068"  
          }  
        },  
        {  
          "resourceType": "CLIENT_ID",  
          "resourceIdentifier": {
```

```
        "clientId": "ZipxgAIl"
      },
      "additionalInfo": {
        "CONNECTION_TIME": "1560081552187",
        "DISCONNECTION_TIME": "1560086371552"
      }
    },
    {
      "resourceType": "CLIENT_ID",
      "resourceIdentifier": {
        "clientId": "ZipxgAIl"
      },
      "additionalInfo": {
        "CONNECTION_TIME": "1559289863631",
        "DISCONNECTION_TIME": "1560081532716"
      }
    }
  ],
  "reasonForNonCompliance": "Certificate shared by one or more devices.",
  "reasonForNonComplianceCode": "CERTIFICATE_SHARED_BY_MULTIPLE_DEVICES"
},
{
  "taskId": "bade6b5efd2e1b1569822f6021b39cf5",
  "checkName": "DEVICE_CERTIFICATE_SHARED_CHECK",
  "taskStartTime": 1559988217.27,
  "findingTime": 1559988217.655,
  "severity": "CRITICAL",
  "nonCompliantResource": {
    "resourceType": "DEVICE_CERTIFICATE",
    "resourceIdentifier": {
      "deviceCertificateId":
"b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b"
    }
  },
  "relatedResources": [
    {
      "resourceType": "CLIENT_ID",
      "resourceIdentifier": {
        "clientId": "xShGENLW"
      },
      "additionalInfo": {
        "CONNECTION_TIME": "1559972350825"
      }
    }
  ],
}
```

```

        {
            "resourceType": "CLIENT_ID",
            "resourceIdentifier": {
                "clientId": "xShGENLW"
            },
            "additionalInfo": {
                "CONNECTION_TIME": "1559255062002",
                "DISCONNECTION_TIME": "1559972350616"
            }
        }
    ],
    "reasonForNonCompliance": "Certificate shared by one or more devices.",
    "reasonForNonComplianceCode": "CERTIFICATE_SHARED_BY_MULTIPLE_DEVICES"
},
{
    "taskId": "c23f6233ba2d35879c4bb2810fb5ffd6",
    "checkName": "DEVICE_CERTIFICATE_SHARED_CHECK",
    "taskStartTime": 1559901817.31,
    "findingTime": 1559901817.767,
    "severity": "CRITICAL",
    "nonCompliantResource": {
        "resourceType": "DEVICE_CERTIFICATE",
        "resourceIdentifier": {
            "deviceCertificateId":
"b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b"
        }
    },
    "relatedResources": [
        {
            "resourceType": "CLIENT_ID",
            "resourceIdentifier": {
                "clientId": "TvnQoEoU"
            },
            "additionalInfo": {
                "CONNECTION_TIME": "1559826729768"
            }
        },
        {
            "resourceType": "CLIENT_ID",
            "resourceIdentifier": {
                "clientId": "TvnQoEoU"
            },
            "additionalInfo": {
                "CONNECTION_TIME": "1559345920964",

```

```

        "DISCONNECTION_TIME": "1559826728402"
      }
    }
  ],
  "reasonForNonCompliance": "Certificate shared by one or more devices.",
  "reasonForNonComplianceCode": "CERTIFICATE_SHARED_BY_MULTIPLE_DEVICES"
}
]
}

```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[审计命令](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAuditFindings](#) 中的。

## list-audit-mitigation-actions-executions

以下代码示例演示了如何使用 `list-audit-mitigation-actions-executions`。

### AWS CLI

列出审计缓解措施执行的详细信息

审计缓解措施任务对 AWS IoT Device Defender 审计中的一个或多个发现应用缓解措施。以下 `list-audit-mitigation-actions-executions` 示例列出了具有指定结果的缓解操作任务 `taskId` 和指定结果的详细信息。

```

aws iot list-audit-mitigation-actions-executions \
  --task-id myActionsTaskId \
  --finding-id 0edbaaec-2fe1-4cf5-abc9-d4c3e51f7464

```

输出：

```

{
  "actionsExecutions": [
    {
      "taskId": "myActionsTaskId",
      "findingId": "0edbaaec-2fe1-4cf5-abc9-d4c3e51f7464",
      "actionName": "ResetPolicyVersionAction",
      "actionId": "1ea0b415-bef1-4a01-bd13-72fb63c59afb",
      "status": "COMPLETED",
      "startTime": "2019-12-10T15:19:13.279000-08:00",
    }
  ]
}

```

```
        "endTime": "2019-12-10T15:19:13.337000-08:00"
      }
    ]
  }
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [ListAuditMitigationActionsExecutions](#)（[缓解操作命令](#)）。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAuditMitigationActionsExecutions](#)中的。

## list-audit-mitigation-actions-tasks

以下代码示例演示了如何使用 `list-audit-mitigation-actions-tasks`。

### AWS CLI

#### 列出审计缓解措施任务

以下`list-audit-mitigation-actions-tasks`示例列出了在指定时间段内应用于发现的缓解措施。

```
aws iot list-audit-mitigation-actions-tasks \
  --start-time 1594157400 \
  --end-time 1594157430
```

输出：

```
{
  "tasks": [
    {
      "taskId": "0062f2d6-3999-488f-88c7-bef005414103",
      "startTime": "2020-07-07T14:30:15.172000-07:00",
      "taskStatus": "COMPLETED"
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [ListAuditMitigationActionsTasks](#)（[缓解操作命令](#)）。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAuditMitigationActionsTasks](#)中的。

## list-audit-suppressions

以下代码示例演示了如何使用 `list-audit-suppressions`。

### AWS CLI

列出所有审计结果抑制项

以下 `list-audit-suppressions` 示例列出了所有有效的审计结果抑制。

```
aws iot list-audit-suppressions
```

输出：

```
{
  "suppressions": [
    {
      "checkName": "DEVICE_CERTIFICATE_EXPIRING_CHECK",
      "resourceIdentifier": {
        "deviceCertificateId": "c7691e<shortened>"
      },
      "expirationDate": 1597881600.0,
      "suppressIndefinitely": false
    }
  ]
}
```

有关更多信息，请参阅《物AWS 联网开发人员指南》中的 [“审计发现抑制”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAuditSuppressions](#)中的。

## list-audit-tasks

以下代码示例演示了如何使用 `list-audit-tasks`。

### AWS CLI

列出审计的所有结果

以下 `list-audit-tasks` 示例列出了在 2019 年 6 月 5 日至 2019 年 6 月 12 日之间运行的审计任务。



```
aws iot list-audit-tasks \  
  --start-time 1559747125 \  
  --end-time 1560357228
```

输出：

```
{  
  "tasks": [  
    {  
      "taskId": "a3aea009955e501a31b764abe1bebd3d",  
      "taskStatus": "COMPLETED",  
      "taskType": "ON_DEMAND_AUDIT_TASK"  
    },  
    {  
      "taskId": "f76b4b5102b632cd9ae38a279c266da1",  
      "taskStatus": "COMPLETED",  
      "taskType": "SCHEDULED_AUDIT_TASK"  
    },  
    {  
      "taskId": "51d9967d9f9ff4d26529505f6d2c444a",  
      "taskStatus": "COMPLETED",  
      "taskType": "SCHEDULED_AUDIT_TASK"  
    },  
    {  
      "taskId": "eef61068b0eb03c456d746c5a26ee04",  
      "taskStatus": "COMPLETED",  
      "taskType": "SCHEDULED_AUDIT_TASK"  
    },  
    {  
      "taskId": "041c49557b7c7b04c079a49514b55589",  
      "taskStatus": "COMPLETED",  
      "taskType": "SCHEDULED_AUDIT_TASK"  
    },  
    {  
      "taskId": "82c7f2afac1562d18a4560be73998acc",  
      "taskStatus": "COMPLETED",  
      "taskType": "SCHEDULED_AUDIT_TASK"  
    },  
    {  
      "taskId": "bade6b5efd2e1b1569822f6021b39cf5",  
      "taskStatus": "COMPLETED",  
      "taskType": "SCHEDULED_AUDIT_TASK"  
    }  
  ],  
}
```

```
{
  "taskId": "c23f6233ba2d35879c4bb2810fb5ffd6",
  "taskStatus": "COMPLETED",
  "taskType": "SCHEDULED_AUDIT_TASK"
},
{
  "taskId": "ac9086b7222a2f5e2e17bb6fd30b3aeb",
  "taskStatus": "COMPLETED",
  "taskType": "SCHEDULED_AUDIT_TASK"
}
]
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[审计命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAuditTasks](#)中的。

## list-authorizers

以下代码示例演示了如何使用 list-authorizers。

### AWS CLI

列出您的自定义授权方

以下list-authorizers示例列出了您 AWS 账户中的自定义授权方。

```
aws iot list-authorizers
```

输出：

```
{
  "authorizers": [
    {
      "authorizerName": "CustomAuthorizer",
      "authorizerArn": "arn:aws:iot:us-west-2:123456789012:authorizer/CustomAuthorizer"
    },
    {
      "authorizerName": "CustomAuthorizer2",
      "authorizerArn": "arn:aws:iot:us-west-2:123456789012:authorizer/CustomAuthorizer2"
    }
  ]
}
```

```
{
  "authorizerName": "CustomAuthorizer3",
  "authorizerArn": "arn:aws:iot:us-west-2:123456789012:authorizer/
CustomAuthorizer3"
}
]
```

有关更多信息，请参阅 AWS IoT API 参考[ListAuthorizers](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAuthorizers](#)中的。

## list-billing-groups

以下代码示例演示了如何使用 list-billing-groups。

### AWS CLI

列出您的 AWS 账户和地区的账单组

以下list-billing-groups示例列出了为您的 AWS 账户和 AWS 地区定义的所有账单组。

```
aws iot list-billing-groups
```

输出：

```
{
  "billingGroups": [
    {
      "groupName": "GroupOne",
      "groupArn": "arn:aws:iot:us-west-2:123456789012:billinggroup/GroupOne"
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[账单组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBillingGroups](#)中的。

## list-ca-certificates

以下代码示例演示了如何使用 list-ca-certificates。

## AWS CLI

列出在您的 AWS 账户中注册的 CA 证书

以下`list-ca-certificates`示例列出了在您的 AWS 账户中注册的 CA 证书。

```
aws iot list-ca-certificates
```

输出：

```
{
  "certificates": [
    {
      "certificateArn": "arn:aws:iot:us-west-2:123456789012:cacert/
f4efed62c0142f16af278166f61962501165c4f0536295207426460058cd1467",
      "certificateId":
"f4efed62c0142f16af278166f61962501165c4f0536295207426460058cd1467",
      "status": "INACTIVE",
      "creationDate": 1569365372.053
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[使用自己的证书](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCaCertificates](#)中的。

## **list-certificates-by-ca**

以下代码示例演示了如何使用 `list-certificates-by-ca`。

## AWS CLI

列出所有使用 CA 证书签名的设备证书

以下`list-certificates-by-ca`示例列出了您 AWS 账户中使用指定 CA 证书签名的所有设备证书。

```
aws iot list-certificates-by-ca \
  --ca-certificate-id
f4efed62c0142f16af278166f61962501165c4f0536295207426460058cd1467
```

输出：

```
{
  "certificates": [
    {
      "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/488b6a7f2acdeb00a77384e63c4e40b18b1b3caaae57b7272ba44c45e3448142",
      "certificateId":
"488b6a7f2acdeb00a77384e63c4e40b18b1b3caaae57b7272ba44c45e3448142",
      "status": "ACTIVE",
      "creationDate": 1569363250.557
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网 API 参考》中的 [ListCertificatesByCA](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCertificatesByCa](#)中的。

## list-certificates

以下代码示例演示了如何使用 list-certificates。

### AWS CLI

示例 1：列出在您的 AWS 账户中注册的证书

以下list-certificates示例列出了在您的账户中注册的所有证书。如果您的分页限制超过默认的 25，则可以使用此命令中的nextMarker响应值并将其提供给下一个命令以获取下一批结果。重复此操作，直到nextMarker返回时没有值。

```
aws iot list-certificates
```

输出：

```
{
  "certificates": [
    {
      "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36",
      "certificateId":
"604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36",
      "status": "ACTIVE",
      "creationDate": 1556810537.617
    }
  ]
}
```

```
    },
    {
      "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e",
      "certificateId":
"262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e",
      "status": "ACTIVE",
      "creationDate": 1546447050.885
    },
    {
      "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
      "certificateId":
"b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
      "status": "ACTIVE",
      "creationDate": 1546292258.322
    },
    {
      "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42",
      "certificateId":
"7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42",
      "status": "ACTIVE",
      "creationDate": 1541457693.453
    },
    {
      "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3",
      "certificateId":
"54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3",
      "status": "ACTIVE",
      "creationDate": 1541113568.611
    },
    {
      "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e",
      "certificateId":
"4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e",
      "status": "ACTIVE",
      "creationDate": 1541022751.983
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCertificates](#)中的。

## list-custom-metrics

以下代码示例演示了如何使用 `list-custom-metrics`。

### AWS CLI

列出您的自定义指标

以下`list-custom-metrics`示例列出了您的所有自定义指标。

```
aws iot list-custom-metrics \  
  --region us-east-1
```

输出：

```
{  
  "metricNames": [  
    "batteryPercentage"  
  ]  
}
```

有关更多信息，请参阅 AWS IoT Core 开发者指南中的[自定义指标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCustomMetrics](#)中的。

## list-dimensions

以下代码示例演示了如何使用 `list-dimensions`。

### AWS CLI

列出您 AWS 账户的维度

以下`list-dimensions`示例列出了在您的 AWS 账户中定义的所有 AWS IoT Device Defender 维度。

```
aws iot list-dimensions
```

输出：

```
{
  "dimensionNames": [
    "TopicFilterForAuthMessages",
    "TopicFilterForActivityMessages"
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDimensions](#)中的。

## list-domain-configurations

以下代码示例演示了如何使用 list-domain-configurations。

### AWS CLI

#### 列出域配置

以下list-domain-configurations示例列出了您 AWS 账户中具有指定服务类型的域配置。

```
aws iot list-domain-configurations \
  --service-type "DATA"
```

#### 输出：

```
{
  "domainConfigurations":
  [
    {
      "domainConfigurationName": "additionalDataDomain",
      "domainConfigurationArn": "arn:aws:iot:us-
west-2:123456789012:domainconfiguration/additionalDataDomain/dikMh",
      "serviceType": "DATA"
    },
    {
      "domainConfigurationName": "iot:Jobs",
      "domainConfigurationArn": "arn:aws:iot:us-
west-2:123456789012:domainconfiguration/iot:Jobs",
      "serviceType": "JOBS"
    },
  ],
}
```



```
{
  "domainConfigurationName": "iot:Data-ATS",
  "domainConfigurationArn": "arn:aws:iot:us-
west-2:123456789012:domainconfiguration/iot:Data-ATS",
  "serviceType": "DATA"
},
{
  "domainConfigurationName": "iot:CredentialProvider",
  "domainConfigurationArn": "arn:aws:iot:us-
west-2:123456789012:domainconfiguration/iot:CredentialProvider",
  "serviceType": "CREDENTIAL_PROVIDER"
}
]
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[可配置终端节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListDomainConfigurations](#)中的。

## list-indices

以下代码示例演示了如何使用 list-indices。

### AWS CLI

列出已配置的搜索索引

以下 list-indices 示例列出了您 AWS 账户中所有已配置的搜索索引。如果您尚未启用事物索引，则可能没有任何索引。

```
aws iot list-indices
```

输出：

```
{
  "indexNames": [
    "AWS_Things"
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[管理事物索引](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListIndices](#)中的。

## list-job-executions-for-job

以下代码示例演示了如何使用 `list-job-executions-for-job`。

### AWS CLI

列出您 AWS 账户中的职位

以下 `list-job-executions-for-job` 示例列出了由 `jobid` 指定的您 AWS 账户中某项任务的所有任务执行情况。

```
aws iot list-job-executions-for-job \  
  --job-id my-ota-job
```

输出：

```
{  
  "executionSummaries": [  
    {  
      "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/my_thing",  
      "jobExecutionSummary": {  
        "status": "QUEUED",  
        "queuedAt": "2022-03-07T15:58:42.195000-08:00",  
        "lastUpdatedAt": "2022-03-07T15:58:42.195000-08:00",  
        "executionNumber": 1,  
        "retryAttempt": 0  
      }  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[创建和管理作业 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListJobExecutionsForJob](#) 中的。

## list-job-executions-for-thing

以下代码示例演示了如何使用 `list-job-executions-for-thing`。

### AWS CLI

列出为某件事执行的作业

以下`list-job-executions-for-thing`示例列出了为名为的事物执行的所有作业`MyRaspberryPi`。

```
aws iot list-job-executions-for-thing \  
  --thing-name "MyRaspberryPi"
```

输出：

```
{  
  "executionSummaries": [  
    {  
      "jobId": "example-job-01",  
      "jobExecutionSummary": {  
        "status": "QUEUED",  
        "queuedAt": 1560787023.636,  
        "lastUpdatedAt": 1560787023.636,  
        "executionNumber": 1  
      }  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[创建和管理作业 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListJobExecutionsForThing](#)中的。

## list-jobs

以下代码示例演示了如何使用 `list-jobs`。

### AWS CLI

列出您 AWS 账户中的职位

以下`list-jobs`示例列出了您 AWS 账户中的所有任务，并按任务状态排序。

```
aws iot list-jobs
```

输出：

```
{  
  "jobs": [  

```

```
{
  "jobArn": "arn:aws:iot:us-west-2:123456789012:job/example-job-01",
  "jobId": "example-job-01",
  "targetSelection": "SNAPSHOT",
  "status": "IN_PROGRESS",
  "createdAt": 1560787022.733,
  "lastUpdatedAt": 1560787026.294
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[创建和管理作业 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListJobs](#)中的。

## list-mitigation-actions

以下代码示例演示了如何使用 list-mitigation-actions。

### AWS CLI

列出所有已定义的缓解措施

以下list-mitigation-actions示例列出了针对您的 AWS 账户和地区定义的所有缓解措施。列出了每个操作的名称、ARN 和创建日期。

```
aws iot list-mitigation-actions
```

输出：

```
{
  "actionIdentifiers": [
    {
      "actionName": "DeactivateCACertAction",
      "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/DeactivateCACertAction",
      "creationDate": "2019-12-10T11:12:47.574000-08:00"
    },
    {
      "actionName": "ResetPolicyVersionAction",
      "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/ResetPolicyVersionAction",

```

```

        "creationDate": "2019-12-10T11:11:48.920000-08:00"
    },
    {
        "actionName": "PublishFindingToSNSAction",
        "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/
PublishFindingToSNSAction",
        "creationDate": "2019-12-10T11:10:49.546000-08:00"
    },
    {
        "actionName": "AddThingsToQuarantineGroupAction",
        "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/
AddThingsToQuarantineGroupAction",
        "creationDate": "2019-12-10T11:09:35.999000-08:00"
    },
    {
        "actionName": "UpdateDeviceCertAction",
        "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/
UpdateDeviceCertAction",
        "creationDate": "2019-12-10T11:08:44.263000-08:00"
    },
    {
        "actionName": "SampleMitigationAction",
        "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/
SampleMitigationAction",
        "creationDate": "2019-12-10T11:03:41.840000-08:00"
    }
]
}

```

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [ListMitigationActions \( 缓解操作命令 \)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListMitigationActions](#) 中的。

## list-mitigations-actions

以下代码示例演示了如何使用 list-mitigations-actions。

### AWS CLI

列出所有已定义的缓解措施

以下 list-mitigations-actions 示例列出了针对您的 AWS 账户和地区定义的所有缓解措施。列出了每个操作的名称、ARN 和创建日期。

```
aws iot list-mitigation-actions
```

输出：

```
{
  "actionIdentifiers": [
    {
      "actionName": "DeactivateCACertAction",
      "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/DeactivateCACertAction",
      "creationDate": "2019-12-10T11:12:47.574000-08:00"
    },
    {
      "actionName": "ResetPolicyVersionAction",
      "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/ResetPolicyVersionAction",
      "creationDate": "2019-12-10T11:11:48.920000-08:00"
    },
    {
      "actionName": "PublishFindingToSNSAction",
      "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/PublishFindingToSNSAction",
      "creationDate": "2019-12-10T11:10:49.546000-08:00"
    },
    {
      "actionName": "AddThingsToQuarantineGroupAction",
      "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/AddThingsToQuarantineGroupAction",
      "creationDate": "2019-12-10T11:09:35.999000-08:00"
    },
    {
      "actionName": "UpdateDeviceCertAction",
      "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/UpdateDeviceCertAction",
      "creationDate": "2019-12-10T11:08:44.263000-08:00"
    },
    {
      "actionName": "SampleMitigationAction",
      "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/SampleMitigationAction",
      "creationDate": "2019-12-10T11:03:41.840000-08:00"
    }
  ]
}
```

```
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [ListMitigationActions](#)（[缓解操作命令](#)）。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListMitigationsActions](#)中的。

## list-ota-updates

以下代码示例演示了如何使用 `list-ota-updates`。

### AWS CLI

列出账户的 OTA 更新

以下`list-ota-updates`示例列出了可用的 OTA 更新。

```
aws iot list-ota-updates
```

输出：

```
{
  "otaUpdates": [
    {
      "otaUpdateId": "itsaupdate",
      "otaUpdateArn": "arn:aws:iot:us-west-2:123456789012:otaupdate/itsaupdate",
      "creationDate": 1557863215.995
    }
  ]
}
```

有关更多信息，请参阅《物联网 API 参考》中的 [List otaUpdates](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListOtaUpdates](#)中的。

## list-outgoing-certificates

以下代码示例演示了如何使用 `list-outgoing-certificates`。

### AWS CLI

列出正在转移到其他 AWS 账户的证书

以下`list-outgoing-certificates`示例列出了正在使用`transfer-certificate`命令转移到其他 AWS 账户的所有设备证书。

```
aws iot list-outgoing-certificates
```

输出：

```
{
  "outgoingCertificates": [
    {
      "certificateArn": "arn:aws:iot:us-west-2:030714055129:cert/488b6a7f2acdeb00a77384e63c4e40b18b1b3caaae57b7272ba44c45e3448142",
      "certificateId":
      "488b6a7f2acdeb00a77384e63c4e40b18b1b3caaae57b7272ba44c45e3448142",
      "transferredTo": "030714055129",
      "transferDate": 1569427780.441,
      "creationDate": 1569363250.557
    }
  ]
}
```

有关更多信息，请参阅 AWS IoT API 参考[ListOutgoingCertificates](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListOutgoingCertificates](#)中的。

## list-policies

以下代码示例演示了如何使用 `list-policies`。

AWS CLI

列出您的 AWS 账户中定义的政策

以下`list-policies`示例列出了您的 AWS 账户中定义的所有政策。

```
aws iot list-policies
```

输出：

```
{
  "policies": [
```



```
{
  "policyName": "UpdateDeviceCertPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
UpdateDeviceCertPolicy"
},
{
  "policyName": "PlantIoTPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/PlantIoTPolicy"
},
{
  "policyName": "MyPiGroup_Core-policy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/MyPiGroup_Core-
policy"
}
]
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》中的AWS IoT 政策](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPolicies](#)中的。

## list-policy-versions

以下代码示例演示了如何使用 list-policy-versions。

### AWS CLI

示例 1：查看策略的所有版本

以下list-policy-versions示例列出了指定策略的所有版本及其创建日期。

```
aws iot list-policy-versions \
  --policy-name LightBulbPolicy
```

输出：

```
{
  "policyVersions": [
    {
      "versionId": "2",
      "isDefaultVersion": true,
      "createDate": 1559925941.924
    },
  ],
}
```

```
{
  "versionId": "1",
  "isDefaultVersion": false,
  "createDate": 1559925941.924
}
]
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》中的AWS IoT 政策](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPolicyVersions](#)中的。

## list-principal-things

以下代码示例演示了如何使用 list-principal-things。

### AWS CLI

列出校长附带的东西

以下list-principal-things示例列出了附加到由 ARN 指定的委托人的内容。

```
aws iot list-principal-things \
  --principal arn:aws:iot:us-
west-2:123456789012:cert/2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8
```

输出：

```
{
  "things": [
    "DeskLamp",
    "TableLamp"
  ]
}
```

有关更多信息，请参阅 AWS IoT API 参考[ListPrincipalThings](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPrincipalThings](#)中的。

## list-provisioning-template-versions

以下代码示例演示了如何使用 list-provisioning-template-versions。

## AWS CLI

### 列出配置模板版本

以下`list-provisioning-template-versions`示例列出了指定配置模板的可用版本。

```
aws iot list-provisioning-template-versions \  
  --template-name "widget-template"
```

输出：

```
{  
  "versions": [  
    {  
      "versionId": 1,  
      "creationDate": 1574800471.339,  
      "isDefaultVersion": true  
    },  
    {  
      "versionId": 2,  
      "creationDate": 1574801192.317,  
      "isDefaultVersion": false  
    }  
  ]  
}
```

有关更多信息，请参阅 [《AWS 物联网核心开发者指南》中的 Io AWS T 安全隧道](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListProvisioningTemplateVersions](#) 中的。

## list-provisioning-templates

以下代码示例演示了如何使用 `list-provisioning-templates`。

### AWS CLI

#### 列出配置模板

以下`list-provisioning-templates`示例列出了您 AWS 账户中的所有配置模板。

```
aws iot list-provisioning-templates
```

输出：

```
{
  "templates": [
    {
      "templateArn": "arn:aws:iot:us-east-1:123456789012:provisioningtemplate/widget-template",
      "templateName": "widget-template",
      "description": "A provisioning template for widgets",
      "creationDate": 1574800471.367,
      "lastModifiedDate": 1574801192.324,
      "enabled": false
    }
  ]
}
```

有关更多信息，请参阅《[AWS 物联网核心开发者指南](#)》中的 [Io AWS T 安全隧道](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListProvisioningTemplates](#) 中的。

## list-role-aliases

以下代码示例演示了如何使用 list-role-aliases。

### AWS CLI

列出您 AWS 账户中的 AWS IoT 角色别名

以下 list-role-aliases 示例列出了您 AWS 账户中的 AWS IoT 角色别名。

```
aws iot list-role-aliases
```

输出：

```
{
  "roleAliases": [
    "ResidentAlias",
    "ElectricianAlias"
  ]
}
```

有关更多信息，请参阅 AWS IoT API 参考 [ListRoleAliases](#) 中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRoleAliases](#)中的。

## list-scheduled-audits

以下代码示例演示了如何使用 list-scheduled-audits。

### AWS CLI

列出您 AWS 账户的预定审计

以下list-scheduled-audits示例列出了为您的 AWS 账户安排的所有审计。

```
aws iot list-scheduled-audits
```

输出：

```
{
  "scheduledAudits": [
    {
      "scheduledAuditName": "AWSIoTDeviceDefenderDailyAudit",
      "scheduledAuditArn": "arn:aws:iot:us-west-2:123456789012:scheduledaudit/AWSIoTDeviceDefenderDailyAudit",
      "frequency": "DAILY"
    },
    {
      "scheduledAuditName": "AWSDeviceDefenderWeeklyAudit",
      "scheduledAuditArn": "arn:aws:iot:us-west-2:123456789012:scheduledaudit/AWSDeviceDefenderWeeklyAudit",
      "frequency": "WEEKLY",
      "dayOfWeek": "SUN"
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[审计命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListScheduledAudits](#)中的。

## list-security-profiles-for-target

以下代码示例演示了如何使用 list-security-profiles-for-target。

## AWS CLI

列出附加到目标的安全配置文件

以下`list-security-profiles-for-target`示例列出了附加到未注册设备的 AWS IoT Device Defender 安全配置文件。

```
aws iot list-security-profiles-for-target \
  --security-profile-target-arn "arn:aws:iot:us-west-2:123456789012:all/
  unregistered-things"
```

输出：

```
{
  "securityProfileTargetMappings": [
    {
      "securityProfileIdentifier": {
        "name": "Testprofile",
        "arn": "arn:aws:iot:us-west-2:123456789012:securityprofile/
Testprofile"
      },
      "target": {
        "arn": "arn:aws:iot:us-west-2:123456789012:all/unregistered-things"
      }
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListSecurityProfilesForTarget](#)中的。

## **list-security-profiles**

以下代码示例演示了如何使用 `list-security-profiles`。

### AWS CLI

列出您 AWS 账户的安全配置文件

以下`list-security-profiles`示例列出了在您的 AWS 账户中定义的所有 AWS IoT Device Defender 安全配置文件。

```
aws iot list-security-profiles
```

输出：

```
{
  "securityProfileIdentifiers": [
    {
      "name": "Testprofile",
      "arn": "arn:aws:iot:us-west-2:123456789012:securityprofile/Testprofile"
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListSecurityProfiles](#) 中的。

## list-streams

以下代码示例演示了如何使用 list-streams。

### AWS CLI

列出账号中的直播

以下 list-streams 示例列出了您 AWS 账户中的所有直播。

```
aws iot list-streams
```

输出：

```
{
  "streams": [
    {
      "streamId": "stream12345",
      "streamArn": "arn:aws:iot:us-west-2:123456789012:stream/stream12345",
      "streamVersion": 1,
      "description": "This stream is used for Amazon FreeRTOS OTA Update
12345."
    },
    {
      "streamId": "stream54321",
```

```
        "streamArn": "arn:aws:iot:us-west-2:123456789012:stream/stream54321",
        "streamVersion": 1,
        "description": "This stream is used for Amazon FreeRTOS OTA Update
54321."
    }
  ]
}
```

有关更多信息，请参阅 AWS IoT API 参考[ListStreams](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStreams](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

显示与资源关联的标签及其值

以下`list-tags-for-resource`示例显示了与事物组关联的标签和值LightBulbs。

```
aws iot list-tags-for-resource \
  --resource-arn "arn:aws:iot:us-west-2:094249569039:thinggroup/LightBulbs"
```

输出：

```
{
  "tags": [
    {
      "Key": "Assembly",
      "Value": "Fact1NW"
    },
    {
      "Key": "MyTag",
      "Value": "777"
    }
  ]
}
```

有关更多信息，请参阅《[AWS 物联网开发者指南](#)》中的为物AWS 联网资源添加标签。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。



## list-targets-for-policy

以下代码示例演示了如何使用 `list-targets-for-policy`。

### AWS CLI

列出与 AWS IoT 策略关联的委托人

以下 `list-targets-for-policy` 示例列出了附加指定策略的设备证书。

```
aws iot list-targets-for-policy \  
  --policy-name UpdateDeviceCertPolicy
```

输出：

```
{  
  "targets": [  
    "arn:aws:iot:us-west-2:123456789012:cert/488b6a7f2acdeb00a77384e63c4e40b18b1b3caaae57b7272ba44c45e3448142",  
    "arn:aws:iot:us-west-2:123456789012:cert/  
d1eb269fb55a628552143c8f96eb3c258fcd5331ea113e766ba0c82bf225f0be"  
  ]  
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的事物组。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTargetsForPolicy](#) 中的。

## list-targets-for-security-profile

以下代码示例演示了如何使用 `list-targets-for-security-profile`。

### AWS CLI

列出应用安全配置文件的目标

以下 `list-targets-for-security-profile` 示例列出了应用名 `PossibleIssue` 为的 AWS IoT Device Defender 安全配置文件的目标。

```
aws iot list-targets-for-security-profile \  
  --security-profile-name Testprofile
```

输出：

```
{
  "securityProfileTargets": [
    {
      "arn": "arn:aws:iot:us-west-2:123456789012:all/unregistered-things"
    },
    {
      "arn": "arn:aws:iot:us-west-2:123456789012:all/registered-things"
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTargetsForSecurityProfile](#) 中的。

## list-thing-groups-for-thing

以下代码示例演示了如何使用 `list-thing-groups-for-thing`。

### AWS CLI

列出事物所属的群组

以下 `list-thing-groups-for-thing` 示例列出了指定事物所属的群组。

```
aws iot list-thing-groups-for-thing \
  --thing-name MyLightBulb
```

输出：

```
{
  "thingGroups": [
    {
      "groupName": "DeadBulbs",
      "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/DeadBulbs"
    },
    {
      "groupName": "LightBulbs",
      "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[事物组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListThingGroupsForThing](#)中的。

## list-thing-groups

以下代码示例演示了如何使用 list-thing-groups。

### AWS CLI

列出您的 AWS 账户中定义的事物组

以下describe-thing-group示例列出了您的 AWS 账户中定义的所有事物组。

```
aws iot list-thing-groups
```

输出：

```
{
  "thingGroups": [
    {
      "groupName": "HalogenBulbs",
      "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/HalogenBulbs"
    },
    {
      "groupName": "LightBulbs",
      "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[事物组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListThingGroups](#)中的。

## list-thing-principals

以下代码示例演示了如何使用 list-thing-principals。

### AWS CLI

列出与某件事相关的委托人

以下 `list-thing-principals` 示例列出了与指定事物关联的委托人 ( X.509 证书、IAM 用户、群组、角色、Amazon Cognito 身份或联合身份 )。

```
aws iot list-thing-principals \  
  --thing-name MyRaspberryPi
```

输出：

```
{  
  "principals": [  
    "arn:aws:iot:us-west-2:123456789012:cert/33475ac865079a5ffd5ecd44240640349293facc760642d7d8d5dbb6b4c86893"  
  ]  
}
```

有关更多信息，请参阅 AWS IoT API 参考 [ListThingPrincipals](#) 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListThingPrincipals](#) 中的。

## list-thing-types

以下代码示例演示了如何使用 `list-thing-types`。

### AWS CLI

列出已定义的事物类型

以下 `list-thing-types` 示例显示了您的 AWS 账户中定义的事物类型列表。

```
aws iot list-thing-types
```

输出：

```
{  
  "thingTypes": [  
    {  
      "thingTypeName": "LightBulb",  
      "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/  
LightBulb",  
      "thingTypeProperties": {  
        "thingTypeDescription": "light bulb type",  

```

```

        "searchableAttributes": [
            "model",
            "wattage"
        ]
    },
    "thingTypeMetadata": {
        "deprecated": false,
        "creationDate": 1559772562.498
    }
}
]
}

```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的事物[类型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListThingTypes](#)中的。

## list-things-in-billing-group

以下代码示例演示了如何使用 list-things-in-billing-group。

### AWS CLI

列出账单组中的内容

以下list-things-in-billing-group示例列出了指定账单组中的内容。

```
aws iot list-things-in-billing-group \
  --billing-group-name GroupOne
```

输出：

```
{
  "things": [
    "MyOtherLightBulb",
    "MyLightBulb"
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[账单组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListThingsInBillingGroup](#)中的。

## list-things-in-thing-group

以下代码示例演示了如何使用 `list-things-in-thing-group`。

### AWS CLI

列出属于某个群组的事物

以下 `list-things-in-thing-group` 示例列出了属于指定事物组的事物。

```
aws iot list-things-in-thing-group \  
  --thing-group-name LightBulbs
```

输出：

```
{  
  "things": [  
    "MyLightBulb"  
  ]  
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的事物组。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListThingsInThingGroup](#) 中的。

## list-things

以下代码示例演示了如何使用 `list-things`。

### AWS CLI

示例 1：列出注册表中的所有事物

以下 `list-things` 示例列出了在您的 AWS 账户的 AWS IoT 注册表中定义的事物（设备）。

```
aws iot list-things
```

输出：

```
{  
  "things": [  
    {  
      "thingName": "ThirdBulb",
```

```
    "thingTypeName": "LightBulb",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/ThirdBulb",
    "attributes": {
      "model": "123",
      "wattage": "75"
    },
    "version": 2
  },
  {
    "thingName": "MyOtherLightBulb",
    "thingTypeName": "LightBulb",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyOtherLightBulb",
    "attributes": {
      "model": "123",
      "wattage": "75"
    },
    "version": 3
  },
  {
    "thingName": "MyLightBulb",
    "thingTypeName": "LightBulb",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "attributes": {
      "model": "123",
      "wattage": "75"
    },
    "version": 1
  },
  {
    "thingName": "SampleIoTThing",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/SampleIoTThing",
    "attributes": {},
    "version": 1
  }
]
```

示例 2：列出具有特定属性的已定义事物

以下 `list-things` 示例显示了具有名为 `wattage` 的属性的事物列表。

```
aws iot list-things \
  --attribute-name wattage
```

输出：

```
{
  "things": [
    {
      "thingName": "MyLightBulb",
      "thingTypeName": "LightBulb",
      "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
      "attributes": {
        "model": "123",
        "wattage": "75"
      },
      "version": 1
    },
    {
      "thingName": "MyOtherLightBulb",
      "thingTypeName": "LightBulb",
      "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyOtherLightBulb",
      "attributes": {
        "model": "123",
        "wattage": "75"
      },
      "version": 3
    }
  ]
}
```

有关更多信息，请参阅《AWS IoT 开发人员指南》中的[如何使用注册表管理事物](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListThings](#)中的。

## list-topic-rule-destinations

以下代码示例演示了如何使用 list-topic-rule-destinations。

AWS CLI

列出您的主题规则目的地

以下list-topic-rule-destinations示例列出了您在当前 AWS 区域中定义的所有主题规则目的地。

```
aws iot list-topic-rule-destinations
```



输出：

```
{
  "destinationSummaries": [
    {
      "arn": "arn:aws:iot:us-west-2:123456789012:ruledestination/http/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "status": "ENABLED",
      "httpUrlSummary": {
        "confirmationUrl": "https://example.com"
      }
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[使用主题规则目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTopicRuleDestinations](#)中的。

## list-topic-rules

以下代码示例演示了如何使用 list-topic-rules。

### AWS CLI

列出您的规则

以下list-topic-rules示例列出了您已定义的所有规则。

```
aws iot list-topic-rules
```

输出：

```
{
  "rules": [
    {
      "ruleArn": "arn:aws:iot:us-west-2:123456789012:rule/
MyRPiLowMoistureAlertRule",
      "ruleName": "MyRPiLowMoistureAlertRule",
      "topicPattern": "$aws/things/MyRPi/shadow/update/accepted",
      "createdAt": 1558624363.0,
      "ruleDisabled": false
    },
  ],
}
```

```
{
  "ruleArn": "arn:aws:iot:us-west-2:123456789012:rule/
MyPlantPiMoistureAlertRule",
  "ruleName": "MyPlantPiMoistureAlertRule",
  "topicPattern": "$aws/things/MyPlantPi/shadow/update/accepted",
  "createdAt": 1541458459.0,
  "ruleDisabled": false
}
]
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[查看您的规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTopicRules](#) 中的。

## list-v2-logging-levels

以下代码示例演示了如何使用 `list-v2-logging-levels`。

### AWS CLI

#### 列出日志级别

以下 `list-v2-logging-levels` 示例列出了配置的日志级别。如果未设置日志级别，则运行此命令时 `NotConfiguredException` 会出现。

```
aws iot list-v2-logging-levels
```

输出：

```
{
  "logTargetConfigurations": [
    {
      "logTarget": {
        "targetType": "DEFAULT"
      },
      "logLevel": "ERROR"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》LoggingLevels 中的 [ListV2](#)。

## list-violation-events

以下代码示例演示了如何使用 `list-violation-events`。

### AWS CLI

列出一段时间内违反安全配置文件的情况

以下 `list-violation-events` 示例列出了当前 AWS 账户和 AWS 地区的所有 AWS IoT Device Defender 安全配置文件在 2019 年 6 月 5 日至 2019 年 6 月 12 日期间发生的违规行为。

```
aws iot list-violation-events \  
  --start-time 1559747125 \  
  --end-time 1560351925
```

输出：

```
{  
  "violationEvents": [  
    {  
      "violationId": "174db59167fa474c80a652ad1583fd44",  
      "thingName": "iotconsole-1560269126751-1",  
      "securityProfileName": "Testprofile",  
      "behavior": {  
        "name": "Authorization",  
        "metric": "aws:num-authorization-failures",  
        "criteria": {  
          "comparisonOperator": "greater-than",  
          "value": {  
            "count": 10  
          },  
          "durationSeconds": 300,  
          "consecutiveDatapointsToAlarm": 1,  
          "consecutiveDatapointsToClear": 1  
        }  
      },  
      "metricValue": {  
        "count": 0  
      },  
      "violationEventType": "in-alarm",  
      "violationEventTime": 1560279000.0  
    },  
  ]  
}
```

```
"violationId": "c8a9466a093d3b7b35cd44ca58bdbbeab",
"thingName": "TvnQoEoU",
"securityProfileName": "Testprofile",
"behavior": {
  "name": "CellularBandwidth",
  "metric": "aws:message-byte-size",
  "criteria": {
    "comparisonOperator": "greater-than",
    "value": {
      "count": 128
    },
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  }
},
"metricValue": {
  "count": 110
},
"violationEventType": "in-alarm",
"violationEventTime": 1560276600.0
},
{
  "violationId": "74aa393adea02e6648f3ac362beed55e",
  "thingName": "iotconsole-1560269232412-2",
  "securityProfileName": "Testprofile",
  "behavior": {
    "name": "Authorization",
    "metric": "aws:num-authorization-failures",
    "criteria": {
      "comparisonOperator": "greater-than",
      "value": {
        "count": 10
      },
      "durationSeconds": 300,
      "consecutiveDatapointsToAlarm": 1,
      "consecutiveDatapointsToClear": 1
    }
  },
  "metricValue": {
    "count": 0
  },
  "violationEventType": "in-alarm",
  "violationEventTime": 1560276600.0
},
```

```
{
  "violationId": "1e6ab5f7cf39a1466fcd154e1377e406",
  "thingName": "TvnQoEoU",
  "securityProfileName": "Testprofile",
  "behavior": {
    "name": "Authorization",
    "metric": "aws:num-authorization-failures",
    "criteria": {
      "comparisonOperator": "greater-than",
      "value": {
        "count": 10
      },
      "durationSeconds": 300,
      "consecutiveDatapointsToAlarm": 1,
      "consecutiveDatapointsToClear": 1
    }
  },
  "metricValue": {
    "count": 0
  },
  "violationEventType": "in-alarm",
  "violationEventTime": 1560276600.0
}
]
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListViolationEvents](#)中的。

## register-ca-certificate

以下代码示例演示了如何使用 register-ca-certificate。

### AWS CLI

#### 注册证书颁发机构 (CA) 证书

以下register-ca-certificate示例注册了一个 CA 证书。该命令提供 CA 证书和密钥验证证书，证明您拥有与 CA 证书关联的私钥。

```
aws iot register-ca-certificate \
  --ca-certificate file://rootCA.pem \
```

```
--verification-cert file://verificationCert.pem
```

输出：

```
{
  "certificateArn": "arn:aws:iot:us-west-2:123456789012:cacert/
f4efed62c0142f16af278166f61962501165c4f0536295207426460058cd1467",
  "certificateId":
"f4efed62c0142f16af278166f61962501165c4f0536295207426460058cd1467"
}
```

有关更多信息，请参阅《物联网 API 参考》中的 [RegisterCertificate](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterCertificate](#) 中的。

## register-certificate

以下代码示例演示了如何使用 register-certificate。

### AWS CLI

#### 注册自签名设备证书

以下 register-certificate 示例注册由 rootCA.pem CA 证书签名的 deviceCert.pem 设备证书。必须先注册 CA 证书，然后才能使用它来注册自签名设备证书。自签名证书必须由您传递给此命令的相同 CA 证书签名。

```
aws iot register-certificate \
  --certificate-pem file://deviceCert.pem \
  --ca-certificate-pem file://rootCA.pem
```

输出：

```
{
  "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/488b6a7f2acdeb00a77384e63c4e40b18b1b3caaae57b7272ba44c45e3448142",
  "certificateId":
"488b6a7f2acdeb00a77384e63c4e40b18b1b3caaae57b7272ba44c45e3448142"
}
```

有关更多信息，请参阅 AWS IoT API 参考 [RegisterCertificate](#) 中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterCertificate](#)中的。

## register-thing

以下代码示例演示了如何使用 register-thing。

### AWS CLI

#### 注册事物

以下register-thing示例使用配置模板注册事物。

```
aws iot register-thing \
  --template-body '{"Parameters":{"ThingName":
{"Type":"String"},"AWS::IoT::Certificate::Id":{"Type":"String"}},'Resources':
{"certificate":{"Properties":{"CertificateId":
{"Ref":"AWS::IoT::Certificate::Id"},"Status":"Active"},"Type":"AWS::IoT::Certificate"},"poli
{"Properties":{"PolicyName":"MyIotPolicy"},"Type":"AWS::IoT::Policy"},"thing":
{"OverrideSettings":
{"AttributePayload":"MERGE","ThingGroups":"DO_NOTHING","ThingTypeName":"REPLACE"},"Propertie
{"AttributePayload":{},"ThingGroups":[],"ThingName":
{"Ref":"ThingName"},"ThingTypeName":"VirtualThings"},"Type":"AWS::IoT::Thing"}}}' \
  --parameters '{"ThingName":"Register-thing-
trial-1","AWS::IoT::Certificate::Id":"799a9ea048a1e6aea42b55EXAMPLEf8697b4bafcd77a318a3068e3
```

输出：

```
{
  "certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCCAKGgAwIBAgIUYLk81I35cIppobpw
Hi0J2jNjboIwDQYJKoZIhvcNAQEL
\nBQAwTTFLEkGA1UECwxQW1hem9uIFdlYiBTZXJ2aWNlcyBPPUftYXpvbi
5jb20g\nSW5jLiBMPVNlYXR0bGUgU1Q9V2FzaGluZ3RvbiBDPVVTMB4XDTIwMDcyMzE2NDUw
\n0VoXDTQ5MTIzMT
IzNTk10VowHjEcMBoGA1UEAwTQVdTIElvcyBDbDZlX0aWZpY2F0\nZTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCC
AQoCggEBA071uADhdBajqTmgrMV5\nmCFfBZQRMo1MdtVoZr2X+M4MzL
+RARrtUzH9a2SMAckeX8Keb1I0TKz0RI
RDXnyE
\n6lV0wjgAsd0ku22rFxex4eG2ikha7pYYkvuToqA7L3TxItRvfKrxRI4ZfJoFPip4\nkQiuBJVNOGKTcQ
Hd1RN0rddwwu6kFJLeKdMEXAMPLEdUF0N+qfR9yKnZQkm
+g6Q2\nGXu7u0W3hn6n1RN8qVoka0uW12p53xM7oHVz
Gf+cxKBx1b0hGkp6yCfTskUBm3Sp\n9zLw35kiHXVm4EVpwn1nk6XcIGIkw8a/iy4pzmvuGAANY1/uU/
zgCjymw
```

```
ZT5S30\nBV0CAwEAAaNgMF4wHwYDVR0jBBgwFoAUGx0tCcU3q2n1WXAuUCv6hugXjKswHQYD
\nVR00BBYEF0VtvZ
9Aj2RYFnkX7Iu01XTRUdxgMAwGA1UdEwEB/wQCMAAwDgYDVR0P\nAQH/
BAQDAgeAMA0GCSqGSIB3DQEBCwUAA4IB
AQXCQCcp0tubS5ft0sDMTcP/jNX
\nDHyArxmjpSc2aCdmm7WX591TKWyAdxGAvqaDVWqTo0oXI7tZ8w7aINlGi5
pXnifx\n3SBebMUoBbTktrC97yUaeL025mCFv8emDnTR/fe7PTsBKjW0g/rrfpwBxZLXDFwN
\nnqkQjy3EDfifj2
6j0xYIqqWMPogyn4sr0CKynS5wMJuQZ1HQ0nabVwnwK4Y0Mf1p
\np9+4susFUR9aT3BT1AcIwqSpzh1Khh4Iz7ND
kRn4amsUT210jg/z0010w+BTHcVQ\nJly8XDu0CWSu04q6SnaBzHmlySIajxuRTP/AdfRouP10Xe
+q1bPOBcvVvF
8o\n-----END CERTIFICATE-----\n",
  "resourceArns": {
    "certificate": "arn:aws:iot:us-
west-2:571032923833:cert/799a9ea048a1e6aea42b55EXAMPLEf8697b4bafcd77a318a3068e30404b9233c",
    "thing": "arn:aws:iot:us-west-2:571032923833:thing/Register-thing-trial-1"
  }
}
```

有关更多信息，请参阅《AWS IoT Core 开发人员指南》中的[可信用户配置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterThing](#)中的。

## reject-certificate-transfer

以下代码示例演示了如何使用 reject-certificate-transfer。

### AWS CLI

#### 拒绝证书转移

以下reject-certificate-transfer示例拒绝从其他 AWS 账户转移指定的设备证书。

```
aws iot reject-certificate-transfer \
  --certificate-id
  f0f33678c7c9a046e5cc87b2b1a58dfa0beec26db78add5e605d630e05c7fc8
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Core 开发者指南》中的[将证书转移到其他账户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RejectCertificateTransfer](#)中的。



## remove-thing-from-billing-group

以下代码示例演示了如何使用 `remove-thing-from-billing-group`。

### AWS CLI

从账单组中移除某件事

以下 `remove-thing-from-billing-group` 示例将指定的内容从账单组中移除。

```
aws iot remove-thing-from-billing-group \  
  --billing-group-name GroupOne \  
  --thing-name MyOtherLightBulb
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[账单组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RemoveThingFromBillingGroup](#) 中的。

## remove-thing-from-thing-group

以下代码示例演示了如何使用 `remove-thing-from-thing-group`。

### AWS CLI

从事物组中移除事物

以下 `remove-thing-from-thing-group` 示例将指定的事物从事物组中移除。

```
aws iot remove-thing-from-thing-group \  
  --thing-name bulb7 \  
  --thing-group-name DeadBulbs
```

此命令不生成任何输出。

有关更多信息，请参阅 < <https://docs.aws.amazon.com/iot/latest/developerguide/thing-groups.html> > 《AWS 物联网开发者指南》中的事物组。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RemoveThingFromThingGroup](#) 中的。

## replace-topic-rule

以下代码示例演示了如何使用 `replace-topic-rule`。

### AWS CLI

#### 更新主题的规则定义

以下 `replace-topic-rule` 示例更新了指定的规则，以便在土壤湿度读数过低时发送 SNS 警报。

```
aws iot replace-topic-rule \  
  --rule-name MyRPiLowMoistureAlertRule \  
  --topic-rule-payload "{\"sql\": \"SELECT * FROM '$aws/things/MyRPi/shadow/  
update/accepted' WHERE state.reported.moisture = 'low'\", \"description\": \"Sends  
an alert when soil moisture level readings are too low.\", \"actions\": [{\"sns  
\": {\"targetArn\": \"arn:aws:sns:us-west-2:123456789012:MyRPiLowMoistureTopic\",  
\"roleArn\": \"arn:aws:iam:123456789012:role/service-role/MyRPiLowMoistureTopicRole  
\", \"messageFormat\": \"RAW\"}}], \"ruleDisabled\": false, \"awsIotSqlVersion\":  
\"2016-03-23\"}"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网开发者指南](#)》中的[创建AWS 物联网规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ReplaceTopicRule](#) 中的。

## search-index

以下代码示例演示了如何使用 `search-index`。

### AWS CLI

#### 查询事物索引

以下 `search-index` 示例在 `AWS_Things` 索引中查询类型为事物 `LightBulb`。

```
aws iot search-index \  
  --index-name "AWS_Things" \  
  --query-string "thingTypeName:LightBulb"
```

输出：

```
{
```

```
"things": [
  {
    "thingName": "MyLightBulb",
    "thingId": "40da2e73-c6af-406e-b415-15acae538797",
    "thingTypeName": "LightBulb",
    "thingGroupNames": [
      "LightBulbs",
      "DeadBulbs"
    ],
    "attributes": {
      "model": "123",
      "wattage": "75"
    },
    "connectivity": {
      "connected": false
    }
  },
  {
    "thingName": "ThirdBulb",
    "thingId": "615c8455-33d5-40e8-95fd-3ee8b24490af",
    "thingTypeName": "LightBulb",
    "attributes": {
      "model": "123",
      "wattage": "75"
    },
    "connectivity": {
      "connected": false
    }
  },
  {
    "thingName": "MyOtherLightBulb",
    "thingId": "6dae0d3f-40c1-476a-80c4-1ed24ba6aa11",
    "thingTypeName": "LightBulb",
    "attributes": {
      "model": "123",
      "wattage": "75"
    },
    "connectivity": {
      "connected": false
    }
  }
]
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的管理事物[索引](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SearchIndex](#)中的。

## set-default-authorizer

以下代码示例演示了如何使用 set-default-authorizer。

### AWS CLI

#### 设置默认授权者

以下set-default-authorizer示例将名为默认授权方的自定义授权方设置CustomAuthorizer为默认授权方。

```
aws iot set-default-authorizer \  
  --authorizer-name CustomAuthorizer
```

输出：

```
{  
  "authorizerName": "CustomAuthorizer",  
  "authorizerArn": "arn:aws:iot:us-west-2:123456789012:authorizer/  
CustomAuthorizer"  
}
```

有关更多信息，请参阅 AWS IoT API 参考[CreateDefaultAuthorizer](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetDefaultAuthorizer](#)中的。

## set-default-policy-version

以下代码示例演示了如何使用 set-default-policy-version。

### AWS CLI

#### 为策略设置默认版本

以下set-default-policy-version示例将名2为的策略的默认版本设置为UpdateDeviceCertPolicy。

```
aws iot set-default-policy-version \  
  --policy-name UpdateDeviceCertPolicy
```

```
--policy-name UpdateDeviceCertPolicy \  
--policy-version-id 2
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetDefaultPolicyVersion](#)中的。

## set-v2-logging-level

以下代码示例演示了如何使用 set-v2-logging-level。

### AWS CLI

为事物组设置日志级别

以下set-v2-logging-level示例将日志级别设置为记录指定事物组的警告。

```
aws iot set-v2-logging-level \  
  --log-target "{\"targetType\":\"THING_GROUP\",\"targetName\":\"LightBulbs\"}" \  
  --log-level WARN
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考LoggingLevel》中的 [setV2](#)。

## set-v2-logging-options

以下代码示例演示了如何使用 set-v2-logging-options。

### AWS CLI

设置日志选项

以下set-v2-logging-options示例将默认的日志详细级别设置为 ERROR，并指定用于记录的 ARN。

```
aws iot set-v2-logging-options \  
  --default-log-level ERROR \  
  --role-arn "arn:aws:iam::094249569039:role/service-role/iotLoggingRole"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考LoggingOptions》中的 [setV2](#)。

## start-audit-mitigation-actions-task

以下代码示例演示了如何使用 start-audit-mitigation-actions-task。

### AWS CLI

对审计结果采取缓解措施

以下start-audit-mitigation-actions-task示例将ResetPolicyVersionAction操作（清除策略）应用于指定的单个查找结果。

```
aws iot start-audit-mitigation-actions-task \
  --task-id "myActionsTaskId" \
  --target "findingIds=[\"0edbaaec-2fe1-4cf5-abc9-d4c3e51f7464\"]" \
  --audit-check-to-actions-mapping
  "IOT_POLICY_OVERLY_PERMISSIVE_CHECK=[\"ResetPolicyVersionAction\"]" \
  --client-request-token "adhadhahda"
```

输出：

```
{
  "taskId": "myActionsTaskId"
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [StartAuditMitigationActionsTask（缓解操作命令）](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartAuditMitigationActionsTask](#)中的。

## start-on-demand-audit-task

以下代码示例演示了如何使用 start-on-demand-audit-task。

### AWS CLI

立即开始审计

以下start-on-demand-audit-task示例启动 AWS IoT Device Defender 审核并执行三次证书检查。

```
aws iot start-on-demand-audit-task \
```

```
--target-check-names CA_CERTIFICATE_EXPIRING_CHECK  
DEVICE_CERTIFICATE_EXPIRING_CHECK REVOKED_CA_CERTIFICATE_STILL_ACTIVE_CHECK
```

输出：

```
{  
  "taskId": "a3aea009955e501a31b764abe1bebd3d"  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[审计命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartOnDemandAuditTask](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源指定标签键和值

以下tag-resource示例将带有键Assembly和值的标签应用Fact1NW于事物组LightBulbs。

```
aws iot tag-resource \  
  --tags Key=Assembly,Value="Fact1NW" \  
  --resource-arn "arn:aws:iot:us-west-2:094249569039:thinggroup/LightBulbs"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网开发者指南](#)》中的[为物AWS 联网资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## test-authorization

以下代码示例演示了如何使用 test-authorization。

### AWS CLI

测试您的 AWS 物联网政策

以下test-authorization示例测试与指定委托人关联的 AWS IoT 策略。

```
aws iot test-authorization \  
  --auth-infos actionType=CONNECT,resources=arn:aws:iot:us-  
east-1:123456789012:client/client1 \  
  --principal arn:aws:iot:us-west-2:123456789012:cert/  
aab1068f7f43ac3e3cae4b3a8aa3f308d2a750e6350507962e32c1eb465d9775
```

输出：

```
{  
  "authResults": [  
    {  
      "authInfo": {  
        "actionType": "CONNECT",  
        "resources": [  
          "arn:aws:iot:us-east-1:123456789012:client/client1"  
        ]  
      },  
      "allowed": {  
        "policies": [  
          {  
            "policyName": "TestPolicyAllowed",  
            "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/  
TestPolicyAllowed"  
          }  
        ]  
      },  
      "denied": {  
        "implicitDeny": {  
          "policies": [  
            {  
              "policyName": "TestPolicyDenied",  
              "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/  
TestPolicyDenied"  
            }  
          ]  
        },  
        "explicitDeny": {  
          "policies": [  
            {  
              "policyName": "TestPolicyExplicitDenied",  
              "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/  
TestPolicyExplicitDenied"  
            }  
          ]  
        }  
      }  
    ]  
  }  
}
```



```

    ]
  },
  "authDecision": "IMPLICIT_DENY",
  "missingContextValues": []
}
]
}

```

有关更多信息，请参阅 AWS IoT API 参考[TestAuthorization](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[TestAuthorization](#)中的。

## test-invoke-authorizer

以下代码示例演示了如何使用 test-invoke-authorizer。

### AWS CLI

测试您的自定义授权方

以下 test-invoke-authorizer 示例测试您的自定义授权方。

```

aws iot test-invoke-authorizer \
  --authorizer-name IoTAuthorizer \
  --token allow \
  --token-signature "mE0GvaHqy9nER/
FdgtJX51XYEJ3b3vE7t1gEszc0TKGgLKWXTnPk2AbKn0AZ81GyoN5dVtWDWVmr25m7+
+zjbYIMk2TBvyGXh0mvKFBPkdgyA43KL6SiZy0cTq1PMcQDsP7VX2rXr7CTowCxSNKphGXdQe0/
I5dQ+J06KUaHwCmupt0/MejKtaNwiiA064j6wpr0AUwG5S1IYFuRd0X
+wfo8pb0DubAIX1Ua705kuhRUCtx4SxUSHEyKmN4IDEvLB6FsIr0B2wvB7y4iPmcajxzG102ExvyCUNctCV9dY1RRGJj

```

输出：

```

{
  "isAuthenticated": true,
  "principalId": "principalId",
  "policyDocuments": [
    {"Version": "2012-10-17", "Statement":
[{"Action": "iot:Publish", "Effect": "Allow", "Resource": "arn:aws:iot:us-
west-2:123456789012:topic/customauthtesting"}]}]
  },
  "refreshAfterInSeconds": 600,

```

```
"disconnectAfterInSeconds": 3600
}
```

有关更多信息，请参阅 AWS IoT API 参考[TestInvokeAuthorizer](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[TestInvokeAuthorizer](#)中的。

## transfer-certificate

以下代码示例演示了如何使用 transfer-certificate。

### AWS CLI

将设备证书转移到其他 AWS 账户

以下 transfer-certificate 示例将设备证书转移到另一个 AWS 账户。证书和 AWS 账户由 ID 标识。

```
aws iot transfer-certificate \
  --certificate-id
  488b6a7f2acdeb00a77384e63c4e40b18b1b3caaae57b7272ba44c45e3448142 \
  --target-aws-account 030714055129
```

输出：

```
{
  "transferredCertificateArn": "arn:aws:iot:us-
west-2:030714055129:cert/488b6a7f2acdeb00a77384e63c4e40b18b1b3caaae57b7272ba44c45e3448142"
}
```

有关更多信息，请参阅《AWS IoT Core 开发者指南》中的将[证书转移到其他账户](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[TransferCertificate](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从资源中移除标签密钥

以下 `untag-resource` 示例将标签 `MyTag` 及其值从事物组中移除 `LightBulbs`。

```
command
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网开发者指南](#)》中的[为物AWS 联网资源添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-account-audit-configuration

以下代码示例演示了如何使用 `update-account-audit-configuration`。

### AWS CLI

示例 1：为审计通知启用 Amazon SNS 通知

以下 `update-account-audit-configuration` 示例为 AWS IoT Device Defender 审计通知启用 Amazon SNS 通知，指定目标和用于写入该目标的角色。

```
aws iot update-account-audit-configuration \  
  --audit-notification-target-configurations "SNS={targetArn=\"arn:aws:sns:us-  
west-2:123456789012:ddauidits\",roleArn=\"arn:aws:iam::123456789012:role/service-  
role/AWSIoTDeviceDefenderAudit\",enabled=true}"
```

此命令不生成任何输出。

示例 2：启用审计检查

以下 `update-account-audit-configuration` 示例启用名为的 AWS IoT Device Defender 审核检查 `AUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK`。如果审计检查是 AWS 账户一项或多项计划审计的一部分，则不能将其禁用。 `targetCheckNames`

```
aws iot update-account-audit-configuration \  
  --audit-check-configurations  
  "{\"AUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK\":{\"enabled\":true}}"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网开发者指南](#)》中的[审计命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAccountAuditConfiguration](#)中的。

## update-audit-suppression

以下代码示例演示了如何使用 update-audit-suppression。

### AWS CLI

#### 更新审计结果隐藏

以下update-audit-suppression示例将审计结果抑制的到期日期更新为 2020-09-21。

```
aws iot update-audit-suppression \  
  --check-name DEVICE_CERTIFICATE_EXPIRING_CHECK \  
  --resource-identifier deviceCertificateId=c7691e<shortened> \  
  --no-suppress-indefinitely \  
  --expiration-date 2020-09-21
```

此命令不生成任何输出。

有关更多信息，请参阅《物AWS 联网开发人员指南》中的“[审计发现抑制](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAuditSuppression](#)中的。

## update-authorizer

以下代码示例演示了如何使用 update-authorizer。

### AWS CLI

#### 更新自定义授权方

在下面的update-authorizer例子中，他说CustomAuthorizer2的是INACTIVE。

```
aws iot update-authorizer \  
  --authorizer-name CustomAuthorizer2 \  
  --status INACTIVE
```

输出：

```
{  
  "authorizerName": "CustomAuthorizer2",
```

```
"authorizerArn": "arn:aws:iot:us-west-2:123456789012:authorizer/
CustomAuthorizer2"
}
```

有关更多信息，请参阅 AWS IoT API 参考[UpdateAuthorizer](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAuthorizer](#)中的。

## update-billing-group

以下代码示例演示了如何使用 update-billing-group。

### AWS CLI

更新账单组的相关信息

以下update-billing-group示例更新了指定账单组的描述。

```
aws iot update-billing-group \
  --billing-group-name GroupOne \
  --billing-group-properties "billingGroupDescription=\"Primary bulb billing group
\""
```

输出：

```
{
  "version": 2
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[账单组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateBillingGroup](#)中的。

## update-ca-certificate

以下代码示例演示了如何使用 update-ca-certificate。

### AWS CLI

更新证书颁发机构 (CA) 证书

以下update-ca-certificate示例将指定的 CA 证书设置为活动状态。

```
aws iot update-ca-certificate \  
  --certificate-id  
  f4efed62c0142f16af278166f61962501165c4f0536295207426460058cd1467 \  
  --new-status ACTIVE
```

此命令不生成任何输出。

有关更多信息，请参阅《物联网 API 参考》中的 [updateCertificate](#) 更新证书。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateCaCertificate](#)中的。

## update-certificate

以下代码示例演示了如何使用 update-certificate。

### AWS CLI

#### 更新设备证书

以下update-certificate示例将指定的设备证书设置为非活动状态。

```
aws iot update-certificate \  
  --certificate-id  
  d1eb269fb55a628552143c8f96eb3c258fcd5331ea113e766ba0c82bf225f0be \  
  --new-status INACTIVE
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT API 参考[UpdateCertificate](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateCertificate](#)中的。

## update-custom-metric

以下代码示例演示了如何使用 update-custom-metric。

### AWS CLI

#### 更新自定义指标

以下update-custom-metric示例将自定义指标更新为新指标display-name。

```
aws iot update-custom-metric \  
  --metric-name batteryPercentage \  
  --display-name 'remaining battery percentage on device' \  
  --region us-east-1
```

输出：

```
{  
  "metricName": "batteryPercentage",  
  "metricArn": "arn:aws:iot:us-east-1:1234564789012:custommetric/  
batteryPercentage",  
  "metricType": "number",  
  "displayName": "remaining battery percentage on device",  
  "creationDate": "2020-11-17T23:01:35.110000-08:00",  
  "lastModifiedDate": "2020-11-17T23:02:12.879000-08:00"  
}
```

有关更多信息，请参阅 AWS IoT Core 开发者指南中的[自定义指标](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateCustomMetric](#)中的。

## update-dimension

以下代码示例演示了如何使用 update-dimension。

### AWS CLI

#### 更新维度

以下 update-dimension 示例更新维度。

```
aws iot update-dimension \  
  --name TopicFilterForAuthMessages \  
  --string-values device/${iot:ClientId}/auth
```

输出：

```
{  
  "name": "TopicFilterForAuthMessages",  
  "lastModifiedDate": 1585866222.317,  
  "stringValues": [  
    "device/${iot:ClientId}/auth"  
  ]  
}
```

```
    "device/${iot:ClientId}/auth"  
  ],  
  "creationDate": 1585854500.474,  
  "type": "TOPIC_FILTER",  
  "arn": "arn:aws:iot:us-west-2:1234564789012:dimension/  
TopicFilterForAuthMessages"  
}
```

有关更多信息，请参阅《AWS IoT Core 开发者指南》中的[使用维度对安全配置文件中的指标进行限定范围](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDimension](#)中的。

## update-domain-configuration

以下代码示例演示了如何使用 update-domain-configuration。

### AWS CLI

#### 更新域配置

以下update-domain-configuration示例禁用了指定的域配置。

```
aws iot update-domain-configuration \  
  --domain-configuration-name "additionalDataDomain" \  
  --domain-configuration-status "DISABLED"
```

输出：

```
{  
  "domainConfigurationName": "additionalDataDomain",  
  "domainConfigurationArn": "arn:aws:iot:us-  
west-2:123456789012:domainconfiguration/additionalDataDomain/dikMh"  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[可配置终端节点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDomainConfiguration](#)中的。

## update-dynamic-thing-group

以下代码示例演示了如何使用 update-dynamic-thing-group。



## AWS CLI

### 更新动态事物组

以下update-dynamic-thing-group示例更新了指定的动态事物组。它提供描述并更新查询字符串以更改组成员资格标准。

```
aws iot update-dynamic-thing-group \  
  --thing-group-name "RoomTooWarm" \  
  --thing-group-properties "thingGroupDescription=\"This thing group contains rooms warmer than 65F.\"\" \  
  --query-string "attributes.temperature>65"
```

输出：

```
{  
  "version": 2  
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[动态事物组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDynamicThingGroup](#)中的。

## update-event-configurations

以下代码示例演示了如何使用 update-event-configurations。

## AWS CLI

### 显示已发布的事件类型

以下update-event-configurations示例更新配置以在添加、更新或删除 CA 证书时启用消息。

```
aws iot update-event-configurations \  
  --event-configurations "{\"CA_CERTIFICATE\":{\"Enabled\":true}}"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[事件消息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateEventConfigurations](#)中的。

## update-indexing-configuration

以下代码示例演示了如何使用 update-indexing-configuration。

### AWS CLI

#### 启用事物索引

以下update-indexing-configuration示例启用事物索引，以支持使用 AWS\_Things 索引搜索注册表数据、影子数据和事物连接状态。

```
aws iot update-indexing-configuration
  --thing-indexing-configuration
  thingIndexingMode=REGISTRY_AND_SHADOW,thingConnectivityIndexingMode=STATUS
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发人员指南》中的管理事物[索引](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateIndexingConfiguration](#)中的。

## update-job

以下代码示例演示了如何使用 update-job。

### AWS CLI

#### 获取任务的详细状态

以下update-job示例获取 ID 为的任务的详细状态example-job-01。

```
aws iot describe-job \
  --job-id "example-job-01"
```

输出：

```
{
  "job": {
    "jobArn": "arn:aws:iot:us-west-2:123456789012:job/example-job-01",
    "jobId": "example-job-01",
    "targetSelection": "SNAPSHOT",
    "status": "IN_PROGRESS",
    "targets": [
```

```

        "arn:aws:iot:us-west-2:123456789012:thing/MyRaspberryPi"
    ],
    "description": "example job test",
    "presignedUrlConfig": {},
    "jobExecutionsRolloutConfig": {},
    "createdAt": 1560787022.733,
    "lastUpdatedAt": 1560787026.294,
    "jobProcessDetails": {
        "numberOfCanceledThings": 0,
        "numberOfSucceededThings": 0,
        "numberOfFailedThings": 0,
        "numberOfRejectedThings": 0,
        "numberOfQueuedThings": 1,
        "numberOfInProgressThings": 0,
        "numberOfRemovedThings": 0,
        "numberOfTimedOutThings": 0
    },
    "timeoutConfig": {}
}
}

```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[创建和管理作业 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateJob](#)中的。

## update-mitigation-action

以下代码示例演示了如何使用 update-mitigation-action。

### AWS CLI

#### 更新缓解措施

以下update-mitigation-action示例更新名为的指定缓解操作AddThingsToQuarantineGroupAction，更改事物组名称并将其设置overrideDynamicGroups为false。您可以使用describe-mitigation-action命令验证您的更改。

```

aws iot update-mitigation-action \
  --cli-input-json "{ \"actionName\": \"AddThingsToQuarantineGroupAction\",
  \"actionParams\": { \"addThingsToThingGroupParams\": {\"thingGroupNames\":
  [\"QuarantineGroup2\"],\"overrideDynamicGroups\": false}}}"

```

输出：

```
{
  "actionArn": "arn:aws:iot:us-west-2:123456789012:mitigationaction/
AddThingsToQuarantineGroupAction",
  "actionId": "2fd2726d-98e1-4abf-b10f-09465ccd6bfa"
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的 [UpdateMitigationAction \(缓解操作命令\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateMitigationAction](#)中的。

## update-provisioning-template

以下代码示例演示了如何使用 update-provisioning-template。

### AWS CLI

#### 更新配置模板

以下update-provisioning-template示例修改了指定配置模板的描述和角色 arn，并启用了该模板。

```
aws iot update-provisioning-template \
  --template-name widget-template \
  --enabled \
  --description "An updated provisioning template for widgets" \
  --provisioning-role-arn arn:aws:iam::504350838278:role/Provision_role
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网核心开发者指南](#)》中的 [Io AWS T 安全隧道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateProvisioningTemplate](#)中的。

## update-role-alias

以下代码示例演示了如何使用 update-role-alias。

### AWS CLI

#### 更新角色别名

以下update-role-alias示例更新了LightBulbRole角色别名。

```
aws iot update-role-alias \  
  --role-alias LightBulbRole \  
  --role-arn arn:aws:iam::123456789012:role/lightbulbrole-001
```

输出：

```
{  
  "roleAlias": "LightBulbRole",  
  "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/LightBulbRole"  
}
```

有关更多信息，请参阅 AWS IoT API 参考[UpdateRoleAlias](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRoleAlias](#)中的。

## update-scheduled-audit

以下代码示例演示了如何使用 update-scheduled-audit。

### AWS CLI

#### 更新计划审计定义

以下update-scheduled-audit示例更改了 AWS IoT Device Defender 计划审计的目标检查名称。

```
aws iot update-scheduled-audit \  
  --scheduled-audit-name WednesdayCertCheck \  
  --target-check-names CA_CERTIFICATE_EXPIRING_CHECK  
  DEVICE_CERTIFICATE_EXPIRING_CHECK REVOKED_CA_CERTIFICATE_STILL_ACTIVE_CHECK
```

输出：

```
{  
  "scheduledAuditArn": "arn:aws:iot:us-west-2:123456789012:scheduledaudit/  
  WednesdayCertCheck"  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[审计命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateScheduledAudit](#)中的。

## update-security-profile

以下代码示例演示了如何使用 update-security-profile。

### AWS CLI

#### 更改安全配置文件

以下update-security-profile示例更新了 AWS IoT Device Defender 安全配置文件的描述和行为。

```
aws iot update-security-profile \  
  --security-profile-name PossibleIssue \  
  --security-profile-description "Check to see if authorization fails 12 times in  
  5 minutes or if cellular bandwidth exceeds 128" \  
  --behaviors "[{"name":"CellularBandwidth","metric":"aws:message-byte-size  
\  
","criteria":{"comparisonOperator":"greater-than","value":{"count":128},  
\  
"consecutiveDatapointsToAlarm":1,"consecutiveDatapointsToClear":1}},{"name  
\  
":"Authorization","metric":"aws:num-authorization-failures","criteria":  
\  
{"comparisonOperator":"less-than","value":{"count":12},"durationSeconds  
\  
":300,"consecutiveDatapointsToAlarm":1,"consecutiveDatapointsToClear":1}]]"
```

输出：

```
{  
  "securityProfileName": "PossibleIssue",  
  "securityProfileArn": "arn:aws:iot:us-west-2:123456789012:securityprofile/  
PossibleIssue",  
  "securityProfileDescription": "check to see if authorization fails 12 times in 5  
minutes or if cellular bandwidth exceeds 128",  
  "behaviors": [  
    {  
      "name": "CellularBandwidth",  
      "metric": "aws:message-byte-size",  
      "criteria": {  
        "comparisonOperator": "greater-than",  
        "value": {  
          "count": 128  
        }  
      },  
      "consecutiveDatapointsToAlarm": 1,  
    }  
  ]  
}
```

```
        "consecutiveDatapointsToClear": 1
      }
    },
    {
      "name": "Authorization",
      "metric": "aws:num-authorization-failures",
      "criteria": {
        "comparisonOperator": "less-than",
        "value": {
          "count": 12
        },
        "durationSeconds": 300,
        "consecutiveDatapointsToAlarm": 1,
        "consecutiveDatapointsToClear": 1
      }
    }
  ],
  "version": 2,
  "creationDate": 1560278102.528,
  "lastModifiedDate": 1560352711.207
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateSecurityProfile](#)中的。

## update-stream

以下代码示例演示了如何使用 update-stream。

### AWS CLI

#### 更新直播

以下update-stream示例更新现有直播。直播版本以一为增量。

```
aws iot update-stream \
  --cli-input-json file://update-stream.json
```

update-stream.json 的内容：

```
{
```

```

"streamId": "stream12345",
"description": "This stream is used for Amazon FreeRTOS OTA Update 12345.",
"files": [
  {
    "fileId": 123,
    "s3Location": {
      "bucket": "codesign-ota-bucket",
      "key": "48c67f3c-63bb-4f92-a98a-4ee0fbc2bef6"
    }
  }
]
"roleArn": "arn:aws:iam:us-west-2:123456789012:role/service-role/my_ota_stream_role"
}

```

输出：

```

{
  "streamId": "stream12345",
  "streamArn": "arn:aws:iot:us-west-2:123456789012:stream/stream12345",
  "description": "This stream is used for Amazon FreeRTOS OTA Update 12345.",
  "streamVersion": 2
}

```

有关更多信息，请参阅 AWS IoT API 参考[UpdateStream](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateStream](#)中的。

## update-thing-group

以下代码示例演示了如何使用 update-thing-group。

### AWS CLI

#### 更新事物组的定义

以下 update-thing-group 示例更新了指定事物组的定义，更改了描述和两个属性。

```

aws iot update-thing-group \
  --thing-group-name HalogenBulbs \
  --thing-group-properties "thingGroupDescription=\"Halogen bulb group\",
attributePayload={attributes={Manufacturer=AnyCompany,wattage=60}}"

```



输出：

```
{
  "version": 2
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的事物组。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateThingGroup](#)中的。

## update-thing-groups-for-thing

以下代码示例演示了如何使用 `update-thing-groups-for-thing`。

AWS CLI

更改事物所属的群组

以下`update-thing-groups-for-thing`示例MyLightBulb从名为的组中移除名为DeadBulbs的事物，并将其同时添加到名为replaceableItems的组中。

```
aws iot update-thing-groups-for-thing \
  --thing-name MyLightBulb \
  --thing-groups-to-add "replaceableItems" \
  --thing-groups-to-remove "DeadBulbs"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的事物组。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateThingGroupsForThing](#)中的。

## update-thing

以下代码示例演示了如何使用 `update-thing`。

AWS CLI

将事物与事物类型关联

以下`update-thing`示例将 AWS IoT 注册表中的事物与事物类型相关联。建立关联时，您需要为事物类型定义的属性提供值。

```
aws iot update-thing \  
  --thing-name "MyOtherLightBulb" \  
  --thing-type-name "LightBulb" \  
  --attribute-payload '{"attributes": {"wattage": "75", "model": "123"}}'
```

此命令不产生输出。使用describe-thing命令查看结果。

有关更多信息，请参阅《AWS 物联网开发人员指南》中的事物[类型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateThing](#)中的。

## update-topic-rule-destination

以下代码示例演示了如何使用 update-topic-rule-destination。

### AWS CLI

#### 示例 1：启用主题规则目标

以下update-topic-rule-destination示例启用了到主题规则目标的流量。

```
aws iot update-topic-rule-destination \  
  --arn "arn:aws:iot:us-west-2:123456789012:ruledestination/http/  
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE" \  
  --status ENABLED
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[启用主题规则目标](#)。

#### 示例 2：禁用主题规则目标

以下update-topic-rule-destination示例禁用了到主题规则目标的流量。

```
aws iot update-topic-rule-destination \  
  --arn "arn:aws:iot:us-west-2:123456789012:ruledestination/http/  
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE" \  
  --status DISABLED
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[禁用主题规则目标](#)。

### 示例 3：发送新的确认消息

以下update-topic-rule-destination示例为主题规则目标发送了一条新的确认消息。

```
aws iot update-topic-rule-destination \  
  --arn "arn:aws:iot:us-west-2:123456789012:ruledestination/http/  
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE" \  
  --status IN_PROGRESS
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网开发者指南》中的[发送新的确认消息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateTopicRuleDestination](#)中的。

## validate-security-profile-behaviors

以下代码示例演示了如何使用 validate-security-profile-behaviors。

### AWS CLI

#### 示例 1：验证安全配置文件的行为参数

以下validate-security-profile-behaviors示例验证了 AWS IoT Device Defender 安全配置文件的一组格式正确且正确的行为。

```
aws iot validate-security-profile-behaviors \  
  --behaviors "[{"name":"CellularBandwidth","metric":"aws:message-byte-size",  
"criteria":{"comparisonOperator":"greater-than","value":{"count":128},  
"consecutiveDatapointsToAlarm":1,"consecutiveDatapointsToClear":1}},{"name"  
":"Authorization","metric":"aws:num-authorization-failures","criteria":  
{"comparisonOperator":"greater-than","value":{"count":12},"durationSeconds"  
":"300","consecutiveDatapointsToAlarm":1,"consecutiveDatapointsToClear":1}]"
```

输出：

```
{  
  "valid": true,  
  "validationErrors": []  
}
```

#### 示例 2：验证安全配置文件的不正确行为参数

以下 `validate-security-profile-behaviors` 示例验证了一组包含 AWS IoT Device Defender 安全配置文件错误的行为。

```
aws iot validate-security-profile-behaviors \
  --behaviors "[{\"name\":\"CellularBandwidth\",\"metric\":\"aws:message-byte-size\",
  \"criteria\":{\"comparisonOperator\":\"greater-than\",\"value\":{\"count\":128},
  \"consecutiveDatapointsToAlarm\":1,\"consecutiveDatapointsToClear\":1}}, {\"name
  \":\"Authorization\",\"metric\":\"aws:num-authorization-failures\",\"criteria\":
  {\"comparisonOperator\":\"greater-than\",\"value\":{\"count\":12},\"durationSeconds
  \":300,\"consecutiveDatapointsToAlarm\":100000,\"consecutiveDatapointsToClear
  \":1}}]"
```

输出：

```
{
  "valid": false,
  "validationErrors": [
    {
      "errorMessage": "Behavior Authorization is malformed.
      consecutiveDatapointsToAlarm 100000 should be in range[1,10]"
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[检测命令](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ValidateSecurityProfileBehaviors](#) 中的。

## AWS IoT 1-Click 使用的设备示例 AWS CLI

以下代码示例向您展示了如何使用 `with Devices` 来执行操作和实现常见场 AWS IoT 1-Click 景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### **claim-devices-by-claim-code**

以下代码示例演示了如何使用 `claim-devices-by-claim-code`。

#### AWS CLI

使用优惠码申领一台或多 AWS 台 IoT 1-Click 设备

以下 `claim-devices-by-claim-code` 示例使用优惠码 ( 而不是设备 ID ) 声明指定的 AWS IoT 1-Click 设备。

```
aws iot1click-devices claim-devices-by-claim-code \  
  --claim-code C-123EXAMPLE
```

输出：

```
{  
  "Total": 9  
  "ClaimCode": "C-123EXAMPLE"  
}
```

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ClaimDevicesByClaimCode](#) 中的。

### **describe-device**

以下代码示例演示了如何使用 `describe-device`。

#### AWS CLI

描述设备

以下 `describe-device` 示例描述了指定的设备。

```
aws iot1click-devices describe-device \  
  --device-id
```

```
--device-id G030PM0123456789
```

输出：

```
{
  "DeviceDescription": {
    "Arn": "arn:aws:iot1click:us-west-2:012345678901:devices/G030PM0123456789",
    "Attributes": {
      "projectRegion": "us-west-2",
      "projectName": "AnytownDumpsters",
      "placementName": "customer217",
      "deviceTemplateName": "empty-dumpster-request"
    },
    "DeviceId": "G030PM0123456789",
    "Enabled": false,
    "RemainingLife": 99.9,
    "Type": "button",
    "Tags": {}
  }
}
```

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDevice](#) 中的。

## finalize-device-claim

以下代码示例演示了如何使用 `finalize-device-claim`。

### AWS CLI

使用设备 ID 完成 AWS IoT 1-Click 设备的索赔申请

以下 `finalize-device-claim` 示例使用设备 ID（而不是折扣码）最终确定了针对指定 AWS IoT 1-Click 设备的索赔请求。

```
aws iot1click-devices finalize-device-claim \
  --device-id G030PM0123456789
```

输出：

```
{
  "State": "CLAIMED"
}
```

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 设备与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [FinalizeDeviceClaim](#) 中的。

## get-device-methods

以下代码示例演示了如何使用 get-device-methods。

### AWS CLI

列出设备的可用方法

以下 get-device-methods 示例列出了设备的可用方法。

```
aws iot1click-devices get-device-methods \
  --device-id G030PM0123456789
```

输出：

```
{
  "DeviceMethods": [
    {
      "MethodName": "getDeviceHealthParameters"
    },
    {
      "MethodName": "setDeviceHealthMonitorCallback"
    },
    {
      "MethodName": "getDeviceHealthMonitorCallback"
    },
    {
      "MethodName": "setOnClickCallback"
    },
    {
      "MethodName": "getOnClickCallback"
    }
  ]
}
```

```
}
```

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS k 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDeviceMethods](#) 中的。

## **initiate-device-claim**

以下代码示例演示了如何使用 `initiate-device-claim`。

### AWS CLI

使用设备 ID 为 AWS IoT 1-Click 设备发起索赔请求

以下 `initiate-device-claim` 示例使用设备 ID (而不是折扣码) 为指定的 IoT AWS T 1-Click 设备发起索赔请求。

```
aws iot1click-devices initiate-device-claim \  
  --device-id G030PM0123456789
```

输出：

```
{  
  "State": "CLAIM_INITIATED"  
}
```

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS k 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [InitiateDeviceClaim](#) 中的。

## **invoke-device-method**

以下代码示例演示了如何使用 `invoke-device-method`。

### AWS CLI

在设备上调用设备方法

以下 `invoke-device-method` 示例在设备上调用指定的方法。



```
aws iot1click-devices invoke-device-method \  
  --cli-input-json file://invoke-device-method.json
```

invoke-device-method.json 的内容：

```
{  
  "DeviceId": "G030PM0123456789",  
  "DeviceMethod": {  
    "DeviceType": "device",  
    "MethodName": "getDeviceHealthParameters"  
  }  
}
```

输出：

```
{  
  "DeviceMethodResponse": "{\"remainingLife\": 99.8}"  
}
```

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [InvokeDeviceMethod](#) 中的。

## list-device-events

以下代码示例演示了如何使用 list-device-events。

AWS CLI

列出指定时间范围内的设备事件

以下 list-device-events 示例列出了指定设备在指定时间范围内的事件。

```
aws iot1click-devices list-device-events \  
  --device-id G030PM0123456789 \  
  --from-time-stamp 2019-07-17T15:45:12.880Z --to-time-stamp  
  2019-07-19T15:45:12.880Z
```

输出：

```
{
  "Events": [
    {
      "Device": {
        "Attributes": {},
        "DeviceId": "G030PM0123456789",
        "Type": "button"
      },
      "StdEvent": "{\\"clickType\\": \\"SINGLE\\",
\\"reportedTime\\": \\"2019-07-18T23:47:55.015Z\\", \\"certificateId\\":
\\"fe8798a6c97c62ef8756b80eeefdcd2280f3352f82faa8080c74cc4f4a4d1811\\",
\\"remainingLife\\": 99.85000000000001, \\"testMode\\": false}"
    },
    {
      "Device": {
        "Attributes": {},
        "DeviceId": "G030PM0123456789",
        "Type": "button"
      },
      "StdEvent": "{\\"clickType\\": \\"DOUBLE\\",
\\"reportedTime\\": \\"2019-07-19T00:14:41.353Z\\", \\"certificateId\\":
\\"fe8798a6c97c62ef8756b80eeefdcd2280f3352f82faa8080c74cc4f4a4d1811\\",
\\"remainingLife\\": 99.8, \\"testMode\\": false}"
    }
  ]
}
```

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDeviceEvents](#) 中的。

## list-devices

以下代码示例演示了如何使用 list-devices。

### AWS CLI

列出指定类型的设备

以下 list-devices 示例列出了指定类型的设备。

```
aws iot1click-devices list-devices \
```

```
--device-type button
```

此命令不生成任何输出。

输出：

```
{
  "Devices": [
    {
      "remainingLife": 99.9,
      "attributes": {
        "arn": "arn:aws:iot1click:us-west-2:123456789012:devices/
G030PM0123456789",
        "type": "button",
        "deviceId": "G030PM0123456789",
        "enabled": false
      }
    }
  ]
}
```

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDevices](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

AWS CLI

列出设备的标签

以下 `list-tags-for-resource` 示例列出了指定设备的标签。

```
aws iot1click-devices list-tags-for-resource \
  --resource-arn "arn:aws:iot1click:us-west-2:012345678901:devices/
G030PM0123456789"
```

输出：

```
{
  "Tags": {
    "Driver Phone": "123-555-0199",
    "Driver": "Jorge Souza"
  }
}
```

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

向设备 AWS 资源添加标签

以下 tag-resource 示例向指定资源添加了两个标签。

```
aws iot1click-devices tag-resource \
  --cli-input-json file://devices-tag-resource.json
```

devices-tag-resource.json 的内容：

```
{
  "ResourceArn": "arn:aws:iot1click:us-west-2:123456789012:devices/
G030PM0123456789",
  "Tags": {
    "Driver": "Jorge Souza",
    "Driver Phone": "123-555-0199"
  }
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## unclaim-device

以下代码示例演示了如何使用 unclaim-device。

### AWS CLI

从您的账户中取消认领（注销）设备 AWS

以下unclaim-device示例从您的 AWS 账户中取消对指定设备的声明（注销）。

```
aws iot1click-devices unclaim-device \  
  --device-id G030PM0123456789
```

输出：

```
{  
  "State": "UNCLAIMED"  
}
```

有关更多信息，请参阅《[AWS IoT 1-Click 开发者指南](#)》中的[将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UnclaimDevice](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从设备 AWS 资源中移除标签

以下untag-resource示例Driver从指定的设备资源中删除名称为Driver Phone和的标签。

```
aws iot1click-devices untag-resource \  
  --resource-arn "arn:aws:iot1click:us-west-2:123456789012:projects/  
AnytownDumpsters" \  
  --tag-keys "Driver Phone" "Driver"
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-device-state

以下代码示例演示了如何使用 `update-device-state`。

### AWS CLI

更新设备的“已启用”状态

以下内容 `update-device-state` 将指定设备的状态设置为 `enabled`。

```
aws iot1click-devices update-device-state \  
  --device-id G030PM0123456789 \  
  --enabled
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateDeviceState](#) 中的。

## AWS IoT 1-Click 使用项目示例 AWS CLI

以下代码示例向您展示了如何使用 `with Pro AWS IoT 1-Click projects` 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 [GitHub](#)，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### **associate-device-with-placement**

以下代码示例演示了如何使用 `associate-device-with-placement`。

#### AWS CLI

将 I AWS oT 1-Click 设备与现有展示位置关联

以下 `associate-device-with-placement` 示例将指定的 AWS IoT 1-Click 设备与现有展示位置相关联。

```
aws iot1click-projects associate-device-with-placement \  
  --project-name AnytownDumpsters \  
  --placement-name customer217 \  
  --device-template-name empty-dumpster-request \  
  --device-id G030PM0123456789
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 设备与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateDeviceWithPlacement](#) 中的。

### **create-placement**

以下代码示例演示了如何使用 `create-placement`。

#### AWS CLI

为项目创建 I AWS oT 1-Click 展示位置

以下 `create-placement` 示例为指定项目创建 AWS IoT 1-Click 展示位置。

```
aws iot1click-projects create-placement \  
  --project-name AnytownDumpsters \  
  --placement-name customer217
```

```
--project-name AnytownDumpsters \  
--placement-name customer217 \  
--attributes '{"location": "123 Any Street Anytown, USA 10001", "phone":  
"123-456-7890"}'
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS IoT 1-Click 开发者指南](#)》中的[将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreatePlacement](#) 中的。

## create-project

以下代码示例演示了如何使用 create-project。

### AWS CLI

为单个或多个展示位置创建 AWS IoT 1-Click 项目

以下 create-project 示例为展示位置创建一个 AWS IoT 1-Click 项目。

aws iot1click-projects 创建项目 — file: //create-project.json cli-input-json

create-project.json 的内容：

```
{  
  "projectName": "AnytownDumpsters",  
  "description": "All dumpsters in the Anytown region.",  
  "placementTemplate": {  
    "defaultAttributes": {  
      "City" : "Anytown"  
    },  
    "deviceTemplates": {  
      "empty-dumpster-request" : {  
        "deviceType": "button"  
      }  
    }  
  }  
}
```

此命令不生成任何输出。



有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》中的将 IoT 1-Click AWS CLI 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateProject](#) 中的。

## delete-placement

以下代码示例演示了如何使用 delete-placement。

### AWS CLI

从项目中删除展示位置

以下 delete-placement 示例从项目中删除指定的展示位置。

```
aws iot1click-projects delete-placement \  
  --project-name AnytownDumpsters \  
  --placement-name customer217
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》中的将 IoT 1-Click AWS CLI 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeletePlacement](#) 中的。

## delete-project

以下代码示例演示了如何使用 delete-project。

### AWS CLI

从您的 AWS 账户中删除项目

以下 delete-project 示例从您的 AWS 账户中删除指定的项目。

```
aws iot1click-projects delete-project \  
  --project-name AnytownDumpsters
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的将 IoT 1-Click AWS CLI 与 AWS CLI 配合使用。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteProject](#) 中的。

## describe-placement

以下代码示例演示了如何使用 describe-placement。

### AWS CLI

描述项目的展示位置

以下 describe-placement 示例描述了指定项目的展示位置。

```
aws iot1click-projects describe-placement \
  --project-name AnytownDumpsters \
  --placement-name customer217
```

输出：

```
{
  "placement": {
    "projectName": "AnytownDumpsters",
    "placementName": "customer217",
    "attributes": {
      "phone": "123-555-0110",
      "location": "123 Any Street Anytown, USA 10001"
    },
    "createdDate": 1563488454,
    "updatedAt": 1563488454
  }
}
```

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的将 IoT 1-Click AWS CLI 与 AWS CLI 配合使用。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribePlacement](#) 中的。

## describe-project

以下代码示例演示了如何使用 describe-project。

## AWS CLI

### 描述 | AWS IoT 1-Click 项目

以下describe-project示例描述了指定的 AWS IoT 1-Click 项目。

```
aws iot1click-projects describe-project \  
  --project-name AnytownDumpsters
```

输出：

```
{  
  "project": {  
    "arn": "arn:aws:iot1click:us-west-2:012345678901:projects/AnytownDumpsters",  
    "projectName": "AnytownDumpsters",  
    "description": "All dumpsters in the Anytown region.",  
    "createdDate": 1563483100,  
    "updatedDate": 1563483100,  
    "placementTemplate": {  
      "defaultAttributes": {  
        "City": "Anytown"  
      },  
      "deviceTemplates": {  
        "empty-dumpster-request": {  
          "deviceType": "button",  
          "callbackOverrides": {}  
        }  
      }  
    },  
    "tags": {}  
  }  
}
```

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeProject](#) 中的。

### disassociate-device-from-placement

以下代码示例演示了如何使用 disassociate-device-from-placement。

## AWS CLI

### 取消设备与放置位置的关联

以下disassociate-device-from-placement示例取消指定设备与放置位置的关联。

```
aws iot1click-projects disassociate-device-from-placement \  
  --project-name AnytownDumpsters \  
  --placement-name customer217 \  
  --device-template-name empty-dumpster-request
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS IoT 1-Click 开发者指南](#)》中的[将 IoT 1-Click AWS CLI 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateDeviceFromPlacement](#)中的。

## get-devices-in-placement

以下代码示例演示了如何使用 get-devices-in-placement。

### AWS CLI

列出项目中包含的某个位置中的所有设备

以下get-devices-in-placement示例列出了指定项目中包含的位于指定位置的所有设备。

```
aws iot1click-projects get-devices-in-placement \  
  --project-name AnytownDumpsters \  
  --placement-name customer217
```

输出：

```
{  
  "devices": {  
    "empty-dumpster-request": "G030PM0123456789"  
  }  
}
```

有关更多信息，请参阅《[AWS IoT 1-Click 开发者指南](#)》中的[将 IoT 1-Click AWS CLI 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDevicesInPlacement](#)中的。

## list-placements

以下代码示例演示了如何使用 list-placements。

### AWS CLI

列出项目的所有 AWS 物联网 1-Click 展示位置

以下list-placements示例列出了指定项目的所有 AWS IoT 1-Click 展示位置。

```
aws iot1click-projects list-placements \  
  --project-name AnytownDumpsters
```

输出：

```
{  
  "placements": [  
    {  
      "projectName": "AnytownDumpsters",  
      "placementName": "customer217",  
      "createdDate": 1563488454,  
      "updatedAt": 1563488454  
    }  
  ]  
}
```

有关更多信息，请参阅《[AWS IoT 1-Click 开发者指南](#)》中的[将 IoT 1-Click AWS k 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPlacements](#)中的。

## list-projects

以下代码示例演示了如何使用 list-projects。

### AWS CLI

列出所有 AWS IoT 1-Click 项目

以下list-projects示例列出了您账户中的所有 AWS IoT 1-Click 项目。

```
aws iot1click-projects list-projects
```

输出：

```
{
  "projects": [
    {
      "arn": "arn:aws:iot1click:us-west-2:012345678901:projects/AnytownDumpsters",
      "projectName": "AnytownDumpsters",
      "createdDate": 1563483100,
      "updatedDate": 1563483100,
      "tags": {}
    }
  ]
}
```

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListProjects](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

AWS CLI

列出项目资源的标签

以下 `list-tags-for-resource` 示例列出了指定项目资源的标签。

```
aws iot1click-projects list-tags-for-resource \
  --resource-arn "arn:aws:iot1click:us-west-2:123456789012:projects/AnytownDumpsters"
```

输出：

```
{
  "tags": {
    "Manager": "Li Juan",
    "Account": "45215"
  }
}
```

```
}  
}
```

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为项目资源添加标签

以下 tag-resource 示例向指定的项目资源添加了两个标签。

```
aws iot1click-projects tag-resource \  
  --cli-input-json file://devices-tag-resource.json
```

devices-tag-resource.json 的内容：

```
{  
  "resourceArn": "arn:aws:iot1click:us-west-2:123456789012:projects/  
AnytownDumpsters",  
  "tags": {  
    "Account": "45215",  
    "Manager": "Li Juan"  
  }  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

## AWS CLI

从项目资源中移除标签

以下 `untag-resource` 示例 `Manager` 从指定项目中删除带有密钥名称的标签。

```
aws iot1click-projects untag-resource \  
  --resource-arn "arn:aws:iot1click:us-west-2:123456789012:projects/  
AnytownDumpsters" \  
  --tag-keys "Manager"
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-placement

以下代码示例演示了如何使用 `update-placement`。

### AWS CLI

更新展示位置的“属性”键值对

以下 `update-placement` 示例更新展示位置的“属性”键值对。

```
aws iot1click-projects update-placement \  
  --cli-input-json file://update-placement.json
```

`update-placement.json` 的内容：

```
{  
  "projectName": "AnytownDumpsters",  
  "placementName": "customer217",  
  "attributes": {  
    "phone": "123-456-7890",  
    "location": "123 Any Street Anytown, USA 10001"  
  }  
}
```



此命令不生成任何输出。

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS CLI 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdatePlacement](#) 中的。

## update-project

以下代码示例演示了如何使用 update-project。

### AWS CLI

#### 更新项目的设置

以下 update-project 示例更新了项目的描述。

```
aws iot1click-projects update-project \  
  --project-name AnytownDumpsters \  
  --description "All dumpsters (yard waste, recycling, garbage) in the Anytown  
  region."
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS IoT 1-Click 开发者指南》](#) 中的 [将 IoT 1-Click AWS CLI 与 AWS CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateProject](#) 中的。

## AWS IoT Analytics 使用示例 AWS CLI

以下代码示例向您展示了如何使用 with 来执行操作和实现常见场景 AWS IoT Analytics。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### batch-put-message

以下代码示例演示了如何使用 batch-put-message。

#### AWS CLI

向频道发送消息

以下batch-put-message示例向指定频道发送消息。

```
aws iotanalytics batch-put-message \  
  --cli-binary-format raw-in-base64-out \  
  --cli-input-json file://batch-put-message.json
```

batch-put-message.json 的内容：

```
{  
  "channelName": "mychannel",  
  "messages": [  
    {  
      "messageId": "0001",  
      "payload": "eyJhdGVtcGVyYXR1cmUiOiAyMCB9"  
    }  
  ]  
}
```

输出：

```
{  
  "batchPutMessageErrorEntries": []  
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[BatchPutMessage](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchPutMessage](#)中的。

## cancel-pipeline-reprocessing

以下代码示例演示了如何使用 `cancel-pipeline-reprocessing`。

### AWS CLI

取消通过管道对数据的重新处理

以下 `cancel-pipeline-reprocessing` 示例取消了通过指定管道对数据的重新处理。

```
aws iotanalytics cancel-pipeline-reprocessing \  
  --pipeline-name mypipeline \  
  --reprocessing-id "6ad2764f-fb13-4de3-b101-4e74af03b043"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Analytics API 参考》[CancelPipelineReprocessing](#) 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CancelPipelineReprocessing](#) 中的。

## create-channel

以下代码示例演示了如何使用 `create-channel`。

### AWS CLI

创建频道

以下 `create-channel` 示例创建了一个具有指定配置的频道。通道收集来自 MQTT 主题的数据，并在将数据发布到管道之前将原始未处理消息归档。

```
aws iotanalytics create-channel \  
  --cli-input-json file://create-channel.json
```

`create-channel.json` 的内容：

```
{  
  "channelName": "mychannel",  
  "retentionPeriod": {  
    "unlimited": true  
  },  
}
```

```
    "tags": [
      {
        "key": "Environment",
        "value": "Production"
      }
    ]
  }
```

输出：

```
{
  "channelArn": "arn:aws:iotanalytics:us-west-2:123456789012:channel/mychannel",
  "channelName": "mychannel",
  "retentionPeriod": {
    "unlimited": true
  }
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[CreateChannel](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateChannel](#)中的。

## create-dataset-content

以下代码示例演示了如何使用 create-dataset-content。

### AWS CLI

创建数据集的内容

以下create-dataset-content示例通过应用（SQL 查询）或 queryActioncontainerAction（执行容器化应用程序）来创建指定数据集的内容。

```
aws iotanalytics create-dataset-content \
  --dataset-name mydataset
```

输出：

```
{
  "versionId": "d494b416-9850-4670-b885-ca22f1e89d62"
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[CreateDatasetContent](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDatasetContent](#)中的。

## create-dataset

以下代码示例演示了如何使用 create-dataset。

### AWS CLI

#### 创建数据集

以下create-dataset示例创建了一个数据集。数据集存储通过应用 queryAction ( SQL 查询 ) 或 containerAction ( 执行容器化应用程序 ) 从数据存储中检索的数据。此操作创建数据集的骨架。您可以通过调用手动填充数据集，CreateDatasetContent也可以根据trigger您指定的自动填充数据集。

```
aws iotanalytics create-dataset \  
  --cli-input-json file://create-dataset.json
```

create-dataset.json 的内容：

```
{  
  "datasetName": "mydataset",  
  "actions": [  
    {  
      "actionName": "myDatasetAction",  
      "queryAction": {  
        "sqlQuery": "SELECT * FROM mydatastore"  
      }  
    }  
  ],  
  "retentionPeriod": {  
    "unlimited": true  
  },  
  "tags": [  
    {  
      "key": "Environment",  
      "value": "Production"  
    }  
  ]  
}
```

输出：

```
{
  "datasetName": "mydataset",
  "retentionPeriod": {
    "unlimited": true
  },
  "datasetArn": "arn:aws:iotanalytics:us-west-2:123456789012:dataset/mydataset"
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[CreateDataset](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDataset](#)中的。

## create-datastore

以下代码示例演示了如何使用 create-datastore。

### AWS CLI

#### 创建数据存储

以下create-datastore示例创建了一个数据存储，它是一个消息存储库。

```
aws iotanalytics create-datastore \
  --cli-input-json file://create-datastore.json
```

create-datastore.json 的内容：

```
{
  "datastoreName": "mydatastore",
  "retentionPeriod": {
    "numberOfDays": 90
  },
  "tags": [
    {
      "key": "Environment",
      "value": "Production"
    }
  ]
}
```

输出：

```
{
  "datastoreName": "mydatastore",
  "datastoreArn": "arn:aws:iotanalytics:us-west-2:123456789012:datastore/
mydatastore",
  "retentionPeriod": {
    "numberOfDays": 90,
    "unlimited": false
  }
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[CreateDatastore](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDatastore](#)中的。

## create-pipeline

以下代码示例演示了如何使用 create-pipeline。

### AWS CLI

#### 创建 IoT Analytics 管道

以下create-pipeline示例创建了一个管道。管道使用来自通道的消息，并允许您在将消息存储在数据存储之前处理消息。您必须同时指定通道和数据存储活动，并可选择在pipelineActivities数组中指定多达 23 个其他活动。

```
aws iotanalytics create-pipeline \
  --cli-input-json file://create-pipeline.json
```

create-pipeline.json 的内容：

```
{
  "pipelineName": "mypipeline",
  "pipelineActivities": [
    {
      "channel": {
        "name": "myChannelActivity",
        "channelName": "mychannel",
        "next": "myMathActivity"
      }
    }
  ]
}
```

```
    },
    {
      "datastore": {
        "name": "myDatastoreActivity",
        "datastoreName": "mydatastore"
      }
    },
    {
      "math": {
        "name": "myMathActivity",
        "math": "((temp - 32) * 5.0) / 9.0",
        "attribute": "tempC",
        "next": "myDatastoreActivity"
      }
    }
  ],
  "tags": [
    {
      "key": "Environment",
      "value": "Beta"
    }
  ]
}
```

输出：

```
{
  "pipelineArn": "arn:aws:iotanalytics:us-west-2:123456789012:pipeline/
mypipeline",
  "pipelineName": "mypipeline"
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[CreatePipeline](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePipeline](#)中的。

## delete-channel

以下代码示例演示了如何使用 delete-channel。

### AWS CLI

#### 删除 IoT Analytics 频道



以下delete-channel示例删除了指定的频道。

```
aws iotanalytics delete-channel \  
  --channel-name mychannel
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Analytics API 参考》[DeleteChannel](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteChannel](#)中的。

## delete-dataset-content

以下代码示例演示了如何使用 delete-dataset-content。

### AWS CLI

删除数据集内容

以下delete-dataset-content示例删除了指定数据集的内容。

```
aws iotanalytics delete-dataset-content \  
  --dataset-name mydataset
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Analytics API 参考》[DeleteDatasetContent](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDatasetContent](#)中的。

## delete-dataset

以下代码示例演示了如何使用 delete-dataset。

### AWS CLI

删除数据集

以下delete-dataset示例删除了指定的数据集。在执行此操作之前，您不必删除数据集的内容。

```
aws iotanalytics delete-dataset \  
  --dataset-name mydataset
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Analytics API 参考》[DeleteDataset](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDataset](#)中的。

## delete-datastore

以下代码示例演示了如何使用 delete-datastore。

### AWS CLI

删除数据存储

以下delete-datastore示例删除了指定的数据存储。

```
aws iotanalytics delete-datastore \  
  --datastore-name mydatastore
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Analytics API 参考》[DeleteDatastore](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDatastore](#)中的。

## delete-pipeline

以下代码示例演示了如何使用 delete-pipeline。

### AWS CLI

删除管道

以下delete-pipeline示例删除了指定的管道。

```
aws iotanalytics delete-pipeline \  
  --pipeline-name mypipeline
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Analytics API 参考》[DeletePipeline](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePipeline](#)中的。

## describe-channel

以下代码示例演示了如何使用 describe-channel。

### AWS CLI

检索有关频道的信息

以下describe-channel示例显示了指定频道的详细信息，包括统计信息。

```
aws iotanalytics describe-channel \  
  --channel-name mychannel \  
  --include-statistics
```

输出：

```
{  
  "statistics": {  
    "size": {  
      "estimatedSizeInBytes": 402.0,  
      "estimatedOn": 1561504380.0  
    }  
  },  
  "channel": {  
    "status": "ACTIVE",  
    "name": "mychannel",  
    "lastUpdateTime": 1557860351.001,  
    "creationTime": 1557860351.001,  
    "retentionPeriod": {  
      "unlimited": true  
    },  
    "arn": "arn:aws:iotanalytics:us-west-2:123456789012:channel/mychannel"  
  }  
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[DescribeChannel](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeChannel](#)中的。

## describe-dataset

以下代码示例演示了如何使用 describe-dataset。

## AWS CLI

### 检索有关数据集的信息

以下describe-dataset示例显示了指定数据集的详细信息。

```
aws iotanalytics describe-dataset \  
  --dataset-name mydataset
```

输出：

```
{  
  "dataset": {  
    "status": "ACTIVE",  
    "contentDeliveryRules": [],  
    "name": "mydataset",  
    "lastUpdateTime": 1557859240.658,  
    "triggers": [],  
    "creationTime": 1557859240.658,  
    "actions": [  
      {  
        "actionName": "query_32",  
        "queryAction": {  
          "sqlQuery": "SELECT * FROM mydatastore",  
          "filters": []  
        }  
      }  
    ],  
    "retentionPeriod": {  
      "numberOfDays": 90,  
      "unlimited": false  
    },  
    "arn": "arn:aws:iotanalytics:us-west-2:123456789012:dataset/mydataset"  
  }  
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[DescribeDataset](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDataset](#)中的。

## describe-datastore

以下代码示例演示了如何使用 describe-datastore。

## AWS CLI

检索有关数据存储的信息

以下describe-datastore示例显示了指定数据存储的详细信息，包括统计信息。

```
aws iotanalytics describe-datastore \  
  --datastore-name mydatastore \  
  --include-statistics
```

输出：

```
{  
  "datastore": {  
    "status": "ACTIVE",  
    "name": "mydatastore",  
    "lastUpdateTime": 1557858971.02,  
    "creationTime": 1557858971.02,  
    "retentionPeriod": {  
      "unlimited": true  
    },  
    "arn": "arn:aws:iotanalytics:us-west-2:123456789012:datastore/mydatastore"  
  },  
  "statistics": {  
    "size": {  
      "estimatedSizeInBytes": 397.0,  
      "estimatedOn": 1561592040.0  
    }  
  }  
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[DescribeDatastore](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDatastore](#)中的。

## describe-logging-options

以下代码示例演示了如何使用 describe-logging-options。

## AWS CLI

检索当前的日志记录选项

以下describe-logging-options示例显示了当前的 AWS IoT Analytics 日志选项。

```
aws iotanalytics describe-logging-options
```

此命令不生成任何输出。输出：

```
{
  "loggingOptions": {
    "roleArn": "arn:aws:iam::123456789012:role/service-role/myIoTAnalyticsRole",
    "enabled": true,
    "level": "ERROR"
  }
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[DescribeLoggingOptions](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoggingOptions](#)中的。

## describe-pipeline

以下代码示例演示了如何使用 describe-pipeline。

### AWS CLI

检索有关管道的信息

以下describe-pipeline示例显示了指定管道的详细信息。

```
aws iotanalytics describe-pipeline \
  --pipeline-name mypipeline
```

输出：

```
{
  "pipeline": {
    "activities": [
      {
        "channel": {
          "channelName": "mychannel",
          "name": "mychannel_28",
          "next": "mydatastore_29"
        }
      }
    ]
  }
}
```

```
    },
    {
      "datastore": {
        "datastoreName": "mydatastore",
        "name": "mydatastore_29"
      }
    }
  ],
  "name": "mypipeline",
  "lastUpdateTime": 1561676362.515,
  "creationTime": 1557859124.432,
  "reprocessingSummaries": [
    {
      "status": "SUCCEEDED",
      "creationTime": 1561676362.189,
      "id": "6ad2764f-fb13-4de3-b101-4e74af03b043"
    }
  ],
  "arn": "arn:aws:iotanalytics:us-west-2:123456789012:pipeline/mypipeline"
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[DescribePipeline](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePipeline](#)中的。

## get-dataset-content

以下代码示例演示了如何使用 get-dataset-content。

### AWS CLI

检索数据集的内容

以下get-dataset-content示例以预签名 URI 的形式检索数据集的内容。

```
aws iotanalytics get-dataset-content --dataset-name mydataset
```

输出：

```
{
  "status": {
    "state": "SUCCEEDED"
  }
}
```

```
  },
  "timestamp": 1557863215.995,
  "entries": [
    {
      "dataURI": "https://aws-radiant-
dataset-12345678-1234-1234-1234-123456789012.s3.us-west-2.amazonaws.com/
results/12345678-e8b3-46ba-b2dd-efe8d86cf385.csv?X-Amz-Security-Token=...-Amz-
Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20190628T173437Z&X-Amz-SignedHeaders=host&X-
Amz-Expires=7200&X-Amz-Credential=...F20190628%2Fus-west-2%2Fs3%2Faws4_request&X-
Amz-Signature=..."
    }
  ]
}
```

有关更多信息，请参阅指南[GetDatasetContent](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDatasetContent](#)中的。

## list-channels

以下代码示例演示了如何使用 `list-channels`。

### AWS CLI

#### 检索频道列表

以下`list-channels`示例显示可用频道的摘要信息。

```
aws iotanalytics list-channels
```

输出：

```
{
  "channelSummaries": [
    {
      "status": "ACTIVE",
      "channelName": "mychannel",
      "creationTime": 1557860351.001,
      "lastUpdateTime": 1557860351.001
    }
  ]
}
```



有关更多信息，请参阅《AWS IoT Analytics API 参考》[ListChannels](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListChannels](#)中的。

## list-dataset-contents

以下代码示例演示了如何使用 `list-dataset-contents`。

### AWS CLI

列出有关数据集内容的信息

以下`list-dataset-contents`示例列出了有关已创建的数据集内容的信息。

```
aws iotanalytics list-dataset-contents \  
  --dataset-name mydataset
```

输出：

```
{  
  "datasetContentSummaries": [  
    {  
      "status": {  
        "state": "SUCCEEDED"  
      },  
      "scheduleTime": 1557863215.995,  
      "version": "b10ea2a9-66c1-4d99-8d1f-518113b738d0",  
      "creationTime": 1557863215.995  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[ListDatasetContents](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDatasetContents](#)中的。

## list-datasets

以下代码示例演示了如何使用 `list-datasets`。

### AWS CLI

检索有关数据集的信息

以下`list-datasets`示例列出了有关可用数据集的摘要信息。

```
aws iotanalytics list-datasets
```

输出：

```
{
  "datasetSummaries": [
    {
      "status": "ACTIVE",
      "datasetName": "mydataset",
      "lastUpdateTime": 1557859240.658,
      "triggers": [],
      "creationTime": 1557859240.658,
      "actions": [
        {
          "actionName": "query_32",
          "actionType": "QUERY"
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[ListDatasets](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDatasets](#)中的。

## list-datastores

以下代码示例演示了如何使用 `list-datastores`。

### AWS CLI

#### 检索数据存储列表

以下`list-datastores`示例显示有关可用数据存储的摘要信息。

```
aws iotanalytics list-datastores
```

输出：

```
{
  "datastoreSummaries": [
    {
      "status": "ACTIVE",
      "datastoreName": "mydatastore",
      "creationTime": 1557858971.02,
      "lastUpdateTime": 1557858971.02
    }
  ]
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[ListDatastores](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDatastores](#)中的。

## list-pipelines

以下代码示例演示了如何使用 list-pipelines。

### AWS CLI

#### 检索管道列表

以下list-pipelines示例显示了可用管道的列表。

```
aws iotanalytics list-pipelines
```

输出：

```
{
  "pipelineSummaries": [
    {
      "pipelineName": "mypipeline",
      "creationTime": 1557859124.432,
      "lastUpdateTime": 1557859124.432,
      "reprocessingSummaries": []
    }
  ]
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[ListPipelines](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPipelines](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出资源的标签

以下`list-tags-for-resource`示例列出了您已附加到指定资源的标签。

```
aws iotanalytics list-tags-for-resource \  
  --resource-arn "arn:aws:iotanalytics:us-west-2:123456789012:channel/mychannel"
```

输出：

```
{  
  "tags": [  
    {  
      "value": "bar",  
      "key": "foo"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[ListTagsForResource](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## put-logging-options

以下代码示例演示了如何使用 `put-logging-options`。

### AWS CLI

设置或更新日志记录选项

以下`put-logging-options`示例设置或更新 AWS IoT Analytics 日志选项。如果您更新任何`loggingOptions`字段的值，则更改最多可能需要一分钟才能生效。此外，如果您更改附加到您

在“roLearn”字段中指定的角色的策略（例如，更正无效的策略），则该更改最多可能需要五分钟才能生效。

```
aws iotanalytics put-logging-options \  
  --cli-input-json file://put-logging-options.json
```

put-logging-options.json 的内容：

```
{  
  "loggingOptions": {  
    "roleArn": "arn:aws:iam::123456789012:role/service-role/myIoTAnalyticsRole",  
    "level": "ERROR",  
    "enabled": true  
  }  
}
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Analytics API 参考》[PutLoggingOptions](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutLoggingOptions](#)中的。

## run-pipeline-activity

以下代码示例演示了如何使用 run-pipeline-activity。

### AWS CLI

#### 模拟管道活动

以下run-pipeline-activity示例模拟了在消息负载上运行管道活动的结果。

```
aws iotanalytics run-pipeline-activity \  
  --pipeline-activity file://maths.json \  
  --payloads file://payloads.json
```

maths.json 的内容：

```
{  
  "math": {  
    "name": "MyMathActivity",
```

```
    "math": "((temp - 32) * 5.0) / 9.0",  
    "attribute": "tempC"  
  }  
}
```

payloads.json 的内容：

```
[  
  {"humidity": 52, "temp": 68 },  
  {"humidity": 52, "temp": 32 }  
]
```

输出：

```
{  
  "logResult": "",  
  "payloads": [  
    "eyJodW1pZG10eSI6NTIsInRlbXAiOjY4LCJ0ZW1wQyI6MjB9",  
    "eyJodW1pZG10eSI6NTIsInRlbXAiOjMyLCJ0ZW1wQyI6MH0=",  
  ]  
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[RunPipelineActivity](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RunPipelineActivity](#)中的。

## sample-channel-data

以下代码示例演示了如何使用 sample-channel-data。

### AWS CLI

从频道检索示例消息

以下sample-channel-data示例检索在指定时间段内从指定频道收录的消息示例。您最多可以检索 10 条消息。

```
aws iotanalytics sample-channel-data \  
  --channel-name mychannel
```

输出：

```
{
  "payloads": [
    "eyJhdGVtcGVyYXR1cmUiOiAyMCM9",
    "eyJhZm9vIjogImJhcnVzIj0="
  ]
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[SampleChannelData](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SampleChannelData](#)中的。

## start-pipeline-reprocessing

以下代码示例演示了如何使用 start-pipeline-reprocessing。

### AWS CLI

开始管道再处理

以下start-pipeline-reprocessing示例通过指定的管道开始重新处理原始消息数据。

```
aws iotanalytics start-pipeline-reprocessing \
  --pipeline-name mypipeline
```

输出：

```
{
  "reprocessingId": "6ad2764f-fb13-4de3-b101-4e74af03b043"
}
```

有关更多信息，请参阅《AWS IoT Analytics API 参考》[StartPipelineReprocessing](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartPipelineReprocessing](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源添加或修改标签

以下 `tag-resource` 示例添加或修改了附加到指定资源的标签。

```
aws iotanalytics tag-resource \  
  --resource-arn "arn:aws:iotanalytics:us-west-2:123456789012:channel/mychannel" \  
  --tags "[{\"key\": \"Environment\", \"value\": \"Production\"}]"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Analytics API 参考》[TagResource](#) 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 `untag-resource`。

### AWS CLI

要从资源中删除标签

以下 `untag-resource` 示例从指定资源中删除具有指定密钥名称的标签。

```
aws iotanalytics untag-resource \  
  --resource-arn "arn:aws:iotanalytics:us-west-2:123456789012:channel/mychannel" \  
  --tag-keys "[\"Environment\"]"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网分析 API 参考》中的 `UntagResource` < [https://docs.aws.amazon.com/iotanalytics/latest/APIReference/API\\_UntagResource.html](https://docs.aws.amazon.com/iotanalytics/latest/APIReference/API_UntagResource.html) >。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-channel

以下代码示例演示了如何使用 `update-channel`。

### AWS CLI

修改频道

以下 `update-channel` 示例修改了指定频道的设置。



```
aws iotanalytics update-channel \  
  --cli-input-json file://update-channel.json
```

update-channel.json 的内容：

```
{  
  "channelName": "mychannel",  
  "retentionPeriod": {  
    "numberOfDays": 92  
  }  
}
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Analytics API 参考》[UpdateChannel](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateChannel](#)中的。

## update-dataset

以下代码示例演示了如何使用 update-dataset。

### AWS CLI

#### 更新数据集

以下update-dataset示例修改了指定数据集的设置。

```
aws iotanalytics update-dataset \  
  --cli-input-json file://update-dataset.json
```

update-dataset.json 的内容：

```
{  
  "datasetName": "mydataset",  
  "actions": [  
    {  
      "actionName": "myDatasetUpdateAction",  
      "queryAction": {  
        "sqlQuery": "SELECT * FROM mydatastore"  
      }  
    }  
  ]  
}
```

```
    }
  }
],
"retentionPeriod": {
  "numberOfDays": 92
}
}
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网分析 API 参考》中的 UpdateDataset < [https://docs.aws.amazon.com/iotanalytics/latest/APIReference/API\\_UpdateDataset.html](https://docs.aws.amazon.com/iotanalytics/latest/APIReference/API_UpdateDataset.html) >。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDataset](#)中的。

## update-datastore

以下代码示例演示了如何使用 update-datastore。

### AWS CLI

#### 更新数据存储

以下update-datastore示例修改了指定数据存储的设置。

```
aws iotanalytics update-datastore \
  --cli-input-json file://update-datastore.json
```

update-datastore.json 的内容：

```
{
  "datastoreName": "mydatastore",
  "retentionPeriod": {
    "numberOfDays": 93
  }
}
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Analytics API 参考》[UpdateDatastore](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDatastore](#)中的。

## update-pipeline

以下代码示例演示了如何使用 update-pipeline。

### AWS CLI

#### 更新管道

以下update-pipeline示例修改了指定管道的设置。您必须在pipelineActivities数组中同时指定通道和数据存储活动，还可以指定多达 23 个其他活动。

```
aws iotanalytics update-pipeline \  
  --cli-input-json file://update-pipeline.json
```

update-pipeline.json 的内容：

```
{  
  "pipelineName": "mypipeline",  
  "pipelineActivities": [  
    {  
      "channel": {  
        "name": "myChannelActivity",  
        "channelName": "mychannel",  
        "next": "myMathActivity"  
      }  
    },  
    {  
      "datastore": {  
        "name": "myDatastoreActivity",  
        "datastoreName": "mydatastore"  
      }  
    },  
    {  
      "math": {  
        "name": "myMathActivity",  
        "math": "(((temp - 32) * 5.0) / 9.0) + 273.15",  
        "attribute": "tempK",  
        "next": "myDatastoreActivity"  
      }  
    }  
  ]  
}
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS IoT Analytics API 参考》[UpdatePipeline](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePipeline](#)中的。

## 使用“设备顾问”示例 AWS CLI

以下代码示例向您展示了如何使用 with Device Advisor 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-suite-definition**

以下代码示例演示了如何使用 create-suite-definition。

AWS CLI

示例 1：创建 IoT 设备顾问测试套件

以下create-suite-definition示例使用指定的套件定义配置在 AWS IoT 中创建设备顾问测试套件。

```
aws iotdeviceadvisor create-suite-definition \
  --suite-definition-configuration '{ \
    "suiteDefinitionName": "TestSuiteName", \
    "devices": [{"thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyIotThing"}], \
    "intendedForQualification": false, \
```

```

    "rootGroup": "{\\"configuration\\":{\\},\\"tests\\":[{\\"name\\":\\"MQTT Connect\\",
\\"configuration\\":{\\"EXECUTION_TIMEOUT\\":120},\\"tests\\":[{\\"name\\":\\"MQTT_Connect\\",
\\"configuration\\":{\\},\\"test\\":{\\"id\\":\\"MQTT_Connect\\",\\"testCase\\":null,\\"version
\\":\\"0.0.0\\"}}]}]}", \
    "devicePermissionRoleArn": "arn:aws:iam::123456789012:role/Myrole"}'

```

输出：

```

{
  "suiteDefinitionId": "0jtsgio7yenu",
  "suiteDefinitionArn": "arn:aws:iotdeviceadvisor:us-
east-1:123456789012:suitedefinition/0jtsgio7yenu",
  "suiteDefinitionName": "TestSuiteName",
  "createdAt": "2022-12-02T11:38:13.263000-05:00"
}

```

有关更多信息，请参阅《AWS IoT Core 开发者指南》中的[创建测试套件定义](#)。

#### 示例 2：创建 IoT 设备顾问最新资格测试套件

以下 create-suite-definition 示例使用 AWS 物联网中的最新版本和指定的套件定义配置创建设备顾问资格测试套件。

```

aws iotdeviceadvisor create-suite-definition \
  --suite-definition-configuration '{ \
    "suiteDefinitionName": "TestSuiteName", \
    "devices": [{"thingArn":"arn:aws:iot:us-east-1:123456789012:thing/
MyIoTThing"}], \
    "intendedForQualification": true, \
    "rootGroup": "", \
    "devicePermissionRoleArn": "arn:aws:iam::123456789012:role/Myrole"}'

```

输出：

```

{
  "suiteDefinitionId": "txgsuolk2myj",
  "suiteDefinitionArn": "arn:aws:iotdeviceadvisor:us-
east-1:123456789012:suitedefinition/txgsuolk2myj",
  "suiteDefinitionName": "TestSuiteName",
  "createdAt": "2022-12-02T11:38:13.263000-05:00"
}

```

有关更多信息，请参阅《AWS IoT Core 开发者指南》中的[创建测试套件定义](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSuiteDefinition](#)中的。

## delete-suite-definition

以下代码示例演示了如何使用 delete-suite-definition。

### AWS CLI

删除 IoT 设备顾问测试套件

以下delete-suite-definition示例删除具有指定套件定义 ID 的设备顾问测试套件。

```
aws iotdeviceadvisor delete-suite-definition \  
  --suite-definition-id 0jtsgio7yenu
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT API 参考[DeleteSuiteDefinition](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSuiteDefinition](#)中的。

## get-endpoint

以下代码示例演示了如何使用 get-endpoint。

### AWS CLI

示例 1：获取有关 IoT 设备顾问账户级端点的信息

以下get-endpoint示例获取有关设备顾问账户级别测试端点的信息。

```
aws iotdeviceadvisor get-endpoint
```

输出：

```
{  
  "endpoint": "t6y4c143x9sfo.deviceadvisor.iot.us-east-1.amazonaws.com"  
}
```

## 示例 2：获取有关 IoT 设备顾问设备级端点的信息

以下 `get-endpoint` 示例获取有关具有指定内容 `arn` 或 `certificate-arn` 的设备顾问设备级测试端点的信息。

```
aws iotdeviceadvisor get-endpoint \  
  --thing-arn arn:aws:iot:us-east-1:123456789012:thing/MyIotThing
```

输出：

```
{  
  "endpoint": "tdb7719be5t6y4c143x9sfo.deviceadvisor.iot.us-east-1.amazonaws.com"  
}
```

有关更多信息，请参阅 AWS IoT Core 开发者指南中的[获取测试端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetEndpoint](#) 中的。

## get-suite-definition

以下代码示例演示了如何使用 `get-suite-definition`。

### AWS CLI

获取有关 IoT 设备顾问测试套件的信息

以下 `get-suite-definition` 示例获取有关具有指定套件定义 ID 的 aevice Advisor 测试套件的信息。

```
aws iotdeviceadvisor get-suite-definition \  
  --suite-definition-id qqcsmtyyjabl
```

输出：

```
{  
  "suiteDefinitionId": "qqcsmtyyjabl",  
  "suiteDefinitionArn": "arn:aws:iotdeviceadvisor:us-  
east-1:123456789012:suitedefinition/qqcsmtyyjabl",  
  "suiteDefinitionVersion": "v1",  
  "latestVersion": "v1",
```

```

    "suiteDefinitionConfiguration": {
      "suiteDefinitionName": "MQTT connection",
      "devices": [],
      "intendedForQualification": false,
      "isLongDurationTest": false,
      "rootGroup": "{\"configuration\":{},\"tests\": [{\"id\": \"uta5d9j1kvwc\",
        \"name\": \"Test group 1\", \"configuration\": {}, \"tests\": [{\"id\": \"awr8pq5vc9yp\",
        \"name\": \"MQTT Connect\", \"configuration\": {}, \"test\": {\"id\": \"MQTT_Connect\",
        \"testCase\": null, \"version\": \"0.0.0\"}]}]}]}",
      "devicePermissionRoleArn": "arn:aws:iam::123456789012:role/Myrole",
      "protocol": "MqttV3_1_1"
    },
    "createdAt": "2022-11-11T22:28:52.389000-05:00",
    "lastModifiedAt": "2022-11-11T22:28:52.389000-05:00",
    "tags": {}
  }
}

```

有关更多信息，请参阅 AWS IoT Core 开发者指南中的[获取测试套件定义](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetSuiteDefinition](#)中的。

## get-suite-run-report

以下代码示例演示了如何使用 get-suite-run-report。

### AWS CLI

要获取有关 IoT 设备顾问合格测试套件的信息，请运行报告

以下 get-suite-run-report 示例获取了使用指定套件定义 ID 和套件运行 ID 成功运行的设备顾问合格测试套件的报告下载链接。

```

aws iotdeviceadvisor get-suite-run-report \
  --suite-definition-id ztvb5aek4w4x \
  --suite-run-id p6awv83nre6v

```

输出：

```

{
  "qualificationReportDownloadUrl": "https://senate-apn-reports-us-east-1-
  prod.s3.amazonaws.com/report.downloadlink"
}

```



有关更多信息，请参阅《AWS IoT Core 开发者指南》中的[“获取成功运行的资格测试套件的资格报告”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSuiteRunReport](#)中的。

## get-suite-run

以下代码示例演示了如何使用 get-suite-run。

### AWS CLI

获取有关 IoT 设备顾问测试套件运行状态的信息

以下get-suite-run示例使用指定的套件定义 ID 和套件运行 ID 获取有关设备顾问测试套件运行状态的信息。

```
aws iotdeviceadvisor get-suite-run \  
  --suite-definition-id qqcsmtyyjabl \  
  --suite-run-id nzlfyhaa18oa
```

输出：

```
{  
  "suiteDefinitionId": "qqcsmtyyjabl",  
  "suiteDefinitionVersion": "v1",  
  "suiteRunId": "nzlfyhaa18oa",  
  "suiteRunArn": "arn:aws:iotdeviceadvisor:us-east-1:123456789012:suiterun/  
qqcsmtyyjabl/nzlfyhaa18oa",  
  "suiteRunConfiguration": {  
    "primaryDevice": {  
      "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyIotThing",  
      "certificateArn": "arn:aws:iot:us-east-1:123456789012:cert/certFile"  
    },  
    "parallelRun": false  
  },  
  "testResult": {  
    "groups": [  
      {  
        "groupId": "uta5d9j1kvwc",  
        "groupName": "Test group 1",  
        "tests": [  
          {  
            "testCaseRunId": "2ve2twrqyr0s",
```

```

        "testCaseDefinitionId": "awr8pq5vc9yp",
        "testCaseDefinitionName": "MQTT Connect",
        "status": "PASS",
        "startTime": "2022-11-12T00:01:53.693000-05:00",
        "endTime": "2022-11-12T00:02:15.443000-05:00",
        "logUrl": "https://console.aws.amazon.com/
cloudwatch/home?region=us-east-1#logEventViewer:group=/aws/iot/deviceadvisor/
qqcsmtyyjabl;stream=nzlfyhaa18oa_2ve2twrqyr0s",
        "warnings": "null",
        "failure": "null"
    }
  ]
}
},
"startTime": "2022-11-12T00:01:52.673000-05:00",
"endTime": "2022-11-12T00:02:16.496000-05:00",
"status": "PASS",
"tags": {}
}

```

有关更多信息，请参阅《AWS 物联网核心开发者指南》中的启动[测试套件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetSuiteRun](#)中的。

## list-suite-definitions

以下代码示例演示了如何使用 list-suite-definitions。

### AWS CLI

示例 1：列出您创建的 IoT 设备顾问测试套件

以下 list-suite-definitions 示例列出了您在 AWS 物联网中创建的最多 25 个设备顾问测试套件。如果您有超过 25 个测试套件，则输出中将显示“nextToken”。您可以使用这个“nextToken”来显示你创建的其余测试套件。

```
aws iotdeviceadvisor list-suite-definitions
```

输出：

```
{
```

```

    "suiteDefinitionInformationList": [
      {
        "suiteDefinitionId": "3hsn88h4p2g5",
        "suiteDefinitionName": "TestSuite1",
        "defaultDevices": [
          {
            "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/
MyIotThing"
          }
        ],
        "intendedForQualification": false,
        "isLongDurationTest": false,
        "protocol": "MqttV3_1_1",
        "createdAt": "2022-11-17T14:15:56.830000-05:00"
      },
      {
        .....
      }
    ],
    "nextToken": "nextTokenValue"
  }

```

## 示例 2：列出您使用指定设置创建的 IoT 设备顾问测试套件

以下 `list-suite-definitions` 示例列出了您在 AWS 物联网中创建的具有指定最大结果编号的设备顾问测试套件。如果您的测试套件数量超过最大数量，则输出中将显示 “nextToken”。如果您有 “nextToken”，则可以使用 “nextToken” 来显示您创建的以前未显示的测试套件。

```

aws iotdeviceadvisor list-suite-definitions \
  --max-result 1 \
  --next-token "nextTokenValue"

```

输出：

```

{
  "suiteDefinitionInformationList": [
    {
      "suiteDefinitionId": "ztvb5aew4w4x",
      "suiteDefinitionName": "TestSuite2",
      "defaultDevices": [],
      "intendedForQualification": true,
      "isLongDurationTest": false,

```

```
        "protocol": "MqttV3_1_1",
        "createdAt": "2022-11-17T14:15:56.830000-05:00"
      }
    ],
    "nextToken": "nextTokenValue"
  }
```

有关更多信息，请参阅 AWS IoT API 参考[ListSuiteDefinitions](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListSuiteDefinitions](#)中的。

## list-suite-runs

以下代码示例演示了如何使用 `list-suite-runs`。

### AWS CLI

示例 1：列出有关指定 IoT 设备顾问测试套件运行状态的所有信息

以下 `list-suite-runs` 示例列出了有关具有指定套件定义 ID 的设备顾问测试套件运行状态的所有信息。如果您运行的测试套件超过 25 个，则输出中将显示 “nextToken”。你可以使用这个 “nextToken” 来显示测试套件的其余运行情况。

```
aws iotdeviceadvisor list-suite-runs \
  --suite-definition-id ztvb5aew4w4x
```

输出：

```
{
  "suiteRunsList": [
    {
      "suiteDefinitionId": "ztvb5aew4w4x",
      "suiteDefinitionVersion": "v1",
      "suiteDefinitionName": "TestSuite",
      "suiteRunId": "p6awv89nre6v",
      "createdAt": "2022-12-01T16:33:14.212000-05:00",
      "startedAt": "2022-12-01T16:33:15.710000-05:00",
      "endAt": "2022-12-01T16:42:03.323000-05:00",
      "status": "PASS",
      "passed": 6,
      "failed": 0
    }
  ]
}
```

```
]
}
```

示例 2：要列出有关指定 IoT Device Advisor 测试套件的信息，请使用指定设置运行状态

以下 `list-suite-runs` 示例列出了有关具有指定套件定义 ID 和指定的最大结果编号的设备顾问测试套件运行状态的信息。如果您的测试套件运行次数超过了最大数量，则输出中将显示“nextToken”。如果你有“nextToken”，您可以使用“nextToken”来显示以前未显示的测试套件运行情况。

```
aws iotdeviceadvisor list-suite-runs \
  --suite-definition-id qqcsmtyyjam1 \
  --max-result 1 \
  --next-token "nextTokenValue"
```

输出：

```
{
  "suiteRunsList": [
    {
      "suiteDefinitionId": "qqcsmtyyjam1",
      "suiteDefinitionVersion": "v1",
      "suiteDefinitionName": "MQTT connection",
      "suiteRunId": "gz9vm2s6d2jy",
      "createdAt": "2022-12-01T20:10:27.079000-05:00",
      "startedAt": "2022-12-01T20:10:28.003000-05:00",
      "endAt": "2022-12-01T20:10:45.084000-05:00",
      "status": "STOPPED",
      "passed": 0,
      "failed": 0
    }
  ],
  "nextToken": "nextTokenValue"
}
```

有关更多信息，请参阅 AWS IoT API 参考 [ListSuiteRuns](#) 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListSuiteRuns](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

## AWS CLI

列出附加到 IoT 设备顾问资源的标签

以下 `list-tags-for-resource` 示例列出了附加到设备顾问资源的标签。设备顾问资源可以是 `Suitedefinition-Arn` 或 `Suiterun-Arn`。

```
aws iotdeviceadvisor list-tags-for-resource \  
  --resource-arn arn:aws:iotdeviceadvisor:us-east-1:123456789012:suitedefinition/  
ba0uyjpg38ny
```

输出：

```
{  
  "tags": {  
    "TestTagKey": "TestTagValue"  
  }  
}
```

有关更多信息，请参阅 [AWS IoT API 参考](#) 和《[服务授权参考](#)》[ListTagsForResource](#) 中的 [AWS IoT Core Device Advisor 定义的资源类型](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [ListTagsForResource](#) 中的。

## start-suite-run

以下代码示例演示了如何使用 `start-suite-run`。

### AWS CLI

要启动 IoT 设备顾问测试套件，请运行

以下 `start-suite-run` 示例列出了您 AWS 账户中可用的微件。

```
aws iotdeviceadvisor start-suite-run \  
  --suite-definition-id qqcsmtyyjabl \  
  --suite-definition-version v1 \  
  --suite-run-configuration '{"primaryDevice":{"thingArn": "arn:aws:iot:us-  
east-1:123456789012:thing/MyIotThing", "certificateArn": "arn:aws:iot:us-  
east-1:123456789012:cert/certFile"}}'
```

输出：

```
{
  "suiteRunId": "pwmucgw7lt9s",
  "suiteRunArn": "arn:aws:iotdeviceadvisor:us-east-1:123456789012:suiterun/
qqcsmtyyjabl/pwmucgw7lk9s",
  "createdAt": "2022-12-02T15:43:05.581000-05:00"
}
```

有关更多信息，请参阅 AWS IoT Core 开发者指南 [中的启动测试套件运行](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartSuiteRun](#) 中的。

## stop-suite-run

以下代码示例演示了如何使用 stop-suite-run。

### AWS CLI

停止当前正在运行的 IoT 设备顾问测试套件

以下 stop-suite-run 示例停止了当前正在使用指定套件定义 ID 和套件运行 ID 运行的设备顾问测试套件。

```
aws iotdeviceadvisor stop-suite-run \
  --suite-definition-id qqcsmtyyjabl \
  --suite-run-id nzlfyhaa18oa
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网核心开发者指南》中的 [停止测试套件运行](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StopSuiteRun](#) 中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

添加和修改 IoT 设备顾问资源的现有标签

以下 `tag-resource` 示例使用指定的资源 `arn` 和标签添加和修改设备顾问资源的现有标签。设备顾问资源可以是 `Suitedefinition-Arn` 或 `Suiterun-Arn`。

```
aws iotdeviceadvisor tag-resource \  
  --resource-arn arn:aws:iotdeviceadvisor:us-east-1:123456789012:suitedefinition/  
ba0uyjpg38ny \  
  --tags '{"TagKey": "TagValue"}
```

此命令不生成任何输出。

有关更多信息，请参阅 [AWS IoT API 参考](#) 和《[服务授权参考](#)》[TagResource](#) 中的 [AWS IoT Core Device Advisor 定义的资源类型](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 `untag-resource`。

### AWS CLI

从 IoT 设备顾问资源中移除现有标签

以下 `untag-resource` 示例使用指定的资源 `arn` 和标签密钥从设备顾问资源中删除现有标签。设备顾问资源可以是 `Suitedefinition-Arn` 或 `Suiterun-Arn`。

```
aws iotdeviceadvisor untag-resource \  
  --resource-arn arn:aws:iotdeviceadvisor:us-east-1:123456789012:suitedefinition/  
ba0uyjpg38ny \  
  --tag-keys "TagKey"
```

此命令不生成任何输出。

有关更多信息，请参阅 [AWS IoT API 参考](#) 和《[服务授权参考](#)》[UntagResource](#) 中的 [AWS IoT Core Device Advisor 定义的资源类型](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [UntagResource](#) 中的。

## update-suite-definition

以下代码示例演示了如何使用 `update-suite-definition`。



## AWS CLI

## 示例 1：更新 IoT 设备顾问测试套件

以下update-suite-definition示例使用指定的套件定义 ID 和套件定义配置更新 AWS IoT 中的设备顾问测试套件。

```
aws iotdeviceadvisor update-suite-definition \
  --suite-definition-id 3hsn88h4p2g5 \
  --suite-definition-configuration '{ \
    "suiteDefinitionName": "TestSuiteName", \
    "devices": [{"thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyIotThing"}], \
    "intendedForQualification": false, \
    "rootGroup": {"configuration\": {}, "tests\": [{"name\": "MQTT Connect", \
  \configuration\": {"EXECUTION_TIMEOUT\": 120}, "tests\": [{"name\": "MQTT_Connect", \
  \configuration\": {}, "test\": {"id\": "MQTT_Connect", "testCase\": null, "version \
  \": "0.0.0"}]}]}], \
    "devicePermissionRoleArn": "arn:aws:iam::123456789012:role/Myrole"}
```

输出：

```
{
  "suiteDefinitionId": "3hsn88h4p2g5",
  "suiteDefinitionName": "TestSuiteName",
  "suiteDefinitionVersion": "v3",
  "createdAt": "2022-11-17T14:15:56.830000-05:00",
  "lastUpdatedAt": "2022-12-02T16:02:45.857000-05:00"
}
```

## 示例 2：更新 IoT 设备顾问资格测试套件

以下update-suite-definition示例使用指定的套件定义 ID 和套件定义配置更新 AWS 物联网中的设备顾问资格测试套件。

```
aws iotdeviceadvisor update-suite-definition \
  --suite-definition-id txgsuolk2myj \
  --suite-definition-configuration '{
    "suiteDefinitionName": "TestSuiteName", \
    "devices": [{"thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyIotThing"}], \
    "intendedForQualification": true, \
```

```
"rootGroup": "", \  
"devicePermissionRoleArn": "arn:aws:iam::123456789012:role/Myrole"]}'
```

输出：

```
{  
  "suiteDefinitionId": "txgsuolk2myj",  
  "suiteDefinitionName": "TestSuiteName",  
  "suiteDefinitionVersion": "v3",  
  "createdAt": "2022-11-17T14:15:56.830000-05:00",  
  "lastUpdatedAt": "2022-12-02T16:02:45.857000-05:00"  
}
```

有关更多信息，请参阅 AWS IoT API 参考[UpdateSuiteDefinition](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateSuiteDefinition](#)中的。

## AWS IoT data 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS IoT data。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **delete-thing-shadow**

以下代码示例演示了如何使用 `delete-thing-shadow`。

AWS CLI

删除设备的影子文档

以下`delete-thing-shadow`示例删除名为的设备整个影子文档MyRPi。

```
aws iot-data delete-thing-shadow \  
  --thing-name MyRPi \  
  "output.txt"
```

该命令不会在显示屏上生成任何输出，但`output.txt`包含确认您删除的影子文档的版本和时间戳的信息。

```
{"version":2,"timestamp":1560270384}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的[使用阴影](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteThingShadow](#)中的。

## get-thing-shadow

以下代码示例演示了如何使用 `get-thing-shadow`。

### AWS CLI

获取事物影子文档

以下`get-thing-shadow`示例获取了指定 IoT 事物的事物影子文档。

```
aws iot-data get-thing-shadow \  
  --thing-name MyRPi \  
  output.txt
```

该命令不会在显示屏上生成任何输出，但以下显示了以下内容`output.txt`：

```
{  
  "state":{  
    "reported":{  
      "moisture":"low"  
    }  
  },  
  "metadata":{  
    "reported":{  
      "moisture":{
```

```
    "timestamp":1560269319
  }
}
},
"version":1,"timestamp":1560269405
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的 Device Shadow [w 服务数据流](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetThingShadow](#)中的。

## update-thing-shadow

以下代码示例演示了如何使用 update-thing-shadow。

### AWS CLI

#### 更新事物影子

以下update-thing-shadow示例修改了指定事物的设备影子的当前状态并将其保存到文件output.txt中。

```
aws iot-data update-thing-shadow \
  --thing-name MyRPi \
  --payload '{"state":{"reported":{"moisture":"okay"}}}' \
  "output.txt"
```

该命令不会在显示屏上生成任何输出，但以下显示了以下内容output.txt：

```
{
  "state": {
    "reported": {
      "moisture": "okay"
    }
  },
  "metadata": {
    "reported": {
      "moisture": {
        "timestamp": 1560270036
      }
    }
  },
}
```

```
"version": 2,  
"timestamp": 1560270036  
}
```

有关更多信息，请参阅《AWS 物联网开发人员指南》中的 [Device Shadow 服务数据流](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateThingShadow](#) 中的。

## AWS IoT Events 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS IoT Events。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **batch-put-message**

以下代码示例演示了如何使用 `batch-put-message`。

AWS CLI

向 AWS IoT Events 发送消息 ( 输入 )

以下 `batch-put-message` 示例向 AWS IoT Events 系统发送了一组消息。每个消息有效载荷都将转换为您指定的输入 (`inputName`)，并输入到任何监视该输入的检测器中。如果发送了多条消息，则无法保证消息的处理顺序。为了保证顺序，您必须逐一发送一条消息，然后等待成功响应。

```
aws iotevents-data batch-put-message \  
  --cli-input-json file://highPressureMessage.json
```

highPressureMessage.json 的内容：

```
{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "PressureInput",
      "payload": "{\"motorid\": \"Fulton-A32\", \"sensorData\": {\"pressure\": 80, \"temperature\": 39} }"
    }
  ]
}
```

输出：

```
{
  "BatchPutMessageErrorEntries": []
}
```

有关更多信息，请参阅 AWS IoT Events API 参考[BatchPutMessage](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchPutMessage](#)中的。

## batch-update-detector

以下代码示例演示了如何使用 batch-update-detector。

AWS CLI

更新探测器（实例）

以下batch-update-detector示例更新了指定探测器模型的一个或多个探测器（实例）的状态、变量值和计时器设置。

```
aws iotevents-data batch-update-detector \
  --cli-input-json file://budFulton-A32.json
```

budFulton-A32.json 的内容：

```
{
```

```
"detectors": [
  {
    "messageId": "00001",
    "detectorModelName": "motorDetectorModel",
    "keyValue": "Fulton-A32",
    "state": {
      "stateName": "Normal",
      "variables": [
        {
          "name": "pressureThresholdBreached",
          "value": "0"
        }
      ],
      "timers": [
      ]
    }
  }
]
```

输出：

```
{
  "batchUpdateDetectorErrorEntries": []
}
```

有关更多信息，请参阅 AWS IoT Events API 参考[BatchUpdateDetector](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchUpdateDetector](#)中的。

## create-detector-model

以下代码示例演示了如何使用 create-detector-model。

### AWS CLI

#### 创建探测器模型

以下create-detector-model示例创建了一个探测器模型，其配置由参数文件指定。

```
aws iotevents create-detector-model \
  --cli-input-json file://motorDetectorModel.json
```

motorDetectorModel.json 的内容：

```
{
  "detectorModelName": "motorDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "Normal",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "pressureThresholdBreach",
                    "value": "0"
                  }
                }
              ]
            }
          ]
        },
        "onInput": {
          "transitionEvents": [
            {
              "eventName": "Overpressurized",
              "condition": "$input.PressureInput.sensorData.pressure
                > 70",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "pressureThresholdBreach",
                    "value":
                      "$variable.pressureThresholdBreach + 3"
                  }
                }
              ],
              "nextState": "Dangerous"
            }
          ]
        }
      }
    ]
  }
},
```



```

    {
      "stateName": "Dangerous",
      "onEnter": {
        "events": [
          {
            "eventName": "Pressure Threshold Breached",
            "condition": "$variable.pressureThresholdBreached >
1",
            "actions": [
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-
east-1:123456789012:underPressureAction"
                }
              }
            ]
          }
        ],
      },
      "onInput": {
        "events": [
          {
            "eventName": "Overpressurized",
            "condition": "$input.PressureInput.sensorData.pressure
&gt; 70",
            "actions": [
              {
                "setVariable": {
                  "variableName": "pressureThresholdBreached",
                  "value": "3"
                }
              }
            ]
          }
        ],
      },
      {
        "eventName": "Pressure Okay",
        "condition": "$input.PressureInput.sensorData.pressure
&lt;= 70",
        "actions": [
          {
            "setVariable": {
              "variableName": "pressureThresholdBreached",
              "value":
"$variable.pressureThresholdBreached - 1"

```

```

    }
  }
]
},
"transitionEvents": [
  {
    "eventName": "BackToNormal",
    "condition": "$input.PressureInput.sensorData.pressure
&lt;= 70 &amp;&amp; $variable.pressureThresholdBreached &lt;= 1",
    "nextState": "Normal"
  }
],
"onExit": {
  "events": [
    {
      "eventName": "Normal Pressure Restored",
      "condition": "true",
      "actions": [
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-
east-1:123456789012:pressureClearedAction"
          }
        }
      ]
    }
  ]
}
},
"initialStateName": "Normal"
},
"key": "motorid",
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}

```

输出：

```

{
  "detectorModelConfiguration": {
    "status": "ACTIVATING",

```

```
    "lastUpdateTime": 1560796816.077,
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
    "creationTime": 1560796816.077,
    "detectorModelArn": "arn:aws:iotevents:us-west-2:123456789012:detectorModel/
motorDetectorModel",
    "key": "motorid",
    "detectorModelName": "motorDetectorModel",
    "detectorModelVersion": "1"
  }
}
```

有关更多信息，请参阅 AWS IoT Events API 参考[CreateDetectorModel](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateDetectorModel](#)中的。

## create-input

以下代码示例演示了如何使用 create-input。

### AWS CLI

#### 创建输入

以下 create-input 示例创建了一个输入。

```
aws iotevents create-input \
  --cli-input-json file://pressureInput.json
```

pressureInput.json 的内容：

```
{
  "inputName": "PressureInput",
  "inputDescription": "Pressure readings from a motor",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "sensorData.pressure" },
      { "jsonPath": "motorid" }
    ]
  }
}
```

输出：

```
{
  "inputConfiguration": {
    "status": "ACTIVE",
    "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/PressureInput",
    "lastUpdateTime": 1560795312.542,
    "creationTime": 1560795312.542,
    "inputName": "PressureInput",
    "inputDescription": "Pressure readings from a motor"
  }
}
```

有关更多信息，请参阅 AWS IoT Events API 参考[CreateInput](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateInput](#)中的。

## delete-detector-model

以下代码示例演示了如何使用 delete-detector-model。

### AWS CLI

#### 删除探测器模型

以下delete-detector-model示例删除了指定的探测器模型。探测器模型的所有活动实例也将被删除。

```
aws iotevents delete-detector-model \
  --detector-model-name motorDetectorModel
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT Events API 参考[DeleteDetectorModel](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDetectorModel](#)中的。

## delete-input

以下代码示例演示了如何使用 delete-input。

### AWS CLI

#### 删除输入

以下delete-input示例删除了指定的输入。

```
aws iotevents delete-input \  
  --input-name PressureInput
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT Events API 参考[DeleteInput](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInput](#)中的。

## describe-detector-model

以下代码示例演示了如何使用 describe-detector-model。

### AWS CLI

获取有关探测器模型的信息

以下describe-detector-model示例显示了指定探测器型号的详细信息。由于未指定该version参数，因此会返回有关最新版本的信息。

```
aws iotevents describe-detector-model \  
  --detector-model-name motorDetectorModel
```

输出：

```
{  
  "detectorModel": {  
    "detectorModelConfiguration": {  
      "status": "ACTIVE",  
      "lastUpdateTime": 1560796816.077,  
      "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",  
      "creationTime": 1560796816.077,  
      "detectorModelArn": "arn:aws:iotevents:us-  
west-2:123456789012:detectorModel/motorDetectorModel",  
      "key": "motorid",  
      "detectorModelName": "motorDetectorModel",  
      "detectorModelVersion": "1"  
    },  
    "detectorModelDefinition": {  
      "states": [  
        {
```

```

        "onInput": {
            "transitionEvents": [
                {
                    "eventName": "Overpressurized",
                    "actions": [
                        {
                            "setVariable": {
                                "variableName":
"pressureThresholdBreach",
                                "value":
"$variable.pressureThresholdBreach + 3"
                            }
                        }
                    ],
                    "condition":
"$input.PressureInput.sensorData.pressure > 70",
                    "nextState": "Dangerous"
                }
            ],
            "events": []
        },
        "stateName": "Normal",
        "onEnter": {
            "events": [
                {
                    "eventName": "init",
                    "actions": [
                        {
                            "setVariable": {
                                "variableName":
"pressureThresholdBreach",
                                "value": "0"
                            }
                        }
                    ],
                    "condition": "true"
                }
            ]
        },
        "onExit": {
            "events": []
        }
    },
    {

```

```
    "onInput": {
      "transitionEvents": [
        {
          "eventName": "BackToNormal",
          "actions": [],
          "condition":
"$input.PressureInput.sensorData.pressure <= 70 &&
$variable.pressureThresholdBreached <= 1",
          "nextState": "Normal"
        }
      ],
      "events": [
        {
          "eventName": "Overpressurized",
          "actions": [
            {
              "setVariable": {
                "variableName":
"pressureThresholdBreached",
                "value": "3"
              }
            }
          ],
          "condition":
"$input.PressureInput.sensorData.pressure > 70"
        },
        {
          "eventName": "Pressure Okay",
          "actions": [
            {
              "setVariable": {
                "variableName":
"pressureThresholdBreached",
                "value":
"$variable.pressureThresholdBreached - 1"
              }
            }
          ],
          "condition":
"$input.PressureInput.sensorData.pressure <= 70"
        }
      ]
    },
    "stateName": "Dangerous",
```

```

        "onEnter": {
            "events": [
                {
                    "eventName": "Pressure Threshold Breached",
                    "actions": [
                        {
                            "sns": {
                                "targetArn": "arn:aws:sns:us-
east-1:123456789012:underPressureAction"
                            }
                        }
                    ],
                    "condition": "$variable.pressureThresholdBreached >
1"
                }
            ]
        },
        "onExit": {
            "events": [
                {
                    "eventName": "Normal Pressure Restored",
                    "actions": [
                        {
                            "sns": {
                                "targetArn": "arn:aws:sns:us-
east-1:123456789012:pressureClearedAction"
                            }
                        }
                    ],
                    "condition": "true"
                }
            ]
        }
    ],
    "initialStateName": "Normal"
}
}
}

```

有关更多信息，请参阅 AWS IoT Events API 参考[DescribeDetectorModel](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeDetectorModel](#)中的。



## describe-detector

以下代码示例演示了如何使用 describe-detector。

### AWS CLI

获取有关探测器 (实例) 的信息。

以下 describe-detector 示例显示了指定探测器 (实例) 的详细信息。

```
aws iotevents-data describe-detector \  
  --detector-model-name motorDetectorModel \  
  --key-value "Fulton-A32"
```

输出：

```
{  
  "detector": {  
    "lastUpdateTime": 1560797852.776,  
    "creationTime": 1560797852.775,  
    "state": {  
      "variables": [  
        {  
          "name": "pressureThresholdBreached",  
          "value": "3"  
        }  
      ],  
      "stateName": "Dangerous",  
      "timers": []  
    },  
    "keyValue": "Fulton-A32",  
    "detectorModelName": "motorDetectorModel",  
    "detectorModelVersion": "1"  
  }  
}
```

有关更多信息，请参阅 AWS IoT Events API 参考 [DescribeDetector](#) 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDetector](#) 中的。

## describe-input

以下代码示例演示了如何使用 describe-input。

## AWS CLI

获取有关输入的信息

以下describe-input示例显示了指定输入的详细信息。

```
aws iotevents describe-input \  
  --input-name PressureInput
```

输出：

```
{  
  "input": {  
    "inputConfiguration": {  
      "status": "ACTIVE",  
      "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/  
PressureInput",  
      "lastUpdateTime": 1560795312.542,  
      "creationTime": 1560795312.542,  
      "inputName": "PressureInput",  
      "inputDescription": "Pressure readings from a motor"  
    },  
    "inputDefinition": {  
      "attributes": [  
        {  
          "jsonPath": "sensorData.pressure"  
        },  
        {  
          "jsonPath": "motorid"  
        }  
      ]  
    }  
  }  
}
```

有关更多信息，请参阅 AWS IoT Events API 参考[DescribeInput](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInput](#)中的。

### describe-logging-options

以下代码示例演示了如何使用 describe-logging-options。

## AWS CLI

获取有关日志设置的信息

以下describe-logging-options示例检索 AWS IoT Events 日志选项的当前设置。

```
aws iotevents describe-logging-options
```

输出：

```
{
  "loggingOptions": {
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
    "enabled": false,
    "level": "ERROR"
  }
}
```

有关更多信息，请参阅 AWS IoT Events API 参考[DescribeLoggingOptions](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoggingOptions](#)中的。

## list-detector-model-versions

以下代码示例演示了如何使用 list-detector-model-versions。

### AWS CLI

获取有关探测器模型版本的信息

以下list-detector-model-versions示例列出了探测器模型的所有版本。仅返回与每个探测器模型版本关联的元数据。

```
aws iotevents list-detector-model-versions \
  --detector-model-name motorDetectorModel
```

输出：

```
{
  "detectorModelVersionSummaries": [
    {
      "status": "ACTIVE",
```

```
        "lastUpdateTime": 1560796816.077,
        "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
        "creationTime": 1560796816.077,
        "detectorModelArn": "arn:aws:iotevents:us-
west-2:123456789012:detectorModel/motorDetectorModel",
        "detectorModelName": "motorDetectorModel",
        "detectorModelVersion": "1"
    }
]
}
```

有关更多信息，请参阅 AWS IoT Events API 参考[ListDetectorModelVersions](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDetectorModelVersions](#)中的。

## list-detector-models

以下代码示例演示了如何使用 list-detector-models。

### AWS CLI

获取您的探测器型号列表

以下list-detector-models示例列出了您创建的探测器模型。仅返回与每个检测器模型关联的元数据。

```
aws iotevents list-detector-models
```

输出：

```
{
  "detectorModelSummaries": [
    {
      "detectorModelName": "motorDetectorModel",
      "creationTime": 1552072424.212
      "detectorModelDescription": "Detect overpressure in a motor."
    }
  ]
}
```

有关更多信息，请参阅 AWS IoT Events API 参考[ListDetectorModels](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDetectorModels](#)中的。

## list-detectors

以下代码示例演示了如何使用 list-detectors。

### AWS CLI

获取探测器型号的探测器列表

以下list-detectors示例列出了您账户中的探测器（探测器模型的实例）。

```
aws iotevents-data list-detectors \  
  --detector-model-name motorDetectorModel
```

输出：

```
{  
  "detectorSummaries": [  
    {  
      "lastUpdateTime": 1558129925.2,  
      "creationTime": 1552073155.527,  
      "state": {  
        "stateName": "Normal"  
      },  
      "keyValue": "Fulton-A32",  
      "detectorModelName": "motorDetectorModel",  
      "detectorModelVersion": "1"  
    }  
  ]  
}
```

有关更多信息，请参阅 AWS IoT Events API 参考[ListDetectors](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDetectors](#)中的。

## list-inputs

以下代码示例演示了如何使用 list-inputs。

### AWS CLI

列出输入

以下list-inputs示例列出了您在账户中创建的输入。

```
aws iotevents list-inputs
```

此命令不生成任何输出。输出：

```
{
  {
    "status": "ACTIVE",
    "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/PressureInput",
    "lastUpdateTime": 1551742986.768,
    "creationTime": 1551742986.768,
    "inputName": "PressureInput",
    "inputDescription": "Pressure readings from a motor"
  }
}
```

有关更多信息，请参阅 AWS IoT Events API 参考[ListInputs](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListInputs](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出分配给资源的标签。

以下 `list-tags-for-resource` 示例列出了您为资源分配的标签键名称和值。

```
aws iotevents list-tags-for-resource \
  --resource-arn "arn:aws:iotevents:us-west-2:123456789012:input/PressureInput"
```

输出：

```
{
  "tags": [
    {
      "value": "motor",
      "key": "deviceType"
    }
  ]
}
```

有关更多信息，请参阅 AWS IoT Events API 参考[ListTagsForResource](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## put-logging-options

以下代码示例演示了如何使用 put-logging-options。

### AWS CLI

#### 设置日志选项

以下put-logging-options示例设置或更新 AWS IoT Events 日志选项。如果您更新任何loggingOptions` field, it can take up to one minute for the change to take effect. Also, if you change the policy attached to the role you specified in the ``roleArn字段的值（例如，更正无效的策略），则该更改最多可能需要五分钟才能生效。

```
aws iotevents put-logging-options \  
  --cli-input-json file://logging-options.json
```

logging-options.json 的内容：

```
{  
  "loggingOptions": {  
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",  
    "level": "DEBUG",  
    "enabled": true,  
    "detectorDebugOptions": [  
      {  
        "detectorModelName": "motorDetectorModel",  
        "keyValue": "Fulton-A32"  
      }  
    ]  
  }  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT Events API 参考[PutLoggingOptions](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutLoggingOptions](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源添加标签

以下tag-resource示例添加或修改（如果密钥deviceType已存在）附加到指定资源的标签。

```
aws iotevents tag-resource \  
  --cli-input-json file://pressureInput.tag.json
```

pressureInput.tag.json 的内容：

```
{  
  "resourceArn": "arn:aws:iotevents:us-west-2:123456789012:input/PressureInput",  
  "tags": [  
    {  
      "key": "deviceType",  
      "value": "motor"  
    }  
  ]  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT Events API 参考[TagResource](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

要从资源中删除标签

以下untag-resource示例从指定资源中删除具有指定密钥名称的标签。

```
aws iotevents untag-resource \  
  --resource-arn arn:aws:iotevents:us-west-2:123456789012:input/PressureInput \  
  --tag-key deviceType
```



```
--tagkeys deviceType
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT Events API 参考 [UntagResource](#) 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-detector-model

以下代码示例演示了如何使用 `update-detector-model`。

### AWS CLI

#### 更新探测器模型

以下 `update-detector-model` 示例更新了指定的探测器型号。先前版本生成的探测器（实例）会被删除，然后在新输入到来时重新创建。

```
aws iotevents update-detector-model \  
  --cli-input-json file://motorDetectorModel.update.json
```

`motorDetectorModel.update.json` 的内容：

```
{  
  "detectorModelName": "motorDetectorModel",  
  "detectorModelDefinition": {  
    "states": [  
      {  
        "stateName": "Normal",  
        "onEnter": {  
          "events": [  
            {  
              "eventName": "init",  
              "condition": "true",  
              "actions": [  
                {  
                  "setVariable": {  
                    "variableName": "pressureThresholdBreach",  
                    "value": "0"  
                  }  
                }  
              ]  
            }  
          ]  
        }  
      ]  
    }  
  }  
}
```

```

    }
  ]
},
"onInput": {
  "transitionEvents": [
    {
      "eventName": "Overpressurized",
      "condition": "$input.PressureInput.sensorData.pressure >
70",
      "actions": [
        {
          "setVariable": {
            "variableName": "pressureThresholdBreach",
            "value":
"$variable.pressureThresholdBreach + 3"
          }
        }
      ],
      "nextState": "Dangerous"
    }
  ]
},
{
  "stateName": "Dangerous",
  "onEnter": {
    "events": [
      {
        "eventName": "Pressure Threshold Breached",
        "condition": "$variable.pressureThresholdBreach > 1",
        "actions": [
          {
            "sns": {
              "targetArn": "arn:aws:sns:us-
east-1:123456789012:underPressureAction"
            }
          }
        ]
      }
    ]
  },
  "onInput": {
    "events": [
      {

```

```

    "eventName": "Overpressurized",
    "condition": "$input.PressureInput.sensorData.pressure >
70",
    "actions": [
        {
            "setVariable": {
                "variableName": "pressureThresholdBreached",
                "value": "3"
            }
        }
    ]
},
{
    "eventName": "Pressure Okay",
    "condition": "$input.PressureInput.sensorData.pressure
<= 70",
    "actions": [
        {
            "setVariable": {
                "variableName": "pressureThresholdBreached",
                "value":
"$variable.pressureThresholdBreached - 1"
            }
        }
    ]
},
],
"transitionEvents": [
    {
        "eventName": "BackToNormal",
        "condition": "$input.PressureInput.sensorData.pressure
<= 70 && $variable.pressureThresholdBreached <= 1",
        "nextState": "Normal"
    }
]
},
"onExit": {
    "events": [
        {
            "eventName": "Normal Pressure Restored",
            "condition": "true",
            "actions": [
                {
                    "sns": {

```

```

        "targetArn": "arn:aws:sns:us-
east-1:123456789012:pressureClearedAction"
    }
}
]
}
]
}
],
"initialStateName": "Normal"
},
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}

```

输出：

```

{
  "detectorModelConfiguration": {
    "status": "ACTIVATING",
    "lastUpdateTime": 1560799387.719,
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
    "creationTime": 1560799387.719,
    "detectorModelArn": "arn:aws:iotevents:us-west-2:123456789012:detectorModel/
motorDetectorModel",
    "key": "motorid",
    "detectorModelName": "motorDetectorModel",
    "detectorModelVersion": "2"
  }
}

```

有关更多信息，请参阅 AWS IoT Events API 参考[UpdateDetectorModel](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateDetectorModel](#)中的。

## update-input

以下代码示例演示了如何使用 update-input。

### AWS CLI

#### 更新输入

以下update-input示例使用新的描述和定义更新了指定的输入。

```
aws iotevents update-input \  
  --cli-input-json file://pressureInput.json
```

pressureInput.json 的内容：

```
{  
  "inputName": "PressureInput",  
  "inputDescription": "Pressure readings from a motor",  
  "inputDefinition": {  
    "attributes": [  
      { "jsonPath": "sensorData.pressure" },  
      { "jsonPath": "motorid" }  
    ]  
  }  
}
```

输出：

```
{  
  "inputConfiguration": {  
    "status": "ACTIVE",  
    "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/PressureInput",  
    "lastUpdateTime": 1560795976.458,  
    "creationTime": 1560795312.542,  
    "inputName": "PressureInput",  
    "inputDescription": "Pressure readings from a motor"  
  }  
}
```

有关更多信息，请参阅 AWS IoT Events API 参考[UpdateInput](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateInput](#)中的。

## AWS IoT Events-Data 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS IoT Events-Data。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 [GitHub](#)，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## batch-put-message

以下代码示例演示了如何使用 `batch-put-message`。

AWS CLI

向 AWS IoT Events 发送消息 ( 输入 )

以下 `batch-put-message` 示例向 AWS IoT Events 系统发送了一组消息。每个消息有效载荷都将转换为您指定的输入 (`inputName`)，并输入到任何监视该输入的检测器中。如果发送了多条消息，则无法保证消息的处理顺序。为了保证顺序，您必须逐一发送一条消息，然后等待成功响应。

```
aws iotevents-data batch-put-message \  
  --cli-binary-format raw-in-base64-out \  
  --cli-input-json file://highPressureMessage.json
```

`highPressureMessage.json` 的内容：

```
{  
  "messages": [  
    {  
      "messageId": "00001",  
      "inputName": "PressureInput",  
      "payload": "{\"motorid\": \"Fulton-A32\", \"sensorData\": {\"pressure\":  
80, \"temperature\": 39} }"  
    }  
  ]  
}
```

输出：

```
{
```

```
"BatchPutMessageErrorEntries": []
}
```

有关更多信息，请参阅 [BatchPutMessage AWS IoT Events 开发者指南](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchPutMessage](#) 中的。

## batch-update-detector

以下代码示例演示了如何使用 batch-update-detector。

### AWS CLI

#### 更新探测器 (实例)

以下 batch-update-detector 示例更新了指定探测器模型的一个或多个探测器 (实例) 的状态、变量值和计时器设置。

```
aws iotevents-data batch-update-detector \  
  --cli-input-json file://budFulton-A32.json
```

budFulton-A32.json 的内容：

```
{  
  "detectors": [  
    {  
      "messageId": "00001",  
      "detectorModelName": "motorDetectorModel",  
      "keyValue": "Fulton-A32",  
      "state": {  
        "stateName": "Normal",  
        "variables": [  
          {  
            "name": "pressureThresholdBreach",  
            "value": "0"  
          }  
        ],  
        "timers": [  
        ]  
      }  
    }  
  ]  
}
```

```
}
```

输出：

```
{
  "batchUpdateDetectorErrorEntries": []
}
```

有关更多信息，请参阅 [BatchUpdateDetector AWS IoT Events 开发者指南\\*](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchUpdateDetector](#) 中的。

## create-detector-model

以下代码示例演示了如何使用 create-detector-model。

AWS CLI

创建探测器模型

以下 create-detector-model 示例创建探测器模型。

```
aws iotevents create-detector-model \
  --cli-input-json file://motorDetectorModel.json
```

motorDetectorModel.json 的内容：

```
{
  "detectorModelName": "motorDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "Normal",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "pressureThresholdBreach",
                    "value": "0"
                  }
                }
              ]
            }
          ]
        }
      }
    ]
  }
}
```



```

    }
  }
]
},
"onInput": {
  "transitionEvents": [
    {
      "eventName": "Overpressurized",
      "condition": "$input.PressureInput.sensorData.pressure
&gt; 70",
      "actions": [
        {
          "setVariable": {
            "variableName": "pressureThresholdBreach",
            "value":
"$variable.pressureThresholdBreach + 3"
          }
        }
      ],
      "nextState": "Dangerous"
    }
  ]
},
{
  "stateName": "Dangerous",
  "onEnter": {
    "events": [
      {
        "eventName": "Pressure Threshold Breached",
        "condition": "$variable.pressureThresholdBreach &gt;
1",
        "actions": [
          {
            "sns": {
              "targetArn": "arn:aws:sns:us-
east-1:123456789012:underPressureAction"
            }
          }
        ]
      }
    ]
  }
}
]

```

```

    },
    "onInput": {
      "events": [
        {
          "eventName": "Overpressurized",
          "condition": "$input.PressureInput.sensorData.pressure
&gt; 70",
          "actions": [
            {
              "setVariable": {
                "variableName": "pressureThresholdBreached",
                "value": "3"
              }
            }
          ]
        },
        {
          "eventName": "Pressure Okay",
          "condition": "$input.PressureInput.sensorData.pressure
&lt;= 70",
          "actions": [
            {
              "setVariable": {
                "variableName": "pressureThresholdBreached",
                "value":
"$variable.pressureThresholdBreached - 1"
              }
            }
          ]
        }
      ],
      "transitionEvents": [
        {
          "eventName": "BackToNormal",
          "condition": "$input.PressureInput.sensorData.pressure
&lt;= 70 &amp; $variable.pressureThresholdBreached &lt;= 1",
          "nextState": "Normal"
        }
      ]
    },
    "onExit": {
      "events": [
        {
          "eventName": "Normal Pressure Restored",

```

```

        "condition": "true",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-
east-1:123456789012:pressureClearedAction"
                }
            }
        ]
    }
]
},
"initialStateName": "Normal"
},
"key": "motorid",
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}

```

输出：

```

{
  "detectorModelConfiguration": {
    "status": "ACTIVATING",
    "lastUpdateTime": 1560796816.077,
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
    "creationTime": 1560796816.077,
    "detectorModelArn": "arn:aws:iotevents:us-west-2:123456789012:detectorModel/
motorDetectorModel",
    "key": "motorid",
    "detectorModelName": "motorDetectorModel",
    "detectorModelVersion": "1"
  }
}

```

有关更多信息，请参阅 [CreateDetectorModel AWS IoT Events 开发者指南](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDetectorModel](#) 中的。

## create-input

以下代码示例演示了如何使用 create-input。

## AWS CLI

### 创建输入

以下create-input示例创建了一个输入。

```
aws iotevents create-input \  
  --cli-input-json file://pressureInput.json
```

pressureInput.json 的内容：

```
{  
  "inputName": "PressureInput",  
  "inputDescription": "Pressure readings from a motor",  
  "inputDefinition": {  
    "attributes": [  
      { "jsonPath": "sensorData.pressure" },  
      { "jsonPath": "motorid" }  
    ]  
  }  
}
```

输出：

```
{  
  "inputConfiguration": {  
    "status": "ACTIVE",  
    "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/PressureInput",  
    "lastUpdateTime": 1560795312.542,  
    "creationTime": 1560795312.542,  
    "inputName": "PressureInput",  
    "inputDescription": "Pressure readings from a motor"  
  }  
}
```

有关更多信息，请参阅 [CreateInput AWS IoT Events 开发者指南\\*](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateInput](#)中的。

### delete-detector-model

以下代码示例演示了如何使用 delete-detector-model。

## AWS CLI

### 删除探测器模型

以下delete-detector-model示例删除探测器模型。探测器模型的所有活动实例也将被删除。

```
aws iotevents delete-detector-model \  
  --detector-model-name motorDetectorModel*
```

此命令不生成任何输出。

有关更多信息，请参阅 [DeleteDetectorModel AWS IoT Events 开发者指南](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDetectorModel](#)中的。

## delete-input

以下代码示例演示了如何使用 delete-input。

## AWS CLI

### 删除输入

以下delete-input示例删除了输入。

```
aws iotevents delete-input \  
  --input-name PressureInput
```

此命令不生成任何输出。

有关更多信息，请参阅 [DeleteInput AWS IoT Events 开发者指南](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInput](#)中的。

## describe-detector-model

以下代码示例演示了如何使用 describe-detector-model。

## AWS CLI

### 获取有关探测器模型的信息

以下describe-detector-model示例描述了探测器模型。如果未指定该version参数，则该命令将返回有关最新版本的信息。

```
aws iotevents describe-detector-model \  
  --detector-model-name motorDetectorModel
```

输出：

```
{  
  "detectorModel": {  
    "detectorModelConfiguration": {  
      "status": "ACTIVE",  
      "lastUpdateTime": 1560796816.077,  
      "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",  
      "creationTime": 1560796816.077,  
      "detectorModelArn": "arn:aws:iotevents:us-  
west-2:123456789012:detectorModel/motorDetectorModel",  
      "key": "motorid",  
      "detectorModelName": "motorDetectorModel",  
      "detectorModelVersion": "1"  
    },  
    "detectorModelDefinition": {  
      "states": [  
        {  
          "onInput": {  
            "transitionEvents": [  
              {  
                "eventName": "Overpressurized",  
                "actions": [  
                  {  
                    "setVariable": {  
                      "variableName":  
"pressureThresholdBreached",  
                      "value":  
"$variable.pressureThresholdBreached + 3"  
                    }  
                  }  
                ],  
                "condition":  
"$input.PressureInput.sensorData.pressure > 70",  
                "nextState": "Dangerous"  
              }  
            ],  
            "events": []  
          },  
          "stateName": "Normal",  
        }  
      ]  
    }  
  }  
}
```

```

        "onEnter": {
            "events": [
                {
                    "eventName": "init",
                    "actions": [
                        {
                            "setVariable": {
                                "variableName":
"pressureThresholdBreachd",
                                "value": "0"
                            }
                        }
                    ],
                    "condition": "true"
                }
            ]
        },
        "onExit": {
            "events": []
        }
    },
    {
        "onInput": {
            "transitionEvents": [
                {
                    "eventName": "BackToNormal",
                    "actions": [],
                    "condition":
"$input.PressureInput.sensorData.pressure <= 70 &&
$variable.pressureThresholdBreachd <= 1",
                    "nextState": "Normal"
                }
            ],
            "events": [
                {
                    "eventName": "Overpressurized",
                    "actions": [
                        {
                            "setVariable": {
                                "variableName":
"pressureThresholdBreachd",
                                "value": "3"
                            }
                        }
                    ]
                }
            ]
        }
    }
}

```

```

        ],
        "condition":
"$input.PressureInput.sensorData.pressure > 70"
    },
    {
        "eventName": "Pressure Okay",
        "actions": [
            {
                "setVariable": {
                    "variableName":
"pressureThresholdBreach",
                    "value":
"$variable.pressureThresholdBreach - 1"
                }
            }
        ],
        "condition":
"$input.PressureInput.sensorData.pressure <= 70"
    }
]
},
"stateName": "Dangerous",
"onEnter": {
    "events": [
        {
            "eventName": "Pressure Threshold Breached",
            "actions": [
                {
                    "sns": {
                        "targetArn": "arn:aws:sns:us-
east-1:123456789012:underPressureAction"
                    }
                }
            ]
        },
        {
            "condition": "$variable.pressureThresholdBreach >
1"
        }
    ]
},
"onExit": {
    "events": [
        {
            "eventName": "Normal Pressure Restored",
            "actions": [

```



```
        {
            "sns": {
                "targetArn": "arn:aws:sns:us-
east-1:123456789012:pressureClearedAction"
            }
        },
        "condition": "true"
    ]
}
    ]
}
    ],
    "initialStateName": "Normal"
}
}
```

有关更多信息，请参阅 [DescribeDetectorModel AWS IoT Events 开发者指南\\*](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDetectorModel](#) 中的。

## describe-detector

以下代码示例演示了如何使用 describe-detector。

### AWS CLI

获取有关探测器（实例）的信息

以下 describe-detector 示例返回有关指定探测器（实例）的信息。

```
aws iotevents-data describe-detector \
  --detector-model-name motorDetectorModel \
  --key-value "Fulton-A32"
```

输出：

```
{
  "detector": {
    "lastUpdateTime": 1560797852.776,
    "creationTime": 1560797852.775,
```

```
    "state": {
      "variables": [
        {
          "name": "pressureThresholdBreached",
          "value": "3"
        }
      ],
      "stateName": "Dangerous",
      "timers": []
    },
    "keyValue": "Fulton-A32",
    "detectorModelName": "motorDetectorModel",
    "detectorModelVersion": "1"
  }
}
```

有关更多信息，请参阅 [DescribeDetector AWS IoT Events 开发者指南](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDetector](#) 中的。

## describe-input

以下代码示例演示了如何使用 describe-input。

### AWS CLI

获取有关输入的信息

以下 describe-input 示例检索输入的详细信息。

```
aws iotevents describe-input \
  --input-name PressureInput
```

输出：

```
{
  "input": {
    "inputConfiguration": {
      "status": "ACTIVE",
      "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/
PressureInput",
      "lastUpdateTime": 1560795312.542,
```

```
        "creationTime": 1560795312.542,
        "inputName": "PressureInput",
        "inputDescription": "Pressure readings from a motor"
    },
    "inputDefinition": {
        "attributes": [
            {
                "jsonPath": "sensorData.pressure"
            },
            {
                "jsonPath": "motorid"
            }
        ]
    }
}
```

有关更多信息，请参阅 [DescribeInput AWS IoT Events 开发者指南](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeInput](#) 中的。

## describe-logging-options

以下代码示例演示了如何使用 describe-logging-options。

### AWS CLI

获取有关日志设置的信息

以下 describe-logging-options 示例检索当前的 AWS IoT Events 日志选项。

```
aws iotevents describe-logging-options
```

输出：

```
{
  "loggingOptions": {
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
    "enabled": false,
    "level": "ERROR"
  }
}
```

有关更多信息，请参阅 [DescribeLoggingOptions AWSIoT Events 开发者指南\\*](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoggingOptions](#)中的。

## list-detector-model-versions

以下代码示例演示了如何使用 list-detector-model-versions。

### AWS CLI

获取有关探测器模型版本的信息

以下list-detector-model-versions示例列出了探测器模型的所有版本。仅返回与每个探测器模型版本关联的元数据。

```
aws iotevents list-detector-model-versions \  
  --detector-model-name motorDetectorModel
```

输出：

```
{  
  "detectorModelVersionSummaries": [  
    {  
      "status": "ACTIVE",  
      "lastUpdateTime": 1560796816.077,  
      "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",  
      "creationTime": 1560796816.077,  
      "detectorModelArn": "arn:aws:iotevents:us-  
west-2:123456789012:detectorModel/motorDetectorModel",  
      "detectorModelName": "motorDetectorModel",  
      "detectorModelVersion": "1"  
    }  
  ]  
}
```

有关更多信息，请参阅 [ListDetectorModelVersions AWSIoT Events 开发者指南\\*](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDetectorModelVersions](#)中的。

## list-detector-models

以下代码示例演示了如何使用 list-detector-models。

## AWS CLI

### 获取您的探测器型号列表

以下 `list-detector-models` 示例列出了您创建的探测器模型。仅返回与每个检测器模型关联的元数据。

```
aws iotevents list-detector-models
```

输出：

```
{
  "detectorModelSummaries": [
    {
      "detectorModelName": "motorDetectorModel",
      "creationTime": 1552072424.212
      "detectorModelDescription": "Detect overpressure in a motor."
    }
  ]
}
```

有关更多信息，请参阅 [ListDetectorModels AWS IoT Events 开发者指南\\*](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDetectorModels](#) 中的。

## list-detectors

以下代码示例演示了如何使用 `list-detectors`。

### AWS CLI

#### 获取探测器型号的探测器列表

以下 `list-detectors` 示例列出了探测器（探测器模型的实例）。

```
aws iotevents-data list-detectors \
  --detector-model-name motorDetectorModel
```

输出：

```
{
  "detectorSummaries": [
```

```
{
  "lastUpdateTime": 1558129925.2,
  "creationTime": 1552073155.527,
  "state": {
    "stateName": "Normal"
  },
  "keyValue": "Fulton-A32",
  "detectorModelName": "motorDetectorModel",
  "detectorModelVersion": "1"
}
]
```

有关更多信息，请参阅 [ListDetectors AWS IoT Events 开发者指南](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDetectors](#) 中的。

## list-inputs

以下代码示例演示了如何使用 `list-inputs`。

### AWS CLI

#### 列出输入

以下 `list-inputs` 示例列出了您创建的输入。

```
aws iotevents list-inputs
```

输出：

```
{
  "status": "ACTIVE",
  "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/PressureInput",
  "lastUpdateTime": 1551742986.768,
  "creationTime": 1551742986.768,
  "inputName": "PressureInput",
  "inputDescription": "Pressure readings from a motor"
}
```

有关更多信息，请参阅 [ListInputs AWS IoT Events 开发者指南](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListInputs](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出分配给资源的标签

以下 `list-tags-for-resource` 示例列出了您为资源分配的标签（元数据）。

```
aws iotevents list-tags-for-resource \  
  --resource-arn "arn:aws:iotevents:us-west-2:123456789012:input/PressureInput"
```

输出：

```
{  
  "tags": [  
    {  
      "value": "motor",  
      "key": "deviceType"  
    }  
  ]  
}
```

有关更多信息，请参阅 [ListTagsForResource AWS IoT Events 开发者指南](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## put-logging-options

以下代码示例演示了如何使用 `put-logging-options`。

### AWS CLI

设置日志选项

以下 `list-tags-for-resource` 示例设置或更新 AWS IoT Events 日志选项。如果您更新任何 `loggingOptions` 字段的值，则最多需要一分钟，更改才能生效。此外，如果您更改附加到您 `roleArn` 字段中指定的角色的策略（例如，更正无效的策略），则该更改最多需要五分钟才能生效。

```
aws iotevents put-logging-options \  
  --resource-arn "arn:aws:iotevents:us-west-2:123456789012:input/PressureInput"
```

```
--cli-input-json file://logging-options.json
```

logging-options.json 的内容：

```
{
  "loggingOptions": {
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
    "level": "DEBUG",
    "enabled": true,
    "detectorDebugOptions": [
      {
        "detectorModelName": "motorDetectorModel",
        "keyValue": "Fulton-A32"
      }
    ]
  }
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [PutLoggingOptions AWS IoT Events 开发者指南](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutLoggingOptions](#) 中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源添加标签

以下 tag-resource 示例添加或修改给定资源的标签。标签是可用于管理资源的元数据。

```
aws iotevents tag-resource \  
  --cli-input-json file://pressureInput.tag.json
```

pressureInput.tag.json 的内容：

```
{
  "resourceArn": "arn:aws:iotevents:us-west-2:123456789012:input/PressureInput",
  "tags": [
    {
```



```
        "key": "deviceType",
        "value": "motor"
      }
    ]
  }
```

此命令不生成任何输出。

有关更多信息，请参阅 [TagResource AWSIoT Events 开发者指南](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

要从资源中删除标签

以下 untag-resource 示例从资源中移除指定的标签。

```
aws iotevents untag-resource \  
  --cli-input-json file:///pressureInput.untag.json
```

pressureInput.untag.json 的内容：

```
{  
  "resourceArn": "arn:aws:iotevents:us-west-2:123456789012:input/PressureInput",  
  "tagKeys": [  
    "deviceType"  
  ]  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [UntagResource AWSIoT Events 开发者指南](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-detector-model

以下代码示例演示了如何使用 update-detector-model。

## AWS CLI

### 更新探测器模型

以下 `update-detector-model` 示例更新探测器模型。先前版本生成的探测器（实例）会被删除，然后在新输入到来时重新创建。

```
aws iotevents update-detector-model \  
  --cli-input-json file://motorDetectorModel.update.json
```

`motorDetectorModel.update.json` 的内容：

```
{  
  "detectorModelName": "motorDetectorModel",  
  "detectorModelDefinition": {  
    "states": [  
      {  
        "stateName": "Normal",  
        "onEnter": {  
          "events": [  
            {  
              "eventName": "init",  
              "condition": "true",  
              "actions": [  
                {  
                  "setVariable": {  
                    "variableName": "pressureThresholdBreached",  
                    "value": "0"  
                  }  
                }  
              ]  
            }  
          ]  
        },  
        "onInput": {  
          "transitionEvents": [  
            {  
              "eventName": "Overpressurized",  
              "condition": "$input.PressureInput.sensorData.pressure > 70",  
              "actions": [  
                {  
                  "setVariable": {  
                    "variableName": "pressureThresholdBreached",
```

```
        "value": "$variable.pressureThresholdBreach + 3"
      }
    }
  ],
  "nextState": "Dangerous"
}
]
}
},
{
  "stateName": "Dangerous",
  "onEnter": {
    "events": [
      {
        "eventName": "Pressure Threshold Breached",
        "condition": "$variable.pressureThresholdBreach > 1",
        "actions": [
          {
            "sns": {
              "targetArn": "arn:aws:sns:us-
east-1:123456789012:underPressureAction"
            }
          }
        ]
      }
    ]
  },
  "onInput": {
    "events": [
      {
        "eventName": "Overpressurized",
        "condition": "$input.PressureInput.sensorData.pressure > 70",
        "actions": [
          {
            "setVariable": {
              "variableName": "pressureThresholdBreach",
              "value": "3"
            }
          }
        ]
      }
    ],
    {
      "eventName": "Pressure Okay",
      "condition": "$input.PressureInput.sensorData.pressure <= 70",
```

```
    "actions": [
      {
        "setVariable": {
          "variableName": "pressureThresholdBreached",
          "value": "$variable.pressureThresholdBreached - 1"
        }
      }
    ]
  },
  "transitionEvents": [
    {
      "eventName": "BackToNormal",
      "condition": "$input.PressureInput.sensorData.pressure <= 70 &&
$variable.pressureThresholdBreached <= 1",
      "nextState": "Normal"
    }
  ],
  "onExit": {
    "events": [
      {
        "eventName": "Normal Pressure Restored",
        "condition": "true",
        "actions": [
          {
            "sns": {
              "targetArn": "arn:aws:sns:us-
east-1:123456789012:pressureClearedAction"
            }
          }
        ]
      }
    ]
  }
},
"initialStateName": "Normal"
},
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}
```

输出：

```
{
  "detectorModelConfiguration": {
    "status": "ACTIVATING",
    "lastUpdateTime": 1560799387.719,
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
    "creationTime": 1560799387.719,
    "detectorModelArn": "arn:aws:iotevents:us-west-2:123456789012:detectorModel/
motorDetectorModel",
    "key": "motorid",
    "detectorModelName": "motorDetectorModel",
    "detectorModelVersion": "2"
  }
}
```

有关更多信息，请参阅 [UpdateDetectorModel AWS IoT Events 开发者指南\\*](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateDetectorModel](#) 中的。

## update-input

以下代码示例演示了如何使用 update-input。

### AWS CLI

#### 更新输入

以下 update-input 示例更新输入。

```
aws iotevents update-input \
  --cli-input-json file://pressureInput.json
```

pressureInput.json 的内容：

```
{
  "inputName": "PressureInput",
  "inputDescription": "Pressure readings from a motor",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "sensorData.pressure" },
      { "jsonPath": "motorid" }
    ]
  }
}
```

```
}
```

输出：

```
{
  "inputConfiguration": {
    "status": "ACTIVE",
    "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/PressureInput",
    "lastUpdateTime": 1560795976.458,
    "creationTime": 1560795312.542,
    "inputName": "PressureInput",
    "inputDescription": "Pressure readings from a motor"
  }
}
```

有关更多信息，请参阅 [UpdateInput AWS IoT Events 开发者指南](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateInput](#) 中的。

## AWS IoT Greengrass 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS IoT Greengrass。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-role-to-group**

以下代码示例演示了如何使用 `associate-role-to-group`。

## AWS CLI

### 将角色与 Greengrass 群组关联

以下 `associate-role-to-group` 示例将指定的 IAM 角色与 Greengrass 群组相关联。本地 Lambda 函数和连接器使用组角色来访问服务 AWS。例如，您的群组角色可能会授予 CloudWatch 日志集成所需的权限。

```
aws greengrass associate-role-to-group \  
  --group-id 2494ee3f-7f8a-4e92-a78b-d205f808b84b \  
  --role-arn arn:aws:iam::123456789012:role/GG-Group-Role
```

输出：

```
{  
  "AssociatedAt": "2019-09-10T20:03:30Z"  
}
```

有关更多信息，请参阅 AWS IoT Greengrass 开发者指南 [中的配置群组角色](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateRoleToGroup](#) 中的。

## **associate-service-role-to-account**

以下代码示例演示了如何使用 `associate-service-role-to-account`。

## AWS CLI

### 将服务角色与您的 AWS 账户关联

以下 `associate-service-role-to-account` 示例将由其 ARN 指定的 IAM 服务角色与您账户中的 IoT AWS Greengrass 关联起来。AWS 您之前必须已在 IAM 中创建过服务角色，并且必须将允许 IoT AWS Greengrass 担任此角色的策略文档与其相关联。

```
aws greengrass associate-service-role-to-account \  
  --role-arn "arn:aws:iam::123456789012:role/service-role/Greengrass_ServiceRole"
```

输出：

```
{
```

```
"AssociatedAt": "2019-06-25T18:12:45Z"
}
```

有关更多信息，请参阅《物联网 [AWS Greengrass 开发者指南](#)》中的 [Greengrass 服务角色](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateServiceRoleToAccount](#) 中的。

## create-connector-definition-version

以下代码示例演示了如何使用 create-connector-definition-version。

### AWS CLI

#### 创建连接器定义版本

以下 create-connector-definition-version 示例创建连接器定义版本并将其与指定的连接器定义相关联。版本中的所有连接器都为其参数定义值。

```
aws greengrass create-connector-definition-version \
  --connector-definition-id "55d0052b-0d7d-44d6-b56f-21867215e118" \
  --connectors "[{"Id": "MyTwilioNotificationsConnector",
  "ConnectorArn": "arn:aws:greengrass:us-west-2::/connectors/
  TwilioNotifications/versions/2", "Parameters": {"TWILIO_ACCOUNT_SID
  ": "AC1a8d4204890840d7fc482aab38090d57", "TwilioAuthTokenSecretArn":
  "arn:aws:secretsmanager:us-west-2:123456789012:secret:greengrass-TwilioAuthToken-
  ntSlp6", "TwilioAuthTokenSecretArn-ResourceId": "TwilioAuthToken",
  "DefaultFromPhoneNumber": "4254492999"}]]"
```

输出：

```
{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/
  connectors/55d0052b-0d7d-44d6-b56f-21867215e118/versions/33f709a0-c825-49cb-9eea-
  dc8964fbd635",
  "CreationTimestamp": "2019-06-24T20:46:30.134Z",
  "Id": "55d0052b-0d7d-44d6-b56f-21867215e118",
  "Version": "33f709a0-c825-49cb-9eea-dc8964fbd635"
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateConnectorDefinitionVersion](#) 中的。



## create-connector-definition

以下代码示例演示了如何使用 `create-connector-definition`。

### AWS CLI

#### 创建连接器定义

以下 `create-connector-definition` 示例创建连接器定义和初始连接器定义版本。初始版本包含一个连接器。版本中的所有连接器都为其参数定义值。

```
aws greengrass create-connector-definition \  
  --name MySNSConnector \  
  --initial-version '{"Connectors": [{"Id": "MySNSConnector", "ConnectorArn": "arn:aws:greengrass:us-west-2:/connectors/SNS/versions/1", "Parameters": {"DefaultSNSArn": "arn:aws:sns:us-west-2:123456789012:GGConnectorTopic"}}]}'
```

输出：

```
{  
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/connectors/b5c4ebfd-f672-49a3-83cd-31c7216a7bb8",  
  "CreationTimestamp": "2019-06-19T19:30:01.300Z",  
  "Id": "b5c4ebfd-f672-49a3-83cd-31c7216a7bb8",  
  "LastUpdatedTimestamp": "2019-06-19T19:30:01.300Z",  
  "LatestVersion": "63c57963-c7c2-4a26-a7e2-7bf478ea2623",  
  "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/connectors/b5c4ebfd-f672-49a3-83cd-31c7216a7bb8/versions/63c57963-c7c2-4a26-a7e2-7bf478ea2623",  
  "Name": "MySNSConnector"  
}
```

有关更多信息，请参阅《AWS 物联网 Green [grass 开发者指南](#)》中的 [Greengrass 连接器入门 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateConnectorDefinition](#) 中的。

## create-core-definition-version

以下代码示例演示了如何使用 `create-core-definition-version`。

## AWS CLI

### 创建核心定义版本

以下 `create-core-definition-version` 示例创建核心定义版本并将其与指定的核心定义相关联。该版本只能包含一个内核。在创建核心之前，必须先创建和配置相应的 AWS IoT 事物。此过程包括以下 `iot` 命令，这些命令返回 `create-core-definition-version` 命令 `CertificateArn` 所需的 `ThingArn` 和。

创建与核心设备对应的 AWS 物联网事物：

```
aws iot create-thing \  
  --thing-name "MyCoreDevice"
```

输出：

```
{  
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyCoreDevice",  
  "thingName": "MyCoreDevice",  
  "thingId": "cb419a19-9099-4515-9cec-e9b0e760608a"  
}
```

为事物创建公钥和私钥以及核心设备证书。此示例使用 `create-keys-and-certificate` 命令并需要对当前目录的写入权限。或者，您可以使用 `create-certificate-from-csr` 命令。

```
aws iot create-keys-and-certificate \  
  --set-as-active \  
  --certificate-pem-outfile "myCore.cert.pem" \  
  --public-key-outfile "myCore.public.key" \  
  --private-key-outfile "myCore.private.key"
```

输出：

```
{  
  "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/123a15ec415668c2349a76170b64ac0878231c1e21ec83c10e92a1EXAMPLExyz",  
  "certificatePem": "-----BEGIN CERTIFICATE-----  
  \nMIIDWTCAkGgAwIBATgIUCGq6EGqou6zFqWgIZRndgQEFW+gwDQYJKoZIhvc...KdGewQS\n-----END  
  CERTIFICATE-----\n",  
  "keyPair": {
```

```

    "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBzrqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAqKpRgnn6yq26U3y...wIDAQAB\n-----END
PUBLIC KEY-----\n",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIABAKCAQEAqKpRgnn6yq26U3yt5YFZquyukfRjBMXDcNOK4rMCxDR...fvY4+te\n-----END
RSA PRIVATE KEY-----\n"
  },
  "certificateId":
  "123a15ec415668c2349a76170b64ac0878231c1e21ec83c10e92a1EXAMPLExyz"
}

```

创建允许iot和greengrass操作的 AWS IoT 策略。为简单起见，以下策略允许对所有资源执行操作，但您的策略应更具限制性。

```

aws iot create-policy \
  --policy-name "Core_Devices" \
  --policy-document "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect
\n:\":\"Allow\", \"Action\":[\"iot:Publish\", \"iot:Subscribe\", \"iot:Connect
\n\", \"iot:Receive\"], \"Resource\":[\"*\"]}, {\"Effect\":[\"Allow\", \"Action\":[
\n\"iot:GetThingShadow\", \"iot:UpdateThingShadow\", \"iot>DeleteThingShadow\"],
\n\"Resource\":[\"*\"]}, {\"Effect\":[\"Allow\", \"Action\":[\"greengrass:*\"], \"Resource
\n\":[\"*\"]}]}"

```

输出：

```

{
  "policyName": "Core_Devices",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/Core_Devices",
  "policyDocument": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect
\n:\":\"Allow\", \"Action\":[\"iot:Publish\", \"iot:Subscribe\", \"iot:Connect
\n\", \"iot:Receive\"], \"Resource\":[\"*\"]}, {\"Effect\":[\"Allow\", \"Action\":[
\n\"iot:GetThingShadow\", \"iot:UpdateThingShadow\", \"iot>DeleteThingShadow\"],
\n\"Resource\":[\"*\"]}, {\"Effect\":[\"Allow\", \"Action\":[\"greengrass:*\"], \"Resource
\n\":[\"*\"]}]}",
  "policyVersionId": "1"
}

```

将策略附加到证书：

```

aws iot attach-policy \
  --policy-name "Core_Devices" \

```

```
--target "arn:aws:iot:us-west-2:123456789012:cert/123a15ec415668c2349a76170b64ac0878231c1e21ec83c10e92a1EXAMPLExyz"
```

此命令不生成任何输出。

把东西附加到证书上：

```
aws iot attach-thing-principal \  
  --thing-name "MyCoreDevice" \  
  --principal "arn:aws:iot:us-west-2:123456789012:cert/123a15ec415668c2349a76170b64ac0878231c1e21ec83c10e92a1EXAMPLExyz"
```

此命令不生成任何输出。

创建核心定义版本：

```
aws greengrass create-core-definition-version \  
  --core-definition-id "582efe12-b05a-409e-9a24-a2ba1bcc4a12" \  
  --cores "[{\\"Id\\":\\"MyCoreDevice\\",\\"ThingArn\\":\\"arn:aws:iot:us-west-2:123456789012:thing/MyCoreDevice\\",\\"CertificateArn\\":\\"arn:aws:iot:us-west-2:123456789012:cert/123a15ec415668c2349a76170b64ac0878231c1e21ec83c10e92a1EXAMPLExyz\\",\\"SyncShadow\\":true}]"
```

输出：

```
{  
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/cores/582efe12-b05a-409e-9a24-a2ba1bcc4a12/versions/3fdc1190-2ce5-44de-b98b-eec8f9571014",  
  "Version": "3fdc1190-2ce5-44de-b98b-eec8f9571014",  
  "CreationTimestamp": "2019-09-18T00:15:09.838Z",  
  "Id": "582efe12-b05a-409e-9a24-a2ba1bcc4a12"  
}
```

有关更多信息，请参阅 [《AWS 物联网 Greengrass 开发者指南》](#) 中的配置物联网 Greengrass 核心AWS。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCoreDefinitionVersion](#)中的。

## create-core-definition

以下代码示例演示了如何使用 create-core-definition。

## AWS CLI

### 示例 1：创建空核心定义

以下`create-core-definition`示例创建了一个空的（没有初始版本）Greengrass 核心定义。在内核可用之前，必须使用`create-core-definition-version`命令为内核提供其他参数。

```
aws greengrass create-core-definition \  
  --name cliGroup_Core
```

输出：

```
{  
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/cores/  
b5c08008-54cb-44bd-9eec-c121b04283b5",  
  "CreationTimestamp": "2019-06-25T18:23:22.106Z",  
  "Id": "b5c08008-54cb-44bd-9eec-c121b04283b5",  
  "LastUpdatedTimestamp": "2019-06-25T18:23:22.106Z",  
  "Name": "cliGroup_Core"  
}
```

### 示例 2：使用初始版本创建核心定义

以下`create-core-definition`示例创建了一个包含初始核心定义版本的核心定义。该版本只能包含一个内核。在创建核心之前，必须先创建和配置相应的 AWS IoT 事物。此过程包括以下`iot`命令，这些命令返回`create-core-definition`命令`CertificateArn`所需的`ThingArn`和。

创建与核心设备对应的 AWS 物联网事物：

```
aws iot create-thing \  
  --thing-name "MyCoreDevice"
```

输出：

```
{  
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyCoreDevice",  
  "thingName": "MyCoreDevice",  
  "thingId": "cb419a19-9099-4515-9cec-e9b0e760608a"  
}
```

为事物创建公钥和私钥以及核心设备证书。此示例使用`create-keys-and-certificate`命令并需要对当前目录的写入权限。或者，您可以使用 `create-certificate-from-csr` 命令。

```
aws iot create-keys-and-certificate \
  --set-as-active \
  --certificate-pem-outfile "myCore.cert.pem" \
  --public-key-outfile "myCore.public.key" \
  --private-key-outfile "myCore.private.key"
```

输出：

```
{
  "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/123a15ec415668c2349a76170b64ac0878231c1e21ec83c10e92a1EXAMPLExyz",
  "certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCakGgAwIBATgIUCGq6EGqou6zFqWgIZRndgQEFW+gwDQYJKoZIhvc...KdGewQS\n-----END
CERTIFICATE-----\n",
  "keyPair": {
    "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBzrqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAqKpRgnn6yq26U3y...wIDAQAB\n-----END
PUBLIC KEY-----\n",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIABAKCAQEAqKpRgnn6yq26U3yt5YFZquyukfRjBMXDcNOK4rMCxDR...fvY4+te\n-----END
RSA PRIVATE KEY-----\n"
  },
  "certificateId":
  "123a15ec415668c2349a76170b64ac0878231c1e21ec83c10e92a1EXAMPLExyz"
}
```

创建允许iot和greengrass操作的 AWS IoT 策略。为简单起见，以下策略允许对所有资源执行操作，但您的策略应更具限制性。

```
aws iot create-policy \
  --policy-name "Core_Devices" \
  --policy-document "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect
\": \"Allow\", \"Action\": [\"iot:Publish\", \"iot:Subscribe\", \"iot:Connect
\", \"iot:Receive\"], \"Resource\": [\"*\"]}, {\"Effect\": \"Allow\", \"Action\":
[\"iot:GetThingShadow\", \"iot:UpdateThingShadow\", \"iot:DeleteThingShadow\"],
\"Resource\": [\"*\"]}, {\"Effect\": \"Allow\", \"Action\": [\"greengrass:*\"], \"Resource
\": [\"*\"]}]}"
```

输出：

```
{
```

```

    "policyName": "Core_Devices",
    "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/Core_Devices",
    "policyDocument": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",\"Action\":[\"iot:Publish\",\"iot:Subscribe\",\"iot:Connect\",\"iot:Receive\"],\"Resource\":[\"*\"]},{\"Effect\":\"Allow\",\"Action\":[\"iot:GetThingShadow\",\"iot:UpdateThingShadow\",\"iot:DeleteThingShadow\"],\"Resource\":[\"*\"]},{\"Effect\":\"Allow\",\"Action\":[\"greengrass:*\"],\"Resource\":[\"*\"]}]}",
    "policyVersionId": "1"
  }

```

将策略附加到证书：

```

aws iot attach-policy \
  --policy-name "Core_Devices" \
  --target "arn:aws:iot:us-
west-2:123456789012:cert/123a15ec415668c2349a76170b64ac0878231c1e21ec83c10e92a1EXAMPLExyz"

```

此命令不生成任何输出。

把东西附加到证书上：

```

aws iot attach-thing-principal \
  --thing-name "MyCoreDevice" \
  --principal "arn:aws:iot:us-
west-2:123456789012:cert/123a15ec415668c2349a76170b64ac0878231c1e21ec83c10e92a1EXAMPLExyz"

```

此命令不生成任何输出。

创建核心定义：

```

aws greengrass create-core-definition \
  --name "MyCores" \
  --initial-version "{\"Cores\":[{\"Id\":\"MyCoreDevice\",\"ThingArn\":
\"arn:aws:iot:us-west-2:123456789012:thing/MyCoreDevice\",\"CertificateArn\":
\"arn:aws:iot:us-
west-2:123456789012:cert/123a15ec415668c2349a76170b64ac0878231c1e21ec83c10e92a1EXAMPLExyz
\",\"SyncShadow\":true}]}"

```

输出：

```
{
```

```
"LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/cores/582efe12-b05a-409e-9a24-a2ba1bcc4a12/versions/cc87b5b3-8f4b-465d-944c-1d6de5dbfcdb",
  "Name": "MyCores",
  "LastUpdatedTimestamp": "2019-09-18T00:11:06.197Z",
  "LatestVersion": "cc87b5b3-8f4b-465d-944c-1d6de5dbfcdb",
  "CreationTimestamp": "2019-09-18T00:11:06.197Z",
  "Id": "582efe12-b05a-409e-9a24-a2ba1bcc4a12",
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/cores/582efe12-b05a-409e-9a24-a2ba1bcc4a12"
}
```

有关更多信息，请参阅 [《AWS 物联网 Greengrass 开发者指南》中的配置物联网 Greengrass 核AWS 心。](#)

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCoreDefinition](#)中的。

## create-deployment

以下代码示例演示了如何使用 create-deployment。

### AWS CLI

为 Greengrass 群组的某个版本创建部署

以下create-deployment示例部署了指定版本的 Greengrass 群组。

```
aws greengrass create-deployment \
  --deployment-type NewDeployment \
  --group-id "ce2e7d01-3240-4c24-b8e6-f6f6e7a9eeca" \
  --group-version-id "dc40c1e9-e8c8-4d28-a84d-a9cad5f599c9"
```

输出：

```
{
  "DeploymentArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/groups/ce2e7d01-3240-4c24-b8e6-f6f6e7a9eeca/deployments/bfceb608-4e97-45bc-af5c-460144270308",
  "DeploymentId": "bfceb608-4e97-45bc-af5c-460144270308"
}
```

有关更多信息，请参阅 [AWS IoT Greengrass 开发者指南中的连接器入门 \(CLI\)](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDeployment](#)中的。

## create-device-definition-version

以下代码示例演示了如何使用 create-device-definition-version。

### AWS CLI

#### 创建设备定义版本

以下create-device-definition-version示例创建设备定义版本并将其与指定的设备定义相关联。该版本定义了两个设备。在创建 Greengrass 设备之前，必须先创建和配置相应的物联网设备。AWS 此过程包括以下iot命令，您必须运行这些命令才能获取 Greengrass 命令所需的信息：

创建与设备对应的 AWS 物联网事物：

```
aws iot create-thing \  
  --thing-name "InteriorTherm"
```

输出：

```
{  
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/InteriorTherm",  
  "thingName": "InteriorTherm",  
  "thingId": "01d4763c-78a6-46c6-92be-7add080394bf"  
}
```

为事物创建公钥和私钥以及设备证书。此示例使用create-keys-and-certificate命令并需要对当前目录的写入权限。或者，你可以使用以下create-certificate-from-csr命令：

```
aws iot create-keys-and-certificate \  
  --set-as-active \  
  --certificate-pem-outfile "myDevice.cert.pem" \  
  --public-key-outfile "myDevice.public.key" \  
  --private-key-outfile "myDevice.private.key"
```

输出：

```
{  
  "certificateArn": "arn:aws:iot:us-  
west-2:123456789012:cert/66a415ec415668c2349a76170b64ac0878231c1e21ec83c10e92a18bd568eb92",
```

```

    "certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCaKGGAwIBATgIUCGq6EGqou6zFqWgIZRndgQEFW+gwDQYJKoZIhvc...KdGewQS\n-----END
CERTIFICATE-----\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBzrqhkiG9w0BAQEFAA0CAQ8AMIIBCgKCAQEAqKpRgnn6yq26U3y...wIDAQAB\n-----END
PUBLIC KEY-----\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIIEowIABAKCAQEAqKpRgnn6yq26U3yt5YFZquyukfRjBMXDcNOK4rMCxDR...fvY4+te\n-----END
RSA PRIVATE KEY-----\n"
    },
    "certificateId":
    "66a415ec415668c2349a76170b64ac0878231c1e21ec83c10e92a18bd568eb92"
}

```

创建允许*iot*和*greengrass*操作的 AWS IoT 策略。为简单起见，以下策略允许对所有资源执行操作，但您的策略可以更严格：

```

aws iot create-policy \
  --policy-name "GG_Devices" \
  --policy-document "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":"
  "\"Allow\",\"Action\":[\"iot:Publish\",\"iot:Subscribe\",\"iot:Connect
  \",\"iot:Receive\"],\"Resource\":[\"*\"]},{\"Effect\":\"Allow\",\"Action\":"
  "[\"iot:GetThingShadow\",\"iot:UpdateThingShadow\",\"iot>DeleteThingShadow\"],
  \"Resource\":[\"*\"]},{\"Effect\":\"Allow\",\"Action\":[\"greengrass:*\"],\"Resource
  \":[\"*\"]}]}"

```

输出：

```

{
  "policyName": "GG_Devices",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/GG_Devices",
  "policyDocument": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":"
  "\"Allow\",\"Action\":[\"iot:Publish\",\"iot:Subscribe\",\"iot:Connect
  \",\"iot:Receive\"],\"Resource\":[\"*\"]},{\"Effect\":\"Allow\",\"Action\":"
  "[\"iot:GetThingShadow\",\"iot:UpdateThingShadow\",\"iot>DeleteThingShadow\"],
  \"Resource\":[\"*\"]},{\"Effect\":\"Allow\",\"Action\":[\"greengrass:*\"],\"Resource
  \":[\"*\"]}]}",
  "policyVersionId": "1"
}

```

将策略附加到证书：

```
aws iot attach-policy \
  --policy-name "GG_Devices" \
  --target "arn:aws:iot:us-
west-2:123456789012:cert/66a415ec415668c2349a76170b64ac0878231c1e21ec83c10e92a18bd568eb92"
```

把东西附加到证书上

```
aws iot attach-thing-principal \
  --thing-name "InteriorTherm" \
  --principal "arn:aws:iot:us-
west-2:123456789012:cert/66a415ec415668c2349a76170b64ac0878231c1e21ec83c10e92a18bd568eb92"
```

如上所示创建和配置物联网事物后，使用以下示例CertificateArn中前两个命令中的ThingArn和。

```
aws greengrass create-device-definition-version \
  --device-definition-id "f9ba083d-5ad4-4534-9f86-026a45df1ccd" \
  --devices "[{\\"Id\\":\\"InteriorTherm\\",\\"ThingArn\\":\\"arn:aws:iot:us-
west-2:123456789012:thing/InteriorTherm\\",\\"CertificateArn\\":\\"arn:aws:iot:us-
west-2:123456789012:cert/66a415ec415668c2349a76170b64ac0878231c1e21ec83c10e92a18bd568eb92\\",
\\"SyncShadow\\":true},{\\"Id\\":\\"ExteriorTherm\\",\\"ThingArn\\":\\"arn:aws:iot:us-
west-2:123456789012:thing/ExteriorTherm\\",\\"CertificateArn\\":\\"arn:aws:iot:us-
west-2:123456789012:cert/6c52ce1b47bde88a637e9ccdd45fe4e4c2c0a75a6866f8f63d980ee22fa51e02\\",
\\"SyncShadow\\":true}]"
```

输出：

```
{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/devices/f9ba083d-5ad4-4534-9f86-026a45df1ccd/
versions/83c13984-6fed-447e-84d5-5b8aa45d5f71",
  "Version": "83c13984-6fed-447e-84d5-5b8aa45d5f71",
  "CreationTimestamp": "2019-09-11T00:15:09.838Z",
  "Id": "f9ba083d-5ad4-4534-9f86-026a45df1ccd"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDeviceDefinitionVersion](#)中的。

## create-device-definition

以下代码示例演示了如何使用 create-device-definition。

## AWS CLI

### 创建设备定义

以下create-device-definition示例创建了一个包含初始设备定义版本的设备定义。初始版本定义了两个设备。在创建 Greengrass 设备之前，必须先创建和配置相应的物联网设备。AWS 此过程包括以下iot命令，您必须运行这些命令才能获取 Greengrass 命令所需的信息：

创建与设备对应的 AWS 物联网事物：

```
aws iot create-thing \  
  --thing-name "InteriorTherm"
```

输出：

```
{  
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/InteriorTherm",  
  "thingName": "InteriorTherm",  
  "thingId": "01d4763c-78a6-46c6-92be-7add080394bf"  
}
```

为事物创建公钥和私钥以及设备证书。此示例使用create-keys-and-certificate命令并需要对当前目录的写入权限。或者，你可以使用以下create-certificate-from-csr命令：

```
aws iot create-keys-and-certificate \  
  --set-as-active \  
  --certificate-pem-outfile "myDevice.cert.pem" \  
  --public-key-outfile "myDevice.public.key" \  
  --private-key-outfile "myDevice.private.key"
```

输出：

```
{  
  "certificateArn": "arn:aws:iot:us-  
west-2:123456789012:cert/66a415ec415668c2349a76170b64ac0878231c1e21ec83c10e92a18bd568eb92",  
  "certificatePem": "-----BEGIN CERTIFICATE-----  
\nMIIDWTCAkGgAwIBATgIUcGq6EGqou6zFqWgIZRndgQEFW+gwDQYJKoZIhvc...KdGewQS\n-----END  
CERTIFICATE-----\n",  
  "keyPair": {  
    "PublicKey": "-----BEGIN PUBLIC KEY-----  
\nMIIBIjANBzrqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEaqKpRgnn6yq26U3y...wIDAQAB\n-----END  
PUBLIC KEY-----\n",
```

```

    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIIEowIABAKCAQEAqKpRgnn6yq26U3yt5YFZquyukfRjBMXDcNOK4rMCxDR...fvY4+te\n-----END
RSA PRIVATE KEY-----\n"
  },
  "certificateId":
    "66a415ec415668c2349a76170b64ac0878231c1e21ec83c10e92a18bd568eb92"
}

```

创建允许iot和greengrass操作的 AWS IoT 策略。为简单起见，以下策略允许对所有资源执行操作，但您的策略可以更严格：

```

aws iot create-policy \
  --policy-name "GG_Devices" \
  --policy-document "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":"
  \"Allow\",\"Action\":[\"iot:Publish\",\"iot:Subscribe\",\"iot:Connect
  \",\"iot:Receive\"],\"Resource\":[\"*\"]},{\"Effect\":\"Allow\",\"Action\":"
  [\"iot:GetThingShadow\",\"iot:UpdateThingShadow\",\"iot:DeleteThingShadow\"],
  \"Resource\":[\"*\"]},{\"Effect\":\"Allow\",\"Action\":[\"greengrass:*\"],\"Resource
  \":[\"*\"]}]}\"

```

输出：

```

{
  "policyName": "GG_Devices",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/GG_Devices",
  "policyDocument": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":"
  \"Allow\",\"Action\":[\"iot:Publish\",\"iot:Subscribe\",\"iot:Connect
  \",\"iot:Receive\"],\"Resource\":[\"*\"]},{\"Effect\":\"Allow\",\"Action\":"
  [\"iot:GetThingShadow\",\"iot:UpdateThingShadow\",\"iot:DeleteThingShadow\"],
  \"Resource\":[\"*\"]},{\"Effect\":\"Allow\",\"Action\":[\"greengrass:*\"],\"Resource
  \":[\"*\"]}]}\",
  "policyVersionId": "1"
}

```

将策略附加到证书：

```

aws iot attach-policy \
  --policy-name "GG_Devices" \
  --target "arn:aws:iot:us-
west-2:123456789012:cert/66a415ec415668c2349a76170b64ac0878231c1e21ec83c10e92a18bd568eb92"

```

把东西附加到证书上

```
aws iot attach-thing-principal \
  --thing-name "InteriorTherm" \
  --principal "arn:aws:iot:us-
west-2:123456789012:cert/66a415ec415668c2349a76170b64ac0878231c1e21ec83c10e92a18bd568eb92"
```

如上所示创建和配置物联网事物后，使用以下示例CertificateArn中前两个命令中的ThingArn和。

```
aws greengrass create-device-definition \
  --name "Sensors" \
  --initial-version "{ \"Devices\": [{ \"Id\": \"InteriorTherm
\", \"ThingArn\": \"arn:aws:iot:us-west-2:123456789012:thing/
InteriorTherm\", \"CertificateArn\": \"arn:aws:iot:us-
west-2:123456789012:cert/66a415ec415668c2349a76170b64ac0878231c1e21ec83c10e92a18bd568eb92\" },
{ \"Id\": \"ExteriorTherm\", \"ThingArn\": \"arn:aws:iot:us-
west-2:123456789012:thing/ExteriorTherm\", \"CertificateArn\": \"arn:aws:iot:us-
west-2:123456789012:cert/6c52ce1b47bde88a637e9ccdd45fe4e4c2c0a75a6866f8f63d980ee22fa51e02\" },
{ \"Id\": \"SyncShadow\", \"ThingArn\": \"arn:aws:iot:us-west-2:123456789012:thing/SyncShadow\", \"CertificateArn\": \"arn:aws:iot:us-
west-2:123456789012:cert/6c52ce1b47bde88a637e9ccdd45fe4e4c2c0a75a6866f8f63d980ee22fa51e02\" },
{ \"Id\": \"SyncShadow\", \"ThingArn\": \"arn:aws:iot:us-west-2:123456789012:thing/SyncShadow\", \"CertificateArn\": \"arn:aws:iot:us-
west-2:123456789012:cert/6c52ce1b47bde88a637e9ccdd45fe4e4c2c0a75a6866f8f63d980ee22fa51e02\" } ] } }
```

输出：

```
{
  "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/devices/f9ba083d-5ad4-4534-9f86-026a45df1ccd/
versions/3b5cc510-58c1-44b5-9d98-4ad858ffa795",
  "Name": "Sensors",
  "LastUpdatedTimestamp": "2019-09-11T00:11:06.197Z",
  "LatestVersion": "3b5cc510-58c1-44b5-9d98-4ad858ffa795",
  "CreationTimestamp": "2019-09-11T00:11:06.197Z",
  "Id": "f9ba083d-5ad4-4534-9f86-026a45df1ccd",
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/
devices/f9ba083d-5ad4-4534-9f86-026a45df1ccd"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDeviceDefinition](#)中的。

## create-function-definition-version

以下代码示例演示了如何使用 create-function-definition-version。

## AWS CLI

### 创建函数定义的版本

以下`create-function-definition-version`示例创建了指定函数定义的新版本。此版本指定了 ID 为`Hello-World-function`、允许访问文件系统的单个函数，并指定了最大内存大小和超时时间。

```
aws greengrass create-function-definition-version \
  --cli-input-json "{\"FunctionDefinitionId\": \"e626e8c9-3b8f-4bf3-9cdc-
d26ecdeb9fa3\", \"Functions\": [{\"Id\": \"Hello-World-function\", \"FunctionArn\":
  \"arn:aws:lambda:us-
west-2:123456789012:function:Greengrass_HelloWorld_Counter:gghw-alias\"},
  \"FunctionConfiguration\": {\"Environment\": {\"AccessSysfs\": true}, \"Executable\":
  \"greengrassHelloWorldCounter.function_handler\", \"MemorySize\": 16000, \"Pinned\":
  false, \"Timeout\": 25}}]\""
```

输出：

```
{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/functions/e626e8c9-3b8f-4bf3-9cdc-d26ecdeb9fa3/
versions/74abd1cc-637e-4abe-8684-9a67890f4043",
  "CreationTimestamp": "2019-06-25T22:03:43.376Z",
  "Id": "e626e8c9-3b8f-4bf3-9cdc-d26ecdeb9fa3",
  "Version": "74abd1cc-637e-4abe-8684-9a67890f4043"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFunctionDefinitionVersion](#)中的。

## **create-function-definition**

以下代码示例演示了如何使用 `create-function-definition`。

## AWS CLI

### 创建 Lambda 函数定义

以下`create-function-definition`示例通过提供 Lambda 函数列表（在本例中为仅包含一个名为的函数的列表`TempMonitorFunction`）及其配置来创建 Lambda 函数定义和初始版本。在

创建函数定义之前，您需要 Lambda 函数 ARN。要创建函数及其别名，请使用 `aws greengrass create-function` 和 `publish-version` 命令。Lambda 的 `create-function` 命令需要执行角色的 ARN，尽管 AWS 物联网 Greengrass 不使用该角色，因为权限是在 Greengrass 组角色中指定的。您可以使用 `IAM create-role` 命令创建一个空角色以获取 ARN 与 Lambda 一起使用，`create-function` 也可以使用现有的执行角色。

```
aws greengrass create-function-definition \  
  --name MyGreengrassFunctions \  
  --initial-version '{"Functions": [{"Id": "TempMonitorFunction",  
  "FunctionArn": "arn:aws:lambda:us-  
west-2:123456789012:function:TempMonitor:GG_TempMonitor", "FunctionConfiguration  
": {"Executable": "temp_monitor.function_handler", "MemorySize": 16000,  
"Timeout": 5}}]}'
```

输出：

```
{  
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/  
functions/3b0d0080-87e7-48c6-b182-503ec743a08b",  
  "CreationTimestamp": "2019-06-19T22:24:44.585Z",  
  "Id": "3b0d0080-87e7-48c6-b182-503ec743a08b",  
  "LastUpdatedTimestamp": "2019-06-19T22:24:44.585Z",  
  "LatestVersion": "67f918b9-efb4-40b0-b87c-de8c9faf085b",  
  "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/  
definition/functions/3b0d0080-87e7-48c6-b182-503ec743a08b/versions/67f918b9-  
efb4-40b0-b87c-de8c9faf085b",  
  "Name": "MyGreengrassFunctions"  
}
```

有关更多信息，请参阅 [AWS IoT Greengrass 开发人员指南中的如何使用 AWS 命令行界面配置本地资源访问权限](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 CreateFunctionDefinition](#) 中的。

## **create-group-certificate-authority**

以下代码示例演示了如何使用 `create-group-certificate-authority`。

### AWS CLI

为组创建证书颁发机构 (CA)



以下 `create-group-certificate-authority` 示例为指定组创建或轮换 CA。

```
aws greengrass create-group-certificate-authority \  
  --group-id "8eaadd72-ce4b-4f15-892a-0cc4f3a343f1"
```

输出：

```
{  
  "GroupCertificateAuthorityArn": "arn:aws:greengrass:us-west-2:123456789012:/  
greengrass/groups/8eaadd72-ce4b-4f15-892a-0cc4f3a343f1/certificateauthorities/  
d31630d674c4437f6c5dbc0dca56312a902171ce2d086c38e509c8EXAMPLEecc5"  
}
```

有关更多信息，请参阅 [《AWS 物联网 Greengrass 开发者指南》](#) 中的 [Io AWS T Greengrass 安全](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateGroupCertificateAuthority](#) 中的。

## create-group-version

以下代码示例演示了如何使用 `create-group-version`。

### AWS CLI

#### 创建 Greengrass 群组的版本

以下 `create-group-version` 示例创建群组版本并将其与指定的群组关联。该版本引用了内核、资源、连接器、函数和订阅版本，这些版本包含要包含在此组版本中的实体。必须先创建这些实体，然后才能创建组版本。

要使用初始版本创建资源定义，请使用 `create-resource-definition` 命令。要使用初始版本创建连接器定义，请使用该 `create-connector-definition` 命令。要使用初始版本创建函数定义，请使用该 `create-function-definition` 命令。要使用初始版本创建订阅定义，请使用该命令。要 `create-subscription-definition` 检索最新核心定义版本的 ARN，请使用 `get-group-version` 命令并指定最新组版本的 ID。

```
aws greengrass create-group-version \  
  --group-id "ce2e7d01-3240-4c24-b8e6-f6f6e7a9eeca" \  
  --core-definition-version-arn "arn:aws:greengrass:us-west-2:123456789012:/  
greengrass/definition/cores/6a630442-8708-4838-ad36-eb98849d975e/  
versions/6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba" \  
  --function-definition-version-arn "arn:aws:greengrass:us-west-2:123456789012:/  
greengrass/definition/functions/6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba/  
versions/6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba" \  
  --subscription-definition-version-arn "arn:aws:greengrass:us-west-2:123456789012:/  
greengrass/definition/subscriptions/6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba/  
versions/6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba" \  
  --resource-definition-version-arn "arn:aws:greengrass:us-west-2:123456789012:/  
greengrass/definition/resources/6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba/  
versions/6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba" \  
  --connector-definition-version-arn "arn:aws:greengrass:us-west-2:123456789012:/  
greengrass/definition/connectors/6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba/  
versions/6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba" \  
  --core-definition-version-id "6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba" \  
  --function-definition-version-id "6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba" \  
  --subscription-definition-version-id "6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba" \  
  --resource-definition-version-id "6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba" \  
  --connector-definition-version-id "6c87151b-1fb4-4cb2-8b31-6ee715d8f8ba"
```

```
--resource-definition-version-arn "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/resources/c8bb9ebc-c3fd-40a4-9c6a-568d75569d38/versions/a5f94d0b-f6bc-40f4-bb78-7a1c5fe13ba1" \  
--connector-definition-version-arn "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/connectors/55d0052b-0d7d-44d6-b56f-21867215e118/versions/78a3331b-895d-489b-8823-17b4f9f418a0" \  
--function-definition-version-arn "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/functions/3b0d0080-87e7-48c6-b182-503ec743a08b/versions/67f918b9-efb4-40b0-b87c-de8c9faf085b" \  
--subscription-definition-version-arn "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/subscriptions/9d611d57-5d5d-44bd-a3b4-fecbbdd69112/versions/aa645c47-ac90-420d-9091-8c7ffa4f103f"
```

输出：

```
{  
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/groups/ce2e7d01-3240-4c24-b8e6-f6f6e7a9eeca/versions/e10b0459-4345-4a09-88a4-1af1f5d34638",  
  "CreationTimestamp": "2019-06-20T18:42:47.020Z",  
  "Id": "ce2e7d01-3240-4c24-b8e6-f6f6e7a9eeca",  
  "Version": "e10b0459-4345-4a09-88a4-1af1f5d34638"  
}
```

有关更多信息，请参阅 [《AWS 物联网 Greengrass 开发者指南》中的物联网 Greengrass 组对象模型概述](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateGroupVersion](#) 中的。

## create-group

以下代码示例演示了如何使用 create-group。

### AWS CLI

创建 Greengrass 群组

以下 create-group 示例创建了一个名为的群组 cli-created-group。

```
aws greengrass create-group \  
  --name cli-created-group
```

输出：

```
{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
groups/4e22bd92-898c-436b-ade5-434d883ff749",
  "CreationTimestamp": "2019-06-25T18:07:17.688Z",
  "Id": "4e22bd92-898c-436b-ade5-434d883ff749",
  "LastUpdatedTimestamp": "2019-06-25T18:07:17.688Z",
  "Name": "cli-created-group"
}
```

有关更多信息，请参阅 [《AWS 物联网 Greengrass 开发者指南》中的物联网 Greengrass 组对象模型概述](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateGroup](#) 中的。

## create-logger-definition-version

以下代码示例演示了如何使用 create-logger-definition-version。

### AWS CLI

#### 创建记录器定义版本

以下 create-logger-definition-version 示例创建了一个记录器定义版本并将其与记录器定义相关联。该版本定义了四种日志配置：1) 核心设备文件系统上的系统组件日志，2) 核心设备文件系统上用户定义的 Lambda 函数日志，3) Amazon Logs 中的系统组件日志，以及 4) Amazon Lambda CloudWatch logs 中用户定义的 Lambda 函数日志。CloudWatch 注意：对于 CloudWatch 日志集成，您的群组角色必须授予相应的权限。

```
aws greengrass create-logger-definition-version \
  --logger-definition-id "a454b62a-5d56-4ca9-bdc4-8254e1662cb0" \
  --loggers "[{\\"Id\\":\\"1\\",\\"Component\\":\\"GreengrassSystem\\",\\"Level\\":\\"ERROR\\",\\"Space\\":10240,\\"Type\\":\\"FileSystem\\"},{\\"Id\\":\\"2\\",\\"Component\\":\\"Lambda\\",\\"Level\\":\\"INFO\\",\\"Space\\":10240,\\"Type\\":\\"FileSystem\\"},{\\"Id\\":\\"3\\",\\"Component\\":\\"GreengrassSystem\\",\\"Level\\":\\"WARN\\",\\"Type\\":\\"AWSCloudWatch\\"},{\\"Id\\":\\"4\\",\\"Component\\":\\"Lambda\\",\\"Level\\":\\"INFO\\",\\"Type\\":\\"AWSCloudWatch\\"}]"
```

输出：

```
{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/loggers/
a454b62a-5d56-4ca9-bdc4-8254e1662cb0/versions/49aedb1e-01a3-4d39-9871-3a052573f1ea",
```

```

"Version": "49aedb1e-01a3-4d39-9871-3a052573f1ea",
"CreationTimestamp": "2019-07-24T00:04:48.523Z",
"Id": "a454b62a-5d56-4ca9-bdc4-8254e1662cb0"
}

```

有关更多信息，请参阅《物联网 [Greengrass 开发者指南](#)》中的使用 [AWS 物联网 Greengrass 日志进行AWS 监控](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLoggerDefinitionVersion](#)中的。

## create-logger-definition

以下代码示例演示了如何使用 create-logger-definition。

### AWS CLI

#### 创建记录器定义

以下create-logger-definition示例创建了一个包含初始记录器定义版本的记录器定义。初始版本定义了三种日志配置：1) 核心设备文件系统上的系统组件日志，2) 核心设备文件系统上用户定义的 Lambda 函数日志，以及 3) Amazon Logs 中用户定义的 Lambda 函数日志。CloudWatch 注意：对于 CloudWatch 日志集成，您的群组角色必须授予相应的权限。

```

aws greengrass create-logger-definition \
  --name "LoggingConfigs" \
  --initial-version "{\"Loggers\":{\"Id\":\"1\",\"Component\":\"GreengrassSystem\",
  \"Level\":\"ERROR\",\"Space\":\"10240\",\"Type\":\"FileSystem\"},{\"Id\":\"2\",
  \"Component\":\"Lambda\",\"Level\":\"INFO\",\"Space\":\"10240\",\"Type\":\"FileSystem\"},
  {\"Id\":\"3\",\"Component\":\"Lambda\",\"Level\":\"INFO\",\"Type\":\"AWSCloudWatch\"}}}"

```

输出：

```

{
  "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/loggers/a454b62a-5d56-4ca9-bdc4-8254e1662cb0/versions/de1d9854-1588-4525-b25e-b378f60f2322",
  "Name": "LoggingConfigs",
  "LastUpdatedTimestamp": "2019-07-23T23:52:17.165Z",
  "LatestVersion": "de1d9854-1588-4525-b25e-b378f60f2322",
  "CreationTimestamp": "2019-07-23T23:52:17.165Z",
  "Id": "a454b62a-5d56-4ca9-bdc4-8254e1662cb0",
}

```

```
"Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/
loggers/a454b62a-5d56-4ca9-bdc4-8254e1662cb0"
}
```

有关更多信息，请参阅《物联网 [Greengrass 开发者指南](#)》中的使用 [AWS 物联网 Greengrass 日志进行AWS 监控](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLoggerDefinition](#)中的。

## create-resource-definition-version

以下代码示例演示了如何使用 create-resource-definition-version。

### AWS CLI

创建资源定义的版本

以下create-resource-definition-version示例创建了一个的新版本 TwilioAuthToken。

```
aws greengrass create-resource-definition-version \
  --resource-definition-id "c8bb9ebc-c3fd-40a4-9c6a-568d75569d38" \
  --resources "[{"Id": \"TwilioAuthToken\", \"Name\": \"MyTwilioAuthToken
\", \"ResourceDataContainer\": {\"SecretsManagerSecretResourceData\": {\"ARN\":
\"arn:aws:secretsmanager:us-west-2:123456789012:secret:greengrass-TwilioAuthToken-
ntS1p6\"}}}]"
```

输出：

```
{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/
resources/c8bb9ebc-c3fd-40a4-9c6a-568d75569d38/versions/b3bcada0-5fb6-42df-
bf0b-1ee4f15e769e",
  "CreationTimestamp": "2019-06-24T21:17:25.623Z",
  "Id": "c8bb9ebc-c3fd-40a4-9c6a-568d75569d38",
  "Version": "b3bcada0-5fb6-42df-bf0b-1ee4f15e769e"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateResourceDefinitionVersion](#)中的。

## create-resource-definition

以下代码示例演示了如何使用 create-resource-definition。

## AWS CLI

### 创建资源定义

以下 `create-resource-definition` 示例创建了一个资源定义，其中包含要在 Greengrass 组中使用的资源列表。在此示例中，通过提供资源列表来包含资源定义的初始版本。该列表包括一个用于 Twilio 授权令牌的资源和存储在 Secrets AWS Manager 中的密钥的 ARN。必须先创建密钥，然后才能创建资源定义。

```
aws greengrass create-resource-definition \  
  --name MyGreengrassResources \  
  --initial-version "{\"Resources\": [{\"Id\": \"TwilioAuthToken\  
\", \"Name\": \"MyTwilioAuthToken\", \"ResourceDataContainer\":  
  {\"SecretsManagerSecretResourceData\": {\"ARN\": \"arn:aws:secretsmanager:us-  
west-2:123456789012:secret:greengrass-TwilioAuthToken-ntSlp6\"}}]}\"
```

输出：

```
{  
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/  
resources/c8bb9ebc-c3fd-40a4-9c6a-568d75569d38",  
  "CreationTimestamp": "2019-06-19T21:51:28.212Z",  
  "Id": "c8bb9ebc-c3fd-40a4-9c6a-568d75569d38",  
  "LastUpdatedTimestamp": "2019-06-19T21:51:28.212Z",  
  "LatestVersion": "a5f94d0b-f6bc-40f4-bb78-7a1c5fe13ba1",  
  "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/  
definition/resources/c8bb9ebc-c3fd-40a4-9c6a-568d75569d38/versions/a5f94d0b-  
f6bc-40f4-bb78-7a1c5fe13ba1",  
  "Name": "MyGreengrassResources"  
}
```

有关更多信息，请参阅 [AWS IoT Greengrass 开发人员指南中的如何使用 AWS 命令行界面配置本地资源访问权限](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `CreateResourceDefinition`](#) 中的。

### **create-software-update-job**

以下代码示例演示了如何使用 `create-software-update-job`。

## AWS CLI

### 为内核创建软件更新任务

以下`create-software-update-job`示例创建了一个 over-the-air (OTA) 更新任务，用于更新名为的内核上的 AWS IoT Greengrass Core 软件。MyFirstGroup\_Core此命令需要一个 IAM 角色，该角色允许访问 Amazon S3 中的软件更新包，并包含`iot.amazonaws.com`为可信实体。

```
aws greengrass create-software-update-job \  
  --update-targets-architecture armv7l \  
  --update-targets ["arn:aws:iot:us-west-2:123456789012:thing/MyFirstGroup_Core \  
  \"] \  
  --update-targets-operating-system raspbian \  
  --software-to-update core \  
  --s3-url-signer-role arn:aws:iam::123456789012:role/OTA_signer_role \  
  --update-agent-log-level WARN
```

输出：

```
{  
  "IotJobId": "GreengrassUpdateJob_30b353e3-3af7-4786-be25-4c446663c09e",  
  "IotJobArn": "arn:aws:iot:us-west-2:123456789012:job/  
GreengrassUpdateJob_30b353e3-3af7-4786-be25-4c446663c09e",  
  "PlatformSoftwareVersion": "1.9.3"  
}
```

有关更多信息，请参阅《物联网 [Greengrass 开发者 AWS 指南](#)》中的 [IoT Greengrass 核心软件的 O AWS TA 更新](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSoftwareUpdateJob](#)中的。

## create-subscription-definition-version

以下代码示例演示了如何使用 `create-subscription-definition-version`。

### AWS CLI

#### 创建订阅定义的新版本

以下`create-subscription-definition-version`示例创建了包含三个订阅的新版本的订阅定义：触发通知、温度输入和输出状态。

```
aws greengrass create-subscription-definition-version \
  --subscription-definition-id "9d611d57-5d5d-44bd-a3b4-feccbdd69112" \
  --subscriptions "[{"Id": "TriggerNotification", "Source":
  \"arn:aws:lambda:us-west-2:123456789012:function:TempMonitor:GG_TempMonitor
  \", \"Subject\": \"twilio/txt\", \"Target\": \"arn:aws:greengrass:us-west-2:/:
  connectors/TwilioNotifications/versions/1\"}, {"Id": \"TemperatureInput\", \"Source
  \": \"cloud\", \"Subject\": \"temperature/input\", \"Target\": \"arn:aws:lambda:us-
  west-2:123456789012:function:TempMonitor:GG_TempMonitor\"}, {"Id\": \"OutputStatus
  \", \"Source\": \"arn:aws:greengrass:us-west-2:/:connectors/TwilioNotifications/
  versions/1\", \"Subject\": \"twilio/message/status\", \"Target\": \"cloud\"}]"]"
```

输出：

```
{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/
  subscriptions/9d611d57-5d5d-44bd-a3b4-feccbdd69112/versions/7b65dfae-50b6-4d0f-
  b3e0-27728bfb0620",
  "CreationTimestamp": "2019-06-24T21:21:33.837Z",
  "Id": "9d611d57-5d5d-44bd-a3b4-feccbdd69112",
  "Version": "7b65dfae-50b6-4d0f-b3e0-27728bfb0620"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSubscriptionDefinitionVersion](#)中的。

## create-subscription-definition

以下代码示例演示了如何使用 create-subscription-definition。

### AWS CLI

#### 创建订阅定义

以下create-subscription-definition示例创建订阅定义并指定其初始版本。初始版本包含三个订阅：一个用于连接器订阅的 MQTT 主题，一个用于允许函数接收来自物联网 AWS 的温度读数，另一个用于允许物 AWS 联网接收来自连接器的状态信息。该示例为之前使用 Lambda 的命令创建的 Lambda 函数别名提供了 ARN。create-alias

```
aws greengrass create-subscription-definition \
  --initial-version [{"Subscriptions": [{"Id":
  \"TriggerNotification\", \"Source\": \"arn:aws:lambda:us-
  west-2:123456789012:function:TempMonitor:GG_TempMonitor\", \"Subject\":
```



```
\"twilio/txt\", \"Target\": \"arn:aws:greengrass:us-west-2::/connectors/TwilioNotifications/versions/1\"},{\"Id\": \"TemperatureInput\", \"Source\": \"cloud\", \"Subject\": \"temperature/input\", \"Target\": \"arn:aws:lambda:us-west-2:123456789012:function:TempMonitor:GG_TempMonitor\"},{\"Id\": \"OutputStatus\", \"Source\": \"arn:aws:greengrass:us-west-2::/connectors/TwilioNotifications/versions/1\", \"Subject\": \"twilio/message/status\", \"Target\": \"cloud\"}]}"
```

输出：

```
{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/subscriptions/9d611d57-5d5d-44bd-a3b4-feccbdd69112",
  "CreationTimestamp": "2019-06-19T22:34:26.677Z",
  "Id": "9d611d57-5d5d-44bd-a3b4-feccbdd69112",
  "LastUpdatedTimestamp": "2019-06-19T22:34:26.677Z",
  "LatestVersion": "aa645c47-ac90-420d-9091-8c7ffa4f103f",
  "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/subscriptions/9d611d57-5d5d-44bd-a3b4-feccbdd69112/versions/aa645c47-ac90-420d-9091-8c7ffa4f103f"
}
```

有关更多信息，请参阅 [AWS IoT Greengrass 开发者指南中的连接器入门 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 CreateSubscriptionDefinition](#) 中的。

## delete-connector-definition

以下代码示例演示了如何使用 delete-connector-definition。

### AWS CLI

#### 删除连接器定义

以下 delete-connector-definition 示例删除指定的 Greengrass 连接器定义。如果删除组使用的连接器定义，则无法成功部署该组。

```
aws greengrass delete-connector-definition \
  --connector-definition-id "b5c4ebfd-f672-49a3-83cd-31c7216a7bb8"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 DeleteConnectorDefinition](#) 中的。

## delete-core-definition

以下代码示例演示了如何使用 delete-core-definition。

### AWS CLI

#### 删除核心定义

以下delete-core-definition示例删除指定的 Greengrass 核心定义，包括所有版本。如果您删除与 Greengrass 群组关联的核心，则该群组将无法成功部署。

```
aws greengrass delete-core-definition \  
  --core-definition-id "ff36cc5f-9f98-4994-b468-9d9b6dc52abd"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCoreDefinition](#)中的。

## delete-device-definition

以下代码示例演示了如何使用 delete-device-definition。

### AWS CLI

#### 删除设备定义

以下delete-device-definition示例删除了指定的设备定义，包括其所有版本。如果您删除群组版本使用的设备定义版本，则无法成功部署群组版本。

```
aws greengrass delete-device-definition \  
  --device-definition-id "f9ba083d-5ad4-4534-9f86-026a45df1ccd"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDeviceDefinition](#)中的。

## delete-function-definition

以下代码示例演示了如何使用 delete-function-definition。

### AWS CLI

#### 删除函数定义

以下delete-function-definition示例删除指定的 Greengrass 函数定义。如果您删除群组使用的函数定义，则无法成功部署该组。

```
aws greengrass delete-function-definition \  
  --function-definition-id "fd4b906a-dff3-4c1b-96eb-52ebfcfac06a"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFunctionDefinition](#)中的。

## delete-group

以下代码示例演示了如何使用 delete-group。

### AWS CLI

#### 删除群组

以下delete-group示例删除指定的 Greengrass 组。

```
aws greengrass delete-group \  
  --group-id "4e22bd92-898c-436b-ade5-434d883ff749"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteGroup](#)中的。

## delete-logger-definition

以下代码示例演示了如何使用 delete-logger-definition。

### AWS CLI

#### 删除记录器定义

以下delete-logger-definition示例删除了指定的记录器定义，包括所有记录器定义版本。如果删除组版本使用的记录器定义版本，则无法成功部署该组版本。

```
aws greengrass delete-logger-definition \  
  --logger-definition-id "a454b62a-5d56-4ca9-bdc4-8254e1662cb0"
```

此命令不生成任何输出。

有关更多信息，请参阅《物联网 [Greengrass 开发者指南](#)》中的使用 [AWS 物联网 Greengrass 日志进行AWS 监控](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLoggerDefinition](#)中的。

## delete-resource-definition

以下代码示例演示了如何使用 delete-resource-definition。

### AWS CLI

#### 删除资源定义

以下delete-resource-definition示例删除了指定的资源定义，包括所有资源版本。如果您删除某个组使用的资源定义，则该组将无法成功部署。

```
aws greengrass delete-resource-definition \  
  --resource-definition-id "ad8c101d-8109-4b0e-b97d-9cc5802ab658"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteResourceDefinition](#)中的。

## delete-subscription-definition

以下代码示例演示了如何使用 delete-subscription-definition。

### AWS CLI

#### 删除订阅定义

以下delete-subscription-definition示例删除了指定的 Greengrass 订阅定义。如果您删除群组正在使用的订阅，则该群组将无法成功部署。

```
aws greengrass delete-subscription-definition \  
  --subscription-definition-id "cd6f1c37-d9a4-4e90-be94-01a7404f5967"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSubscriptionDefinition](#)中的。

## disassociate-role-from-group

以下代码示例演示了如何使用 disassociate-role-from-group。

### AWS CLI

取消角色与 Greengrass 群组的关联

以下disassociate-role-from-group示例取消了 IAM 角色与指定的 Greengrass 组的关联。

```
aws greengrass disassociate-role-from-group \  
  --group-id 2494ee3f-7f8a-4e92-a78b-d205f808b84b
```

输出：

```
{  
  "DisassociatedAt": "2019-09-10T20:05:49Z"  
}
```

有关更多信息，请参阅 AWS IoT Greengrass 开发者指南[中的配置群组角色](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateRoleFromGroup](#)中的。

## disassociate-service-role-from-account

以下代码示例演示了如何使用 disassociate-service-role-from-account。

### AWS CLI

取消服务角色与您的 AWS 账户的关联

以下disassociate-service-role-from-account示例删除了与您的 AWS 账户关联的服务角色。如果您未在任何 AWS 区域使用服务角色，请使用delete-role-policy命令将AWSGreengrassResourceAccessRolePolicy托管策略与该角色分离，然后使用delete-role命令删除该角色。

```
aws greengrass disassociate-service-role-from-account
```

输出：

```
{
  "DisassociatedAt": "2019-06-25T22:12:55Z"
}
```

有关更多信息，请参阅《物联网 [AWS Greengrass 开发者指南](#)》中的 [Greengrass 服务角色](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateServiceRoleFromAccount](#) 中的。

## get-associated-role

以下代码示例演示了如何使用 `get-associated-role`。

### AWS CLI

获取与 Greengrass 群组关联的角色

以下 `get-associated-role` 示例获取与指定 Greengrass 组关联的 IAM 角色。本地 Lambda 函数和连接器使用组角色来访问服务 AWS。

```
aws greengrass get-associated-role \
  --group-id 2494ee3f-7f8a-4e92-a78b-d205f808b84b
```

输出：

```
{
  "RoleArn": "arn:aws:iam::123456789012:role/GG-Group-Role",
  "AssociatedAt": "2019-09-10T20:03:30Z"
}
```

有关更多信息，请参阅 AWS IoT Greengrass 开发者指南 [中的配置群组角色](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetAssociatedRole](#) 中的。

## get-bulk-deployment-status

以下代码示例演示了如何使用 `get-bulk-deployment-status`。

### AWS CLI

检查批量部署的状态

以下 `get-bulk-deployment-status` 示例检索指定批量部署操作的状态信息。在此示例中，指定要部署的组的文件具有无效的输入记录。

```
aws greengrass get-bulk-deployment-status \  
  --bulk-deployment-id "870fb41b-6288-4e0c-bc76-a7ba4b4d3267"
```

输出：

```
{  
  "BulkDeploymentMetrics": {  
    "InvalidInputRecords": 1,  
    "RecordsProcessed": 1,  
    "RetryAttempts": 0  
  },  
  "BulkDeploymentStatus": "Completed",  
  "CreatedAt": "2019-06-25T16:11:33.265Z",  
  "tags": {}  
}
```

有关更多信息，请参阅 AWS IoT Greengrass [开发人员指南中的为群组创建批量部署](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetBulkDeploymentStatus](#) 中的。

## get-connectivity-info

以下代码示例演示了如何使用 `get-connectivity-info`。

### AWS CLI

获取 Greengrass 内核的连接信息

以下 `get-connectivity-info` 示例显示了设备可用于连接到指定的 Greengrass 核心的端点。连接信息是 IP 地址或域名的列表，以及相应的端口号和可选的客户定义元数据。

```
aws greengrass get-connectivity-info \  
  --thing-name "MyGroup_Core"
```

输出：

```
{  
  "ConnectivityInfo": [  
    {
```

```
    "Metadata": "",
    "PortNumber": 8883,
    "HostAddress": "127.0.0.1",
    "Id": "AUTOIP_127.0.0.1_0"
  },
  {
    "Metadata": "",
    "PortNumber": 8883,
    "HostAddress": "192.168.1.3",
    "Id": "AUTOIP_192.168.1.3_1"
  },
  {
    "Metadata": "",
    "PortNumber": 8883,
    "HostAddress": "::1",
    "Id": "AUTOIP_::1_2"
  },
  {
    "Metadata": "",
    "PortNumber": 8883,
    "HostAddress": "fe80::1e69:ed93:f5b:f6d",
    "Id": "AUTOIP_fe80::1e69:ed93:f5b:f6d_3"
  }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetConnectivityInfo](#)中的。

## get-connector-definition-version

以下代码示例演示了如何使用 `get-connector-definition-version`。

### AWS CLI

检索有关连接器定义的特定版本的信息

以下`get-connector-definition-version`示例检索有关指定连接器定义的指定版本的信息。要检索所有版本的连接器定义的 ID，请使用`list-connector-definition-versions`命令。要检索添加到连接器定义中的上一个版本的 ID，请使用`get-connector-definition`命令并检查`LatestVersion`属性。

```
aws greengrass get-connector-definition-version \
```



```
--connector-definition-id "b5c4ebfd-f672-49a3-83cd-31c7216a7bb8" \  
--connector-definition-version-id "63c57963-c7c2-4a26-a7e2-7bf478ea2623"
```

输出：

```
{  
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/  
connectors/b5c4ebfd-f672-49a3-83cd-31c7216a7bb8/versions/63c57963-c7c2-4a26-  
a7e2-7bf478ea2623",  
  "CreationTimestamp": "2019-06-19T19:30:01.300Z",  
  "Definition": {  
    "Connectors": [  
      {  
        "ConnectorArn": "arn:aws:greengrass:us-west-2:./connectors/SNS/  
versions/1",  
        "Id": "MySNSConnector",  
        "Parameters": {  
          "DefaultSNSArn": "arn:aws:sns:us-  
west-2:123456789012:GGConnectorTopic"  
        }  
      }  
    ]  
  },  
  "Id": "b5c4ebfd-f672-49a3-83cd-31c7216a7bb8",  
  "Version": "63c57963-c7c2-4a26-a7e2-7bf478ea2623"  
}
```

有关更多信息，请参阅《AWS 物联网 Greengrass [开发者指南](#)》中的使用 [Greengrass 连接器与服务](#)和[协议集成](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetConnectorDefinitionVersion](#)中的。

## get-connector-definition

以下代码示例演示了如何使用 get-connector-definition。

### AWS CLI

检索有关连接器定义的信息

以下get-connector-definition示例检索有关指定连接器定义的信息。要检索连接器定义的ID，请使用list-connector-definitions命令。

```
aws greengrass get-connector-definition \  
  --connector-definition-id "b5c4ebfd-f672-49a3-83cd-31c7216a7bb8"
```

输出：

```
{  
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/  
connectors/b5c4ebfd-f672-49a3-83cd-31c7216a7bb8",  
  "CreationTimestamp": "2019-06-19T19:30:01.300Z",  
  "Id": "b5c4ebfd-f672-49a3-83cd-31c7216a7bb8",  
  "LastUpdatedTimestamp": "2019-06-19T19:30:01.300Z",  
  "LatestVersion": "63c57963-c7c2-4a26-a7e2-7bf478ea2623",  
  "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/  
definition/connectors/b5c4ebfd-f672-49a3-83cd-31c7216a7bb8/versions/63c57963-  
c7c2-4a26-a7e2-7bf478ea2623",  
  "Name": "MySNSConnector",  
  "tags": {}  
}
```

有关更多信息，请参阅《AWS 物联网 Greengrass [开发者指南](#)》中的[使用 Greengrass 连接器与服务](#)和[协议集成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetConnectorDefinition](#)中的。

## get-core-definition-version

以下代码示例演示了如何使用 `get-core-definition-version`。

### AWS CLI

检索有关 Greengrass 核心定义特定版本的详细信息

以下 `get-core-definition-version` 示例检索有关指定核心定义的指定版本的信息。要检索所有版本的核心定义的 ID，请使用 `list-core-definition-versions` 命令。要检索添加到核心定义中的上一个版本的 ID，请使用 `get-core-definition` 命令并检查 `LatestVersion` 属性。

```
aws greengrass get-core-definition-version \  
  --core-definition-id "c906ed39-a1e3-4822-a981-7b9bd57b4b46" \  
  --core-definition-version-id "42aeac3-fd9d-4312-a8fd-ffa9404a20e0"
```

输出：

```
{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/cores/
c906ed39-a1e3-4822-a981-7b9bd57b4b46/versions/42aeeac3-fd9d-4312-a8fd-ffa9404a20e0",
  "CreationTimestamp": "2019-06-18T16:21:21.351Z",
  "Definition": {
    "Cores": [
      {
        "CertificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/928dea7b82331b47c3ff77b0e763fc5e64e2f7c884e6ef391baed9b6b8e21b45",
        "Id": "1a39aac7-0885-4417-91f6-23e4cea6c511",
        "SyncShadow": false,
        "ThingArn": "arn:aws:iot:us-west-2:123456789012:thing/
GGGroup4Pi3_Core"
      }
    ]
  },
  "Id": "c906ed39-a1e3-4822-a981-7b9bd57b4b46",
  "Version": "42aeeac3-fd9d-4312-a8fd-ffa9404a20e0"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCoreDefinitionVersion](#)中的。

## get-core-definition

以下代码示例演示了如何使用 `get-core-definition`。

### AWS CLI

检索 Greengrass 核心定义的详细信息

以下`get-core-definition`示例检索有关指定核心定义的信息。要检索核心定义的 ID，请使用`list-core-definitions`命令。

```
aws greengrass get-core-definition \
  --core-definition-id "c906ed39-a1e3-4822-a981-7b9bd57b4b46"
```

输出：

```
{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/
cores/237d6916-27cf-457f-ba0c-e86cfb5d25cd",
```

```
"CreationTimestamp": "2018-10-18T04:47:06.721Z",
"Id": "237d6916-27cf-457f-ba0c-e86cfb5d25cd",
"LastUpdatedTimestamp": "2018-10-18T04:47:06.721Z",
"LatestVersion": "bd2cd6d4-2bc5-468a-8962-39e071e34b68",
"LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/cores/237d6916-27cf-457f-ba0c-e86cfb5d25cd/versions/
bd2cd6d4-2bc5-468a-8962-39e071e34b68",
"tags": {}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCoreDefinition](#)中的。

## get-deployment-status

以下代码示例演示了如何使用 `get-deployment-status`。

### AWS CLI

#### 检索部署的状态

以下`get-deployment-status`示例检索指定 Greengrass 组的指定部署的状态。要获取部署 ID，请使用`list-deployments`命令并指定组 ID。

```
aws greengrass get-deployment-status \
  --group-id "1013db12-8b58-45ff-acc7-704248f66731" \
  --deployment-id "1065b8a0-812b-4f21-9d5d-e89b232a530f"
```

#### 输出：

```
{
  "DeploymentStatus": "Success",
  "DeploymentType": "NewDeployment",
  "UpdatedAt": "2019-06-18T17:04:44.761Z"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDeploymentStatus](#)中的。

## get-device-definition-version

以下代码示例演示了如何使用 `get-device-definition-version`。

## AWS CLI

### 获取设备定义版本

以下`get-device-definition-version`示例检索有关指定设备定义的指定版本的信息。要检索设备定义的所有版本的 ID，请使用`list-device-definition-versions`命令。要检索添加到设备定义中的上一个版本的 ID，请使用`get-device-definition`命令并检查`LatestVersion`属性。

```
aws greengrass get-device-definition-version \  
  --device-definition-id "f9ba083d-5ad4-4534-9f86-026a45df1ccd" \  
  --device-definition-version-id "83c13984-6fed-447e-84d5-5b8aa45d5f71"
```

输出：

```
{  
  "Definition": {  
    "Devices": [  
      {  
        "CertificateArn": "arn:aws:iot:us-west-2:123456789012:cert/6c52ce1b47bde88a637e9ccdd45fe4e4c2c0a75a6866f8f63d980ee22fa51e02",  
        "ThingArn": "arn:aws:iot:us-west-2:123456789012:thing/ExteriorTherm",  
        "SyncShadow": true,  
        "Id": "ExteriorTherm"  
      },  
      {  
        "CertificateArn": "arn:aws:iot:us-west-2:123456789012:cert/66a415ec415668c2349a76170b64ac0878231c1e21ec83c10e92a18bd568eb92",  
        "ThingArn": "arn:aws:iot:us-west-2:123456789012:thing/InteriorTherm",  
        "SyncShadow": true,  
        "Id": "InteriorTherm"  
      }  
    ]  
  },  
  "Version": "83c13984-6fed-447e-84d5-5b8aa45d5f71",  
  "CreationTimestamp": "2019-09-11T00:15:09.838Z",  
  "Id": "f9ba083d-5ad4-4534-9f86-026a45df1ccd",  
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/devices/f9ba083d-5ad4-4534-9f86-026a45df1ccd/versions/83c13984-6fed-447e-84d5-5b8aa45d5f71"
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDeviceDefinitionVersion](#)中的。

## get-device-definition

以下代码示例演示了如何使用 `get-device-definition`。

### AWS CLI

#### 获取设备定义

以下`get-device-definition`示例检索有关指定设备定义的信息。要检索您的设备定义的 ID，请使用`list-device-definitions`命令。

```
aws greengrass get-device-definition \  
  --device-definition-id "f9ba083d-5ad4-4534-9f86-026a45df1ccd"
```

输出：

```
{  
  "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/  
greengrass/definition/devices/f9ba083d-5ad4-4534-9f86-026a45df1ccd/  
versions/83c13984-6fed-447e-84d5-5b8aa45d5f71",  
  "Name": "TemperatureSensors",  
  "tags": {},  
  "LastUpdatedTimestamp": "2019-09-11T00:19:03.698Z",  
  "LatestVersion": "83c13984-6fed-447e-84d5-5b8aa45d5f71",  
  "CreationTimestamp": "2019-09-11T00:11:06.197Z",  
  "Id": "f9ba083d-5ad4-4534-9f86-026a45df1ccd",  
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/  
devices/f9ba083d-5ad4-4534-9f86-026a45df1ccd"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDeviceDefinition](#)中的。

## get-function-definition-version

以下代码示例演示了如何使用 `get-function-definition-version`。

## AWS CLI

### 检索有关 Lambda 函数特定版本的详细信息

以下内容 `get-function-definition-version` 检索有关指定函数定义的指定版本的信息。要检索函数定义的所有版本的 ID，请使用 `list-function-definition-versions` 命令。要检索添加到函数定义中的上一个版本的 ID，请使用 `get-function-definition` 命令并检查 `LatestVersion` 属性。

```
aws greengrass get-function-definition-version \  
  --function-definition-id "063f5d1a-1dd1-40b4-9b51-56f8993d0f85" \  
  --function-definition-version-id "9748fda7-1589-4fcc-ac94-f5559e88678b"
```

输出：

```
{  
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/  
functions/063f5d1a-1dd1-40b4-9b51-56f8993d0f85/versions/9748fda7-1589-4fcc-ac94-  
f5559e88678b",  
  "CreationTimestamp": "2019-06-18T17:04:30.776Z",  
  "Definition": {  
    "Functions": [  
      {  
        "FunctionArn": "arn:aws:lambda::function:GGIPDetector:1",  
        "FunctionConfiguration": {  
          "Environment": {},  
          "MemorySize": 32768,  
          "Pinned": true,  
          "Timeout": 3  
        },  
        "Id": "26b69bdb-e547-46bc-9812-84ec04b6cc8c"  
      },  
      {  
        "FunctionArn": "arn:aws:lambda:us-  
west-2:123456789012:function:Greengrass_HelloWorld:GG_HelloWorld",  
        "FunctionConfiguration": {  
          "EncodingType": "json",  
          "Environment": {  
            "Variables": {}  
          },  
          "MemorySize": 16384,  
          "Pinned": true,  
          "Timeout": 25  
        }  
      }  
    ]  
  }  
}
```

```

        },
        "Id": "384465a8-eedf-48c6-b793-4c35f7bfae9b"
    }
]
},
"Id": "063f5d1a-1dd1-40b4-9b51-56f8993d0f85",
"Version": "9748fda7-1589-4fcc-ac94-f5559e88678b"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFunctionDefinitionVersion](#)中的。

## get-function-definition

以下代码示例演示了如何使用 get-function-definition。

### AWS CLI

#### 检索函数定义

以下get-function-definition示例显示了指定函数定义的详细信息。要检索函数定义的 ID，请使用list-function-definitions命令。

```

aws greengrass get-function-definition \
  --function-definition-id "063f5d1a-1dd1-40b4-9b51-56f8993d0f85"

```

输出：

```

{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/
functions/063f5d1a-1dd1-40b4-9b51-56f8993d0f85",
  "CreationTimestamp": "2019-06-18T16:21:21.431Z",
  "Id": "063f5d1a-1dd1-40b4-9b51-56f8993d0f85",
  "LastUpdatedTimestamp": "2019-06-18T16:21:21.431Z",
  "LatestVersion": "9748fda7-1589-4fcc-ac94-f5559e88678b",
  "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/functions/063f5d1a-1dd1-40b4-9b51-56f8993d0f85/
versions/9748fda7-1589-4fcc-ac94-f5559e88678b",
  "tags": {}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFunctionDefinition](#)中的。



## get-group-certificate-authority

以下代码示例演示了如何使用 `get-group-certificate-authority`。

### AWS CLI

检索与 Greengrass 群组关联的 CA

以下 `get-group-certificate-authority` 示例检索与指定 Greengrass 组关联的证书颁发机构 (CA)。要获取证书颁发机构 ID，请使用 `list-group-certificate-authorities` 命令并指定组 ID。

```
aws greengrass get-group-certificate-authority \
  --group-id "1013db12-8b58-45ff-acc7-704248f66731" \
  --certificate-authority-id
  "f0430e1736ea8ed30cc5d5de9af67a7e3586bad9ae4d89c2a44163f65fdd8cf6"
```

输出：

```
{
  "GroupCertificateAuthorityArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/groups/1013db12-8b58-45ff-acc7-704248f66731/certificateauthorities/f0430e1736ea8ed30cc5d5de9af67a7e3586bad9ae4d89c2a44163f65fdd8cf6",
  "GroupCertificateAuthorityId":
  "f0430e1736ea8ed30cc5d5de9af67a7e3586bad9ae4d89c2a44163f65fdd8cf6",
  "PemEncodedCertificate": "-----BEGIN CERTIFICATE-----
MIICiTCCAFICCCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBWEXAMPLEGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDEXAMPLEEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAEXAMPLESDBb25zb2x1MRIwEAYDVQQDEwLUZXN0Q21sYWxhZAd
BgkqhkiG9w0BCQEWEG5vb25lQGftYXpvbi5jEXAMPLENMTI1MjA0NTIxWhcN
MTIwNDI0MjA0EXAMPLEBiDELMAKGA1UEBhMCMVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWEXAMPLEDASBgNVBA5TC01BTSDBb25z
b2x1MRIwEAYDVQQDEwLUZXN0Q21sYWEXAMPLEGkqhkiG9w0BCQEWEG5vb25lQGft
YXpvbi5EXAMPLE8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CEXAMPLE93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswYEXAMPLEEgPE
Ibb30hjZnzcvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKEXAMPLEAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetGroupCertificateAuthority](#)中的。

## get-group-certificate-configuration

以下代码示例演示了如何使用 `get-group-certificate-configuration`。

### AWS CLI

检索 Greengrass 组使用的证书颁发机构的配置

以下`get-group-certificate-configuration`示例检索指定 Greengrass 组使用的证书颁发机构 (CA) 的配置。

```
aws greengrass get-group-certificate-configuration \  
  --group-id "1013db12-8b58-45ff-acc7-704248f66731"
```

输出：

```
{  
  "CertificateAuthorityExpiryInMilliseconds": 2524607999000,  
  "CertificateExpiryInMilliseconds": 604800000,  
  "GroupId": "1013db12-8b58-45ff-acc7-704248f66731"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetGroupCertificateConfiguration](#)中的。

## get-group-version

以下代码示例演示了如何使用 `get-group-version`。

### AWS CLI

检索有关 Greengrass 群组版本的信息

以下`get-group-version`示例检索有关指定组的指定版本的信息。要检索该组所有版本的 ID，请使用`list-group-versions`命令。要检索上次添加到该组的版本的 ID，请使用`get-group`命令并检查`LatestVersion`属性。

```
aws greengrass get-group-version \  
  --group-id "1013db12-8b58-45ff-acc7-704248f66731" \  
  --group-version-id "115136b3-cfd7-4462-b77f-8741a4b00e5e"
```

输出：

```
{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
groups/1013db12-8b58-45ff-acc7-704248f66731/versions/115136b3-cfd7-4462-
b77f-8741a4b00e5e",
  "CreationTimestamp": "2019-06-18T17:04:30.915Z",
  "Definition": {
    "CoreDefinitionVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/cores/c906ed39-a1e3-4822-a981-7b9bd57b4b46/versions/42aeeac3-
fd9d-4312-a8fd-ffa9404a20e0",
    "FunctionDefinitionVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/functions/063f5d1a-1dd1-40b4-9b51-56f8993d0f85/
versions/9748fda7-1589-4fcc-ac94-f5559e88678b",
    "SubscriptionDefinitionVersionArn": "arn:aws:greengrass:us-
west-2:123456789012:/greengrass/definition/subscriptions/70e49321-83d5-45d2-
bc09-81f4917ae152/versions/88ae8699-12ac-4663-ba3f-4d7f0519140b"
  },
  "Id": "1013db12-8b58-45ff-acc7-704248f66731",
  "Version": "115136b3-cfd7-4462-b77f-8741a4b00e5e"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetGroupVersion](#)中的。

## get-group

以下代码示例演示了如何使用 get-group。

### AWS CLI

检索有关 Greengrass 群组的信息

以下get-group示例检索有关指定 Greengrass 组的信息。要检索您的群组的 ID，请使用list-groups命令。

```
aws greengrass get-group \
  --group-id "1013db12-8b58-45ff-acc7-704248f66731"
```

输出：

```
{
```

```

    "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
groups/1013db12-8b58-45ff-acc7-704248f66731",
    "CreationTimestamp": "2019-06-18T16:21:21.457Z",
    "Id": "1013db12-8b58-45ff-acc7-704248f66731",
    "LastUpdatedTimestamp": "2019-06-18T16:21:21.457Z",
    "LatestVersion": "115136b3-cfd7-4462-b77f-8741a4b00e5e",
    "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
groups/1013db12-8b58-45ff-acc7-704248f66731/versions/115136b3-cfd7-4462-
b77f-8741a4b00e5e",
    "Name": "GGGroup4Pi3",
    "tags": {}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetGroup](#)中的。

## get-logger-definition-version

以下代码示例演示了如何使用 `get-logger-definition-version`。

### AWS CLI

检索有关记录器定义版本的信息

以下`get-logger-definition-version`示例检索有关指定记录器定义的指定版本的信息。要检索所有版本的记录器定义的 ID，请使用`list-logger-definition-versions`命令。要检索添加到记录器定义中的上一个版本的 ID，请使用`get-logger-definition`命令并检查`LatestVersion`属性。

```

aws greengrass get-logger-definition-version \
  --logger-definition-id "49eeeb66-f1d3-4e34-86e3-3617262abf23" \
  --logger-definition-version-id "5e3f6f64-a565-491e-8de0-3c0d8e0f2073"

```

输出：

```

{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/loggers/49eeeb66-f1d3-4e34-86e3-3617262abf23/versions/5e3f6f64-
a565-491e-8de0-3c0d8e0f2073",
  "CreationTimestamp": "2019-05-08T16:10:13.866Z",
  "Definition": {
    "Loggers": []
  }
}

```

```
  },  
  "Id": "49eeeb66-f1d3-4e34-86e3-3617262abf23",  
  "Version": "5e3f6f64-a565-491e-8de0-3c0d8e0f2073"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLoggerDefinitionVersion](#)中的。

## get-logger-definition

以下代码示例演示了如何使用 `get-logger-definition`。

### AWS CLI

检索有关记录器定义的信息

以下`get-logger-definition`示例检索有关指定记录器定义的信息。要检索记录器定义的 ID，请使用`list-logger-definitions`命令。

```
aws greengrass get-logger-definition \  
  --logger-definition-id "49eeeb66-f1d3-4e34-86e3-3617262abf23"
```

输出：

```
{  
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/  
loggers/49eeeb66-f1d3-4e34-86e3-3617262abf23",  
  "CreationTimestamp": "2019-05-08T16:10:13.809Z",  
  "Id": "49eeeb66-f1d3-4e34-86e3-3617262abf23",  
  "LastUpdatedTimestamp": "2019-05-08T16:10:13.809Z",  
  "LatestVersion": "5e3f6f64-a565-491e-8de0-3c0d8e0f2073",  
  "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/  
definition/loggers/49eeeb66-f1d3-4e34-86e3-3617262abf23/versions/5e3f6f64-  
a565-491e-8de0-3c0d8e0f2073",  
  "tags": {}  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLoggerDefinition](#)中的。

## get-resource-definition-version

以下代码示例演示了如何使用 `get-resource-definition-version`。

## AWS CLI

### 检索有关资源定义特定版本的信息

以下`get-resource-definition-version`示例检索有关指定资源定义的指定版本的信息。要检索资源定义的所有版本的 ID，请使用`list-resource-definition-versions`命令。要检索添加到资源定义中的上一个版本的 ID，请使用`get-resource-definition`命令并检查`LatestVersion`属性。

```
aws greengrass get-resource-definition-version \  
  --resource-definition-id "ad8c101d-8109-4b0e-b97d-9cc5802ab658" \  
  --resource-definition-version-id "26e8829a-491a-464d-9c87-664bf6f6f2be"
```

输出：

```
{  
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/  
definition/resources/ad8c101d-8109-4b0e-b97d-9cc5802ab658/  
versions/26e8829a-491a-464d-9c87-664bf6f6f2be",  
  "CreationTimestamp": "2019-06-19T16:40:59.392Z",  
  "Definition": {  
    "Resources": [  
      {  
        "Id": "26ff3f7b-839a-4217-9fdc-a218308b3963",  
        "Name": "usb-port",  
        "ResourceDataContainer": {  
          "LocalDeviceResourceData": {  
            "GroupOwnerSetting": {  
              "AutoAddGroupOwner": false  
            },  
            "SourcePath": "/dev/bus/usb"  
          }  
        }  
      }  
    ]  
  },  
  "Id": "ad8c101d-8109-4b0e-b97d-9cc5802ab658",  
  "Version": "26e8829a-491a-464d-9c87-664bf6f6f2be"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetResourceDefinitionVersion](#)中的。

## get-resource-definition

以下代码示例演示了如何使用 `get-resource-definition`。

### AWS CLI

检索有关资源定义的信息

以下`get-resource-definition`示例检索有关指定资源定义的信息。要检索资源定义的 ID，请使用`list-resource-definitions`命令。

```
aws greengrass get-resource-definition \
  --resource-definition-id "ad8c101d-8109-4b0e-b97d-9cc5802ab658"
```

输出：

```
{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/
resources/ad8c101d-8109-4b0e-b97d-9cc5802ab658",
  "CreationTimestamp": "2019-06-19T16:40:59.261Z",
  "Id": "ad8c101d-8109-4b0e-b97d-9cc5802ab658",
  "LastUpdatedTimestamp": "2019-06-19T16:40:59.261Z",
  "LatestVersion": "26e8829a-491a-464d-9c87-664bf6f6f2be",
  "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/resources/ad8c101d-8109-4b0e-b97d-9cc5802ab658/
versions/26e8829a-491a-464d-9c87-664bf6f6f2be",
  "tags": {}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetResourceDefinition](#)中的。

## get-service-role-for-account

以下代码示例演示了如何使用 `get-service-role-for-account`。

### AWS CLI

检索与您的账户关联的服务角色的详细信息

以下`get-service-role-for-account`示例检索与您的 AWS 账户关联的服务角色的相关信息。

```
aws greengrass get-service-role-for-account
```

输出：

```
{
  "AssociatedAt": "2018-10-18T15:59:20Z",
  "RoleArn": "arn:aws:iam::123456789012:role/service-role/Greengrass_ServiceRole"
}
```

有关更多信息，请参阅《物联网 [AWS Greengrass 开发者指南](#)》中的 [Greengrass 服务角色](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetServiceRoleForAccount](#) 中的。

## get-subscription-definition-version

以下代码示例演示了如何使用 `get-subscription-definition-version`。

AWS CLI

检索有关订阅定义特定版本的信息

以下 `get-subscription-definition-version` 示例检索有关指定订阅定义的指定版本的信息。要检索所有版本的订阅定义的 ID，请使用 `list-subscription-definition-versions` 命令。要检索添加到订阅定义中的上一个版本的 ID，请使用 `get-subscription-definition` 命令并检查 `LatestVersion` 属性。

```
aws greengrass get-subscription-definition-version \
  --subscription-definition-id "70e49321-83d5-45d2-bc09-81f4917ae152" \
  --subscription-definition-version-id "88ae8699-12ac-4663-ba3f-4d7f0519140b"
```

输出：

```
{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/subscriptions/70e49321-83d5-45d2-bc09-81f4917ae152/versions/88ae8699-12ac-4663-ba3f-4d7f0519140b",
  "CreationTimestamp": "2019-06-18T17:03:52.499Z",
  "Definition": {
    "Subscriptions": [
      {
        "Id": "692c4484-d89f-4f64-8edd-1a041a65e5b6",
```



```

        "Source": "arn:aws:lambda:us-
west-2:123456789012:function:Greengrass_HelloWorld:GG_HelloWorld",
        "Subject": "hello/world",
        "Target": "cloud"
    }
]
},
"Id": "70e49321-83d5-45d2-bc09-81f4917ae152",
"Version": "88ae8699-12ac-4663-ba3f-4d7f0519140b"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSubscriptionDefinitionVersion](#)中的。

## get-subscription-definition

以下代码示例演示了如何使用 `get-subscription-definition`。

### AWS CLI

检索有关订阅定义的信息

以下`get-subscription-definition`示例检索有关指定订阅定义的信息。要检索您的订阅定义的 ID，请使用`list-subscription-definitions`命令。

```

aws greengrass get-subscription-definition \
  --subscription-definition-id "70e49321-83d5-45d2-bc09-81f4917ae152"

```

输出：

```

{
  "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/
subscriptions/70e49321-83d5-45d2-bc09-81f4917ae152",
  "CreationTimestamp": "2019-06-18T17:03:52.392Z",
  "Id": "70e49321-83d5-45d2-bc09-81f4917ae152",
  "LastUpdatedTimestamp": "2019-06-18T17:03:52.392Z",
  "LatestVersion": "88ae8699-12ac-4663-ba3f-4d7f0519140b",
  "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/subscriptions/70e49321-83d5-45d2-bc09-81f4917ae152/
versions/88ae8699-12ac-4663-ba3f-4d7f0519140b",
  "tags": {}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSubscriptionDefinition](#)中的。

## get-thing-runtime-configuration

以下代码示例演示了如何使用 `get-thing-runtime-configuration`。

### AWS CLI

检索 Greengrass 内核的运行时配置

以下`get-thing-runtime-configuration`示例检索 Greengrass 内核的运行时配置。在检索运行时配置之前，必须使用`update-thing-runtime-configuration`命令为内核创建运行时配置。

```
aws greengrass get-thing-runtime-configuration \  
  --thing-name SampleGreengrassCore
```

输出：

```
{  
  "RuntimeConfiguration": {  
    "TelemetryConfiguration": {  
      "ConfigurationSyncStatus": "OutOfSync",  
      "Telemetry": "On"  
    }  
  }  
}
```

有关更多信息，请参阅 AWS IoT Greengrass 开发人员指南中的[配置遥测设置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetThingRuntimeConfiguration](#)中的。

## list-bulk-deployment-detailed-reports

以下代码示例演示了如何使用 `list-bulk-deployment-detailed-reports`。

### AWS CLI

列出有关批量部署中各个部署的信息

以下`list-bulk-deployment-detailed-reports`示例显示有关批量部署操作中各个部署的信息，包括状态。

```
aws greengrass list-bulk-deployment-detailed-reports \
  --bulk-deployment-id 42ce9c42-489b-4ed4-b905-8996aa50ef9d
```

输出：

```
{
  "Deployments": [
    {
      "DeploymentType": "NewDeployment",
      "DeploymentStatus": "Success",
      "DeploymentId": "123456789012:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "DeploymentArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/groups/a1b2c3d4-5678-90ab-cdef-EXAMPLE33333/
deployments/123456789012:123456789012:a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "GroupArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/groups/a1b2c3d4-5678-90ab-cdef-EXAMPLE33333/
versions/123456789012:a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
      "CreatedAt": "2020-01-21T21:34:16.501Z"
    },
    {
      "DeploymentType": "NewDeployment",
      "DeploymentStatus": "InProgress",
      "DeploymentId": "123456789012:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "DeploymentArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/groups/a1b2c3d4-5678-90ab-cdef-EXAMPLE55555/
deployments/123456789012:123456789012:a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "GroupArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
groups/a1b2c3d4-5678-90ab-cdef-EXAMPLE55555/versions/a1b2c3d4-5678-90ab-cdef-
EXAMPLE66666",
      "CreatedAt": "2020-01-21T21:34:16.486Z"
    },
    ...
  ]
}
```

有关更多信息，请参阅 AWS IoT Greengrass [开发人员指南中的为群组创建批量部署](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListBulkDeploymentDetailedReports](#) 中的。

## list-bulk-deployments

以下代码示例演示了如何使用 list-bulk-deployments。

## AWS CLI

### 列出批量部署

以下 `list-bulk-deployments` 示例列出了所有批量部署。

```
aws greengrass list-bulk-deployments
```

输出：

```
{
  "BulkDeployments": [
    {
      "BulkDeploymentArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/bulk/deployments/870fb41b-6288-4e0c-bc76-a7ba4b4d3267",
      "BulkDeploymentId": "870fb41b-6288-4e0c-bc76-a7ba4b4d3267",
      "CreatedAt": "2019-06-25T16:11:33.265Z"
    }
  ]
}
```

有关更多信息，请参阅 AWS IoT Greengrass [开发人员指南中的为群组创建批量部署](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListBulkDeployments](#) 中的。

## list-connector-definition-versions

以下代码示例演示了如何使用 `list-connector-definition-versions`。

### AWS CLI

列出可用于连接器定义的版本

以下 `list-connector-definition-versions` 示例列出了可用于指定连接器定义的版本。使用 `list-connector-definitions` 命令获取连接器定义 ID。

```
aws greengrass list-connector-definition-versions \
  --connector-definition-id "b5c4ebfd-f672-49a3-83cd-31c7216a7bb8"
```

输出：

```
{
  "Versions": [
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/connectors/b5c4ebfd-f672-49a3-83cd-31c7216a7bb8/versions/63c57963-
c7c2-4a26-a7e2-7bf478ea2623",
      "CreationTimestamp": "2019-06-19T19:30:01.300Z",
      "Id": "b5c4ebfd-f672-49a3-83cd-31c7216a7bb8",
      "Version": "63c57963-c7c2-4a26-a7e2-7bf478ea2623"
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网 Greengrass [开发者指南](#)》中的使用 [Greengrass 连接器与服务](#)和[协议集成](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListConnectorDefinitionVersions](#)中的。

## list-connector-definitions

以下代码示例演示了如何使用 list-connector-definitions。

### AWS CLI

列出已定义的 Greengrass 连接器

以下list-connector-definitions示例列出了为您的账户定义的所有 Greengrass 连接器。

AWS

```
aws greengrass list-connector-definitions
```

输出：

```
{
  "Definitions": [
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/connectors/b5c4ebfd-f672-49a3-83cd-31c7216a7bb8",
      "CreationTimestamp": "2019-06-19T19:30:01.300Z",
      "Id": "b5c4ebfd-f672-49a3-83cd-31c7216a7bb8",
      "LastUpdatedTimestamp": "2019-06-19T19:30:01.300Z",
```

```
        "LatestVersion": "63c57963-c7c2-4a26-a7e2-7bf478ea2623",
        "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/connectors/b5c4ebfd-f672-49a3-83cd-31c7216a7bb8/
versions/63c57963-c7c2-4a26-a7e2-7bf478ea2623",
        "Name": "MySNSConnector"
    }
]
}
```

有关更多信息，请参阅《AWS 物联网 Greengrass [开发者指南](#)》中的使用 [Greengrass 连接器与服务](#)和[协议集成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListConnectorDefinitions](#) 中的。

## list-core-definition-versions

以下代码示例演示了如何使用 `list-core-definition-versions`。

### AWS CLI

列出 Greengrass 核心定义的版本

以下 `list-core-definitions` 示例列出了指定 Greengrass 核心定义的所有版本。您可以使用 `list-core-definitions` 命令获取版本 ID。

```
aws greengrass list-core-definition-versions \
  --core-definition-id "eaf280cb-138c-4d15-af36-6f681a1348f7"
```

输出：

```
{
  "Versions": [
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/cores/eaf280cb-138c-4d15-af36-6f681a1348f7/versions/467c36e4-c5da-440c-
a97b-084e62593b4c",
      "CreationTimestamp": "2019-06-18T16:14:17.709Z",
      "Id": "eaf280cb-138c-4d15-af36-6f681a1348f7",
      "Version": "467c36e4-c5da-440c-a97b-084e62593b4c"
    }
  ]
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCoreDefinitionVersions](#)中的。

## list-core-definitions

以下代码示例演示了如何使用 list-core-definitions。

### AWS CLI

列出 Greengrass 的核心定义

以下list-core-definitions示例列出了您账户的所有 Greengrass 核心定义。AWS

```
aws greengrass list-core-definitions
```

输出：

```
{
  "Definitions": [
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/cores/0507843c-c1ef-4f06-b051-817030df7e7d",
      "CreationTimestamp": "2018-10-17T04:30:32.786Z",
      "Id": "0507843c-c1ef-4f06-b051-817030df7e7d",
      "LastUpdatedTimestamp": "2018-10-17T04:30:32.786Z",
      "LatestVersion": "bcdf9e86-3793-491e-93af-3cdfbf4e22b7",
      "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/cores/0507843c-c1ef-4f06-b051-817030df7e7d/versions/
bcdf9e86-3793-491e-93af-3cdfbf4e22b7"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/cores/31c22500-3509-4271-bafd-cf0655cda438",
      "CreationTimestamp": "2019-06-18T16:24:16.064Z",
      "Id": "31c22500-3509-4271-bafd-cf0655cda438",
      "LastUpdatedTimestamp": "2019-06-18T16:24:16.064Z",
      "LatestVersion": "2f350395-6d09-4c8a-8336-9ae5b57ace84",
      "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/cores/31c22500-3509-4271-bafd-cf0655cda438/
versions/2f350395-6d09-4c8a-8336-9ae5b57ace84"
    },
  ],
}
```

```

    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/cores/c906ed39-a1e3-4822-a981-7b9bd57b4b46",
      "CreationTimestamp": "2019-06-18T16:21:21.351Z",
      "Id": "c906ed39-a1e3-4822-a981-7b9bd57b4b46",
      "LastUpdatedTimestamp": "2019-06-18T16:21:21.351Z",
      "LatestVersion": "42aeac3-fd9d-4312-a8fd-ffa9404a20e0",
      "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/cores/c906ed39-a1e3-4822-a981-7b9bd57b4b46/versions/42aeac3-
fd9d-4312-a8fd-ffa9404a20e0"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/cores/eaf280cb-138c-4d15-af36-6f681a1348f7",
      "CreationTimestamp": "2019-06-18T16:14:17.709Z",
      "Id": "eaf280cb-138c-4d15-af36-6f681a1348f7",
      "LastUpdatedTimestamp": "2019-06-18T16:14:17.709Z",
      "LatestVersion": "467c36e4-c5da-440c-a97b-084e62593b4c",
      "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/cores/eaf280cb-138c-4d15-af36-6f681a1348f7/versions/467c36e4-
c5da-440c-a97b-084e62593b4c"
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCoreDefinitions](#)中的。

## list-deployments

以下代码示例演示了如何使用 list-deployments。

### AWS CLI

列出 Greengrass 群组的部署

以下list-deployments示例列出了指定 Greengrass 组的部署。您可以使用list-groups命令来查找您的群组 ID。

```
aws greengrass list-deployments \
  --group-id "1013db12-8b58-45ff-acc7-704248f66731"
```

输出：



```
{
  "Deployments": [
    {
      "CreatedAt": "2019-06-18T17:04:32.702Z",
      "DeploymentId": "1065b8a0-812b-4f21-9d5d-e89b232a530f",
      "DeploymentType": "NewDeployment",
      "GroupArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
groups/1013db12-8b58-45ff-acc7-704248f66731/versions/115136b3-cfd7-4462-
b77f-8741a4b00e5e"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDeployments](#)中的。

## list-device-definition-versions

以下代码示例演示了如何使用 `list-device-definition-versions`。

### AWS CLI

列出设备定义的版本

以下`list-device-definition-versions`示例显示了与指定设备定义关联的设备定义版本。

```
aws greengrass list-device-definition-versions \
  --device-definition-id "f9ba083d-5ad4-4534-9f86-026a45df1ccd"
```

输出：

```
{
  "Versions": [
    {
      "Version": "83c13984-6fed-447e-84d5-5b8aa45d5f71",
      "CreationTimestamp": "2019-09-11T00:15:09.838Z",
      "Id": "f9ba083d-5ad4-4534-9f86-026a45df1ccd",
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/devices/f9ba083d-5ad4-4534-9f86-026a45df1ccd/
versions/83c13984-6fed-447e-84d5-5b8aa45d5f71"
    },
    {
```

```

        "Version": "3b5cc510-58c1-44b5-9d98-4ad858ffa795",
        "CreationTimestamp": "2019-09-11T00:11:06.197Z",
        "Id": "f9ba083d-5ad4-4534-9f86-026a45df1ccd",
        "Arn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/devices/f9ba083d-5ad4-4534-9f86-026a45df1ccd/
versions/3b5cc510-58c1-44b5-9d98-4ad858ffa795"
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDeviceDefinitionVersions](#)中的。

## list-device-definitions

以下代码示例演示了如何使用 list-device-definitions。

### AWS CLI

列出您的设备定义

以下list-device-definitions示例显示了有关您 AWS 账户中指定 AWS 区域的设备定义的信息。

```
aws greengrass list-device-definitions \
  --region us-west-2
```

输出：

```

{
  "Definitions": [
    {
      "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/devices/50f3274c-3f0a-4f57-b114-6f46085281ab/versions/
c777b0f5-1059-449b-beaa-f003ebc56c34",
      "LastUpdatedTimestamp": "2019-06-14T15:42:09.059Z",
      "LatestVersion": "c777b0f5-1059-449b-beaa-f003ebc56c34",
      "CreationTimestamp": "2019-06-14T15:42:09.059Z",
      "Id": "50f3274c-3f0a-4f57-b114-6f46085281ab",
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/devices/50f3274c-3f0a-4f57-b114-6f46085281ab"
    },
    {

```

```

    "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/devices/e01951c9-6134-479a-969a-1a15cac11c40/
versions/514d57aa-4ee6-401c-9fac-938a9f7a51e5",
    "Name": "TestDeviceDefinition",
    "LastUpdatedTimestamp": "2019-04-16T23:17:43.245Z",
    "LatestVersion": "514d57aa-4ee6-401c-9fac-938a9f7a51e5",
    "CreationTimestamp": "2019-04-16T23:17:43.245Z",
    "Id": "e01951c9-6134-479a-969a-1a15cac11c40",
    "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/devices/e01951c9-6134-479a-969a-1a15cac11c40"
  },
  {
    "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/devices/f9ba083d-5ad4-4534-9f86-026a45df1ccd/
versions/83c13984-6fed-447e-84d5-5b8aa45d5f71",
    "Name": "TemperatureSensors",
    "LastUpdatedTimestamp": "2019-09-10T00:19:03.698Z",
    "LatestVersion": "83c13984-6fed-447e-84d5-5b8aa45d5f71",
    "CreationTimestamp": "2019-09-11T00:11:06.197Z",
    "Id": "f9ba083d-5ad4-4534-9f86-026a45df1ccd",
    "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/devices/f9ba083d-5ad4-4534-9f86-026a45df1ccd"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDeviceDefinitions](#)中的。

## list-function-definition-versions

以下代码示例演示了如何使用 `list-function-definition-versions`。

### AWS CLI

列出 Lambda 函数的版本

以下`list-function-definition-versions`示例列出了指定 Lambda 函数的所有版本。您可以使用`list-function-definitions`命令获取 ID。

```
aws greengrass list-function-definition-versions \
  --function-definition-id "063f5d1a-1dd1-40b4-9b51-56f8993d0f85"
```

输出：

```
{
  "Versions": [
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/functions/063f5d1a-1dd1-40b4-9b51-56f8993d0f85/versions/9748fda7-1589-4fcc-ac94-f5559e88678b",
      "CreationTimestamp": "2019-06-18T17:04:30.776Z",
      "Id": "063f5d1a-1dd1-40b4-9b51-56f8993d0f85",
      "Version": "9748fda7-1589-4fcc-ac94-f5559e88678b"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/functions/063f5d1a-1dd1-40b4-9b51-56f8993d0f85/versions/9b08df77-26f2-4c29-93d2-769715edcfec",
      "CreationTimestamp": "2019-06-18T17:02:44.087Z",
      "Id": "063f5d1a-1dd1-40b4-9b51-56f8993d0f85",
      "Version": "9b08df77-26f2-4c29-93d2-769715edcfec"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/functions/063f5d1a-1dd1-40b4-9b51-56f8993d0f85/versions/4236239f-94f7-4b90-a2f8-2a24c829d21e",
      "CreationTimestamp": "2019-06-18T17:01:42.284Z",
      "Id": "063f5d1a-1dd1-40b4-9b51-56f8993d0f85",
      "Version": "4236239f-94f7-4b90-a2f8-2a24c829d21e"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/functions/063f5d1a-1dd1-40b4-9b51-56f8993d0f85/versions/343408bb-549a-4fbe-b043-853643179a39",
      "CreationTimestamp": "2019-06-18T16:21:21.431Z",
      "Id": "063f5d1a-1dd1-40b4-9b51-56f8993d0f85",
      "Version": "343408bb-549a-4fbe-b043-853643179a39"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFunctionDefinitionVersions](#)中的。

## list-function-definitions

以下代码示例演示了如何使用 list-function-definitions。

## AWS CLI

### 列出 Lambda 函数

以下`list-function-definitions`示例列出了为您的 AWS 账户定义的所有 Lambda 函数。

```
aws greengrass list-function-definitions
```

输出：

```
{
  "Definitions": [
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/functions/017970a5-8952-46dd-b1c1-020b3ae8e960",
      "CreationTimestamp": "2018-10-17T04:30:32.884Z",
      "Id": "017970a5-8952-46dd-b1c1-020b3ae8e960",
      "LastUpdatedTimestamp": "2018-10-17T04:30:32.884Z",
      "LatestVersion": "4380b302-790d-4ed8-92bf-02e88afecb15",
      "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/functions/017970a5-8952-46dd-b1c1-020b3ae8e960/
versions/4380b302-790d-4ed8-92bf-02e88afecb15"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/functions/063f5d1a-1dd1-40b4-9b51-56f8993d0f85",
      "CreationTimestamp": "2019-06-18T16:21:21.431Z",
      "Id": "063f5d1a-1dd1-40b4-9b51-56f8993d0f85",
      "LastUpdatedTimestamp": "2019-06-18T16:21:21.431Z",
      "LatestVersion": "9748fda7-1589-4fcc-ac94-f5559e88678b",
      "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/functions/063f5d1a-1dd1-40b4-9b51-56f8993d0f85/
versions/9748fda7-1589-4fcc-ac94-f5559e88678b"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/functions/6598e653-a262-440c-9967-e2697f64da7b",
      "CreationTimestamp": "2019-06-18T16:24:16.123Z",
      "Id": "6598e653-a262-440c-9967-e2697f64da7b",
      "LastUpdatedTimestamp": "2019-06-18T16:24:16.123Z",
      "LatestVersion": "38bc6ccd-98a2-4ce7-997e-16c84748fae4",
```

```

    "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/functions/6598e653-a262-440c-9967-e2697f64da7b/
versions/38bc6ccd-98a2-4ce7-997e-16c84748fae4"
  },
  {
    "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/functions/c668df84-fad2-491b-95f4-655d2cad7885",
    "CreationTimestamp": "2019-06-18T16:14:17.784Z",
    "Id": "c668df84-fad2-491b-95f4-655d2cad7885",
    "LastUpdatedTimestamp": "2019-06-18T16:14:17.784Z",
    "LatestVersion": "37dd68c4-a64f-40ba-aa13-71fecc3ebded",
    "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/functions/c668df84-fad2-491b-95f4-655d2cad7885/
versions/37dd68c4-a64f-40ba-aa13-71fecc3ebded"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFunctionDefinitions](#)中的。

## list-group-certificate-authorities

以下代码示例演示了如何使用 `list-group-certificate-authorities`。

### AWS CLI

列出组的当前 CA

以下`list-group-certificate-authorities`示例列出了指定 Greengrass 组的当前证书颁发机构 (CA)。

```

aws greengrass list-group-certificate-authorities \
  --group-id "1013db12-8b58-45ff-acc7-704248f66731"

```

输出：

```

{
  "GroupCertificateAuthorities": [
    {
      "GroupCertificateAuthorityArn": "arn:aws:greengrass:us-
west-2:123456789012:/greengrass/groups/1013db12-8b58-45ff-acc7-704248f66731/

```

```
certificateauthorities/
f0430e1736ea8ed30cc5d5de9af67a7e3586bad9ae4d89c2a44163f65fdd8cf6",
    "GroupCertificateAuthorityId":
    "f0430e1736ea8ed30cc5d5de9af67a7e3586bad9ae4d89c2a44163f65fdd8cf6"
  }
]
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListGroupCertificateAuthorities](#) 中的。

## list-group-versions

以下代码示例演示了如何使用 `list-group-versions`。

### AWS CLI

列出 Greengrass 群组的版本

以下 `list-group-versions` 示例列出了指定 Greengrass 组的版本。

```
aws greengrass list-group-versions \
  --group-id "1013db12-8b58-45ff-acc7-704248f66731"
```

输出：

```
{
  "Versions": [
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
groups/1013db12-8b58-45ff-acc7-704248f66731/versions/115136b3-cfd7-4462-
b77f-8741a4b00e5e",
      "CreationTimestamp": "2019-06-18T17:04:30.915Z",
      "Id": "1013db12-8b58-45ff-acc7-704248f66731",
      "Version": "115136b3-cfd7-4462-b77f-8741a4b00e5e"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/groups/1013db12-8b58-45ff-acc7-704248f66731/versions/4340669d-
d14d-44e3-920c-46c928750750",
      "CreationTimestamp": "2019-06-18T17:03:52.663Z",
      "Id": "1013db12-8b58-45ff-acc7-704248f66731",
      "Version": "4340669d-d14d-44e3-920c-46c928750750"
    }
  ]
}
```

```
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/groups/1013db12-8b58-45ff-acc7-704248f66731/versions/1b06e099-2d5b-4f10-91b9-78c4e060f5da",
      "CreationTimestamp": "2019-06-18T17:02:44.189Z",
      "Id": "1013db12-8b58-45ff-acc7-704248f66731",
      "Version": "1b06e099-2d5b-4f10-91b9-78c4e060f5da"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/groups/1013db12-8b58-45ff-acc7-704248f66731/versions/2d3f27f1-3b43-4554-ab7a-73ec30477efe",
      "CreationTimestamp": "2019-06-18T17:01:42.401Z",
      "Id": "1013db12-8b58-45ff-acc7-704248f66731",
      "Version": "2d3f27f1-3b43-4554-ab7a-73ec30477efe"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/groups/1013db12-8b58-45ff-acc7-704248f66731/versions/d20f7ae9-3444-4c1c-b025-e2ede23cdd31",
      "CreationTimestamp": "2019-06-18T16:21:21.457Z",
      "Id": "1013db12-8b58-45ff-acc7-704248f66731",
      "Version": "d20f7ae9-3444-4c1c-b025-e2ede23cdd31"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListGroupVersions](#)中的。

## list-groups

以下代码示例演示了如何使用 list-groups。

### AWS CLI

列出 Greengrass 群组

以下list-groups示例列出了在您的账户中定义的所有 Greengrass 群组。AWS

```
aws greengrass list-groups
```

输出：



```

{
  "Groups": [
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
groups/1013db12-8b58-45ff-acc7-704248f66731",
      "CreationTimestamp": "2019-06-18T16:21:21.457Z",
      "Id": "1013db12-8b58-45ff-acc7-704248f66731",
      "LastUpdatedTimestamp": "2019-06-18T16:21:21.457Z",
      "LatestVersion": "115136b3-cfd7-4462-b77f-8741a4b00e5e",
      "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/groups/1013db12-8b58-45ff-acc7-704248f66731/versions/115136b3-cfd7-4462-
b77f-8741a4b00e5e",
      "Name": "GGGroup4Pi3"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
groups/1402daf9-71cf-4cfe-8be0-d5e80526d0d8",
      "CreationTimestamp": "2018-10-31T21:52:46.603Z",
      "Id": "1402daf9-71cf-4cfe-8be0-d5e80526d0d8",
      "LastUpdatedTimestamp": "2018-10-31T21:52:46.603Z",
      "LatestVersion": "749af901-60ab-456f-a096-91b12d983c29",
      "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/groups/1402daf9-71cf-4cfe-8be0-d5e80526d0d8/versions/749af901-60ab-456f-
a096-91b12d983c29",
      "Name": "MyTestGroup"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
groups/504b5c8d-bbed-4635-aff1-48ec5b586db5",
      "CreationTimestamp": "2018-12-31T21:39:36.771Z",
      "Id": "504b5c8d-bbed-4635-aff1-48ec5b586db5",
      "LastUpdatedTimestamp": "2018-12-31T21:39:36.771Z",
      "LatestVersion": "46911e8e-f9bc-4898-8b63-59c7653636ec",
      "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/groups/504b5c8d-bbed-4635-aff1-48ec5b586db5/versions/46911e8e-
f9bc-4898-8b63-59c7653636ec",
      "Name": "smp-ggrass-group"
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListGroups](#)中的。

## list-logger-definition-versions

以下代码示例演示了如何使用 `list-logger-definition-versions`。

### AWS CLI

获取记录器定义的版本列表

以下 `list-logger-definition-versions` 示例获取了指定记录器定义的所有版本的列表。

```
aws greengrass list-logger-definition-versions \
  --logger-definition-id "49eeeb66-f1d3-4e34-86e3-3617262abf23"
```

输出：

```
{
  "Versions": [
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/loggers/49eeeb66-f1d3-4e34-86e3-3617262abf23/versions/5e3f6f64-
a565-491e-8de0-3c0d8e0f2073",
      "CreationTimestamp": "2019-05-08T16:10:13.866Z",
      "Id": "49eeeb66-f1d3-4e34-86e3-3617262abf23",
      "Version": "5e3f6f64-a565-491e-8de0-3c0d8e0f2073"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/loggers/49eeeb66-f1d3-4e34-86e3-3617262abf23/versions/3ec6d3af-eb85-48f9-
a16d-1c795fe696d7",
      "CreationTimestamp": "2019-05-08T16:10:13.809Z",
      "Id": "49eeeb66-f1d3-4e34-86e3-3617262abf23",
      "Version": "3ec6d3af-eb85-48f9-a16d-1c795fe696d7"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListLoggerDefinitionVersions](#)中的。

## list-logger-definitions

以下代码示例演示了如何使用 `list-logger-definitions`。

## AWS CLI

### 获取记录器定义列表

以下 `list-logger-definitions` 示例列出了您 AWS 账户的所有记录器定义。

```
aws greengrass list-logger-definitions
```

输出：

```
{
  "Definitions": [
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/loggers/49eeeb66-f1d3-4e34-86e3-3617262abf23",
      "CreationTimestamp": "2019-05-08T16:10:13.809Z",
      "Id": "49eeeb66-f1d3-4e34-86e3-3617262abf23",
      "LastUpdatedTimestamp": "2019-05-08T16:10:13.809Z",
      "LatestVersion": "5e3f6f64-a565-491e-8de0-3c0d8e0f2073",
      "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/loggers/49eeeb66-f1d3-4e34-86e3-3617262abf23/
versions/5e3f6f64-a565-491e-8de0-3c0d8e0f2073"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListLoggerDefinitions](#) 中的。

## list-resource-definition-versions

以下代码示例演示了如何使用 `list-resource-definition-versions`。

## AWS CLI

### 列出资源定义的版本

以下 `list-resource-definition-versions` 示例列出了指定 Greengrass 资源的版本。

```
aws greengrass list-resource-definition-versions \
  --resource-definition-id "ad8c101d-8109-4b0e-b97d-9cc5802ab658"
```

输出：

```
{
  "Versions": [
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/resources/ad8c101d-8109-4b0e-b97d-9cc5802ab658/versions/26e8829a-491a-464d-9c87-664bf6f6f2be",
      "CreationTimestamp": "2019-06-19T16:40:59.392Z",
      "Id": "ad8c101d-8109-4b0e-b97d-9cc5802ab658",
      "Version": "26e8829a-491a-464d-9c87-664bf6f6f2be"
    },
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/resources/ad8c101d-8109-4b0e-b97d-9cc5802ab658/versions/432d92f6-12de-4ec9-a704-619a942a62aa",
      "CreationTimestamp": "2019-06-19T16:40:59.261Z",
      "Id": "ad8c101d-8109-4b0e-b97d-9cc5802ab658",
      "Version": "432d92f6-12de-4ec9-a704-619a942a62aa"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResourceDefinitionVersions](#)中的。

## list-resource-definitions

以下代码示例演示了如何使用 list-resource-definitions。

### AWS CLI

列出已定义的资源

以下list-resource-definitions示例列出了定义供 AWS 物联网 Greengrass 使用的资源。

```
aws greengrass list-resource-definitions
```

输出：

```
{
  "Definitions": [
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/resources/ad8c101d-8109-4b0e-b97d-9cc5802ab658",
```

```

    "CreationTimestamp": "2019-06-19T16:40:59.261Z",
    "Id": "ad8c101d-8109-4b0e-b97d-9cc5802ab658",
    "LastUpdatedTimestamp": "2019-06-19T16:40:59.261Z",
    "LatestVersion": "26e8829a-491a-464d-9c87-664bf6f6f2be",
    "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/resources/ad8c101d-8109-4b0e-b97d-9cc5802ab658/
versions/26e8829a-491a-464d-9c87-664bf6f6f2be"
  },
  {
    "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/resources/c8bb9ebc-c3fd-40a4-9c6a-568d75569d38",
    "CreationTimestamp": "2019-06-19T21:51:28.212Z",
    "Id": "c8bb9ebc-c3fd-40a4-9c6a-568d75569d38",
    "LastUpdatedTimestamp": "2019-06-19T21:51:28.212Z",
    "LatestVersion": "a5f94d0b-f6bc-40f4-bb78-7a1c5fe13ba1",
    "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/resources/c8bb9ebc-c3fd-40a4-9c6a-568d75569d38/versions/
a5f94d0b-f6bc-40f4-bb78-7a1c5fe13ba1",
    "Name": "MyGreengrassResources"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResourceDefinitions](#)中的。

## list-subscription-definition-versions

以下代码示例演示了如何使用 `list-subscription-definition-versions`。

### AWS CLI

列出订阅定义的版本

以下`list-subscription-definition-versions`示例列出了指定订阅的所有版本。您可以使用`list-subscription-definitions`命令来查找订阅 ID。

```
aws greengrass list-subscription-definition-versions \
  --subscription-definition-id "70e49321-83d5-45d2-bc09-81f4917ae152"
```

输出：

```
{
```

```
"Versions": [
  {
    "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/subscriptions/70e49321-83d5-45d2-bc09-81f4917ae152/versions/88ae8699-12ac-4663-ba3f-4d7f0519140b",
    "CreationTimestamp": "2019-06-18T17:03:52.499Z",
    "Id": "70e49321-83d5-45d2-bc09-81f4917ae152",
    "Version": "88ae8699-12ac-4663-ba3f-4d7f0519140b"
  },
  {
    "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/subscriptions/70e49321-83d5-45d2-bc09-81f4917ae152/versions/7e320ba3-c369-4069-a2f0-90acb7f219d6",
    "CreationTimestamp": "2019-06-18T17:03:52.392Z",
    "Id": "70e49321-83d5-45d2-bc09-81f4917ae152",
    "Version": "7e320ba3-c369-4069-a2f0-90acb7f219d6"
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSubscriptionDefinitionVersions](#)中的。

## list-subscription-definitions

以下代码示例演示了如何使用 list-subscription-definitions。

### AWS CLI

获取列表订阅定义

以下list-subscription-definitions示例列出了您的账户中定义的所有 AWS 物联网 Greengrass 订阅。AWS

```
aws greengrass list-subscription-definitions
```

输出：

```
{
  "Definitions": [
    {
      "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/subscriptions/70e49321-83d5-45d2-bc09-81f4917ae152",
```

```

    "CreationTimestamp": "2019-06-18T17:03:52.392Z",
    "Id": "70e49321-83d5-45d2-bc09-81f4917ae152",
    "LastUpdatedTimestamp": "2019-06-18T17:03:52.392Z",
    "LatestVersion": "88ae8699-12ac-4663-ba3f-4d7f0519140b",
    "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/subscriptions/70e49321-83d5-45d2-bc09-81f4917ae152/
versions/88ae8699-12ac-4663-ba3f-4d7f0519140b"
  },
  {
    "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/subscriptions/cd6f1c37-d9a4-4e90-be94-01a7404f5967",
    "CreationTimestamp": "2018-10-18T15:45:34.024Z",
    "Id": "cd6f1c37-d9a4-4e90-be94-01a7404f5967",
    "LastUpdatedTimestamp": "2018-10-18T15:45:34.024Z",
    "LatestVersion": "d1cf8fac-284f-4f6a-98fe-a2d36d089373",
    "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/subscriptions/cd6f1c37-d9a4-4e90-be94-01a7404f5967/versions/
d1cf8fac-284f-4f6a-98fe-a2d36d089373"
  },
  {
    "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/subscriptions/fa81bc84-3f59-4377-a84b-5d0134da359b",
    "CreationTimestamp": "2018-10-22T17:09:31.429Z",
    "Id": "fa81bc84-3f59-4377-a84b-5d0134da359b",
    "LastUpdatedTimestamp": "2018-10-22T17:09:31.429Z",
    "LatestVersion": "086d1b08-b25a-477c-a16f-6f9b3a9c295a",
    "LatestVersionArn": "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/subscriptions/fa81bc84-3f59-4377-a84b-5d0134da359b/
versions/086d1b08-b25a-477c-a16f-6f9b3a9c295a"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSubscriptionDefinitions](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 list-tags-for-resource。

### AWS CLI

列出附加到资源的标签

以下 `list-tags-for-resource` 示例列出了附加到指定资源的标签及其值。

```
aws greengrass list-tags-for-resource \  
  --resource-arn "arn:aws:greengrass:us-west-2:123456789012:/greengrass/  
definition/resources/ad8c101d-8109-4b0e-b97d-9cc5802ab658"
```

输出：

```
{  
  "tags": {  
    "ResourceSubType": "USB",  
    "ResourceType": "Device"  
  }  
}
```

有关更多信息，请参阅《物联网 Greengrass 开发者指南》中的“为 [你的 Greengrass 资源添加标签](#)”AWS。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## reset-deployments

以下代码示例演示了如何使用 `reset-deployments`。

### AWS CLI

清理 Greengrass 群组的部署信息

以下 `reset-deployments` 示例清理指定 Greengrass 组的部署信息。添加时 `--force` option，无需等待核心设备响应即可重置部署信息。

```
aws greengrass reset-deployments \  
  --group-id "1402daf9-71cf-4cfe-8be0-d5e80526d0d8" \  
  --force
```

输出：

```
{  
  "DeploymentArn": "arn:aws:greengrass:us-west-2:123456789012:/  
greengrass/groups/1402daf9-71cf-4cfe-8be0-d5e80526d0d8/  
deployments/7dd4e356-9882-46a3-9e28-6d21900c011a",  
  "DeploymentId": "7dd4e356-9882-46a3-9e28-6d21900c011a"
```



```
}
```

有关更多信息，请参阅 AWS IoT Greengrass 开发者指南中的[重置部署](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ResetDeployments](#)中的。

## start-bulk-deployment

以下代码示例演示了如何使用 start-bulk-deployment。

### AWS CLI

#### 启动批量部署操作

以下 start-bulk-deployment 示例启动批量部署操作，使用存储在 S3 存储桶中的文件来指定要部署的组。

```
aws greengrass start-bulk-deployment \
  --cli-input-json "{\"InputFileUri\": \"https://gg-group-deployment1.s3-us-west-2.amazonaws.com/MyBulkDeploymentInputFile.txt\", \"ExecutionRoleArn\": \"arn:aws:iam::123456789012:role/ggCreateDeploymentRole\", \"AmznClientToken\": \"yourAmazonClientToken\"}"
```

输出：

```
{
  "BulkDeploymentArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/bulk/deployments/870fb41b-6288-4e0c-bc76-a7ba4b4d3267",
  "BulkDeploymentId": "870fb41b-6288-4e0c-bc76-a7ba4b4d3267"
}
```

有关更多信息，请参阅 AWS IoT Greengrass [开发人员指南中的为群组创建批量部署](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[StartBulkDeployment](#)中的。

## stop-bulk-deployment

以下代码示例演示了如何使用 stop-bulk-deployment。

### AWS CLI

#### 停止批量部署

以下 `stop-bulk-deployment` 示例停止指定的批量部署。如果您尝试停止已完成的批量部署，则会收到一条错误消息：`InvalidInputException: Cannot change state of finished execution.`

```
aws greengrass stop-bulk-deployment \  
  --bulk-deployment-id "870fb41b-6288-4e0c-bc76-a7ba4b4d3267"
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT Greengrass [开发人员指南中的为群组创建批量部署](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StopBulkDeployment](#) 中的。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

### AWS CLI

对资源应用标签

以下 `tag-resource` 示例将两个标签 `ResourceType` 和 `ResourceSubType` 应用于指定的 Greengrass 资源。此操作既可以添加新的标签和值，也可以更新现有标签的值。使用 `untag-resource` 命令移除标记。

```
aws greengrass tag-resource \  
  --resource-arn "arn:aws:greengrass:us-west-2:123456789012:/greengrass/  
definition/resources/ad8c101d-8109-4b0e-b97d-9cc5802ab658" \  
  --tags "ResourceType=Device,ResourceSubType=USB"
```

此命令不生成任何输出。

有关更多信息，请参阅《物联网 Greengrass 开发者指南》中的“为 [你的 Greengrass 资源添加标签](#)”AWS。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 `untag-resource`。

## AWS CLI

### 从资源中移除标签及其值

以下 `untag-resource` 示例删除了密钥 `Category` 来自指定 Greengrass 组的标签。如果指定资源的密钥 `Category` 不存在，则不会返回任何错误。

```
aws greengrass untag-resource \  
  --resource-arn "arn:aws:greengrass:us-west-2:123456789012:/greengrass/  
groups/1013db12-8b58-45ff-acc7-704248f66731" \  
  --tag-keys "Category"
```

此命令不生成任何输出。

有关更多信息，请参阅《物联网 Greengrass 开发者指南》中的“为[你的 Greengrass 资源添加标签](#)”AWS。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-connectivity-info

以下代码示例演示了如何使用 `update-connectivity-info`。

### AWS CLI

#### 更新 Greengrass 核心的连接信息

以下 `update-connectivity-info` 示例更改了设备可用于连接到指定 Greengrass 核心的端点。连接信息是 IP 地址或域名的列表，以及相应的端口号和可选的客户定义元数据。当本地网络发生变化时，您可能需要更新连接信息。

```
aws greengrass update-connectivity-info \  
  --thing-name "MyGroup_Core" \  
  --connectivity-info "[{\"Metadata\":\"\",\"PortNumber\":8883,\"HostAddress\  
\":\"127.0.0.1\",\"Id\":\"localhost_127.0.0.1_0\"},{\"Metadata\":\"\",\"PortNumber  
\":\"8883,\"HostAddress\":\"192.168.1.3\",\"Id\":\"localIP_192.168.1.3\"}]"
```

输出：

```
{  
  "Version": "312de337-59af-4cf9-a278-2a23bd39c300"
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateConnectivityInfo](#)中的。

## update-connector-definition

以下代码示例演示了如何使用 update-connector-definition。

### AWS CLI

#### 更新连接器定义的名称

以下update-connector-definition示例更新了指定连接器定义的名称。如果要更新连接器的详细信息，请使用create-connector-definition-version命令创建新版本。

```
aws greengrass update-connector-definition \  
  --connector-definition-id "55d0052b-0d7d-44d6-b56f-21867215e118" \  
  --name "GreengrassConnectors2019"
```

有关更多信息，请参阅 AWS IoT Greengrass [开发人员指南中的使用连接器与服务 and 协议集成](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateConnectorDefinition](#)中的。

## update-core-definition

以下代码示例演示了如何使用 update-core-definition。

### AWS CLI

#### 更新核心定义

以下update-core-definition示例更改了指定核心定义的名称。您只能更新核心定义的名称属性。

```
aws greengrass update-core-definition \  
  --core-definition-id "582efe12-b05a-409e-9a24-a2ba1bcc4a12" \  
  --name "MyCoreDevices"
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS 物联网 Greengrass 开发者指南》中的配置物联网 Greengrass 核心AWS 心](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateCoreDefinition](#)中的。

## update-device-definition

以下代码示例演示了如何使用 `update-device-definition`。

### AWS CLI

#### 更新设备定义

以下`update-device-definition`示例更改了指定设备定义的名称。您只能更新设备定义的`name`属性。

```
aws greengrass update-device-definition \  
  --device-definition-id "f9ba083d-5ad4-4534-9f86-026a45df1ccd" \  
  --name "TemperatureSensors"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDeviceDefinition](#)中的。

## update-function-definition

以下代码示例演示了如何使用 `update-function-definition`。

### AWS CLI

#### 更新函数定义的名称

以下`update-function-definition`示例更新了指定函数定义的名称。如果要更新函数的详细信息，请使用`create-function-definition-version`命令创建新版本。

```
aws greengrass update-function-definition \  
  --function-definition-id "e47952bd-dea9-4e2c-a7e1-37bbe8807f46" \  
  --name ObsoleteFunction
```

此命令不生成任何输出。

有关更多信息，请参阅 IoT AWS Greengrass 开发人员指南中的[运行本地 Lambda 函数](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateFunctionDefinition](#)中的。

## update-group-certificate-configuration

以下代码示例演示了如何使用 `update-group-certificate-configuration`。

### AWS CLI

#### 更新群组证书的到期时间

以下 `update-group-certificate-configuration` 示例为为指定组生成的证书设置了 10 天的有效期。

```
aws greengrass update-group-certificate-configuration \  
  --group-id "8eaadd72-ce4b-4f15-892a-0cc4f3a343f1" \  
  --certificate-expiry-in-milliseconds 864000000
```

输出：

```
{  
  "CertificateExpiryInMilliseconds": 864000000,  
  "CertificateAuthorityExpiryInMilliseconds": 2524607999000,  
  "GroupId": "8eaadd72-ce4b-4f15-892a-0cc4f3a343f1"  
}
```

有关更多信息，请参阅《[AWS 物联网 Greengrass 开发者指南](#)》中的 [Io AWS T Greengrass 安全](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateGroupCertificateConfiguration](#) 中的。

## update-group

以下代码示例演示了如何使用 `update-group`。

### AWS CLI

#### 更新群组名称

以下 `update-group` 示例更新了指定 Greengrass 组的名称。如果要更新群组的详细信息，请使用 `create-group-version` 命令创建新版本。

```
aws greengrass update-group \  
  --group-id "1402daf9-71cf-4cfe-8be0-d5e80526d0d8" \  
  --group-name "New Group Name"
```

```
--name TestGroup4of6
```

有关更多信息，请参阅《[AWS 物联网 Greengrass 开发者指南](#)》中的在物联网上配置 [AWS IoT Greengrass](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateGroup](#)中的。

## update-logger-definition

以下代码示例演示了如何使用 update-logger-definition。

### AWS CLI

#### 更新记录器定义

以下update-logger-definition示例更改了指定记录器定义的名称。您只能更新记录器定义的name属性。

```
aws greengrass update-logger-definition \  
  --logger-definition-id "a454b62a-5d56-4ca9-bdc4-8254e1662cb0" \  
  --name "LoggingConfigsForSensors"
```

此命令不生成任何输出。

有关更多信息，请参阅《[物联网 Greengrass 开发者指南](#)》中的使用 [AWS 物联网 Greengrass 日志进行AWS 监控](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLoggerDefinition](#)中的。

## update-resource-definition

以下代码示例演示了如何使用 update-resource-definition。

### AWS CLI

#### 更新资源定义的名称

以下update-resource-definition示例更新了指定资源定义的名称。如果要更改资源的详细信息，请使用create-resource-definition-version命令创建新版本。

```
aws greengrass update-resource-definition \  
  --name "NewResourceDefinitionName"
```

```
--resource-definition-id "c8bb9ebc-c3fd-40a4-9c6a-568d75569d38" \  
--name GreengrassConnectorResources
```

此命令不生成任何输出。

有关更多信息，请参阅 IoT AWS Greengrass [s 开发人员指南中的使用 Lambda 函数和连接器访问本地资源](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateResourceDefinition](#) 中的。

## update-subscription-definition

以下代码示例演示了如何使用 update-subscription-definition。

### AWS CLI

更新订阅定义的名称

以下 update-subscription-definition 示例更新了指定订阅定义的名称。如果要更改订阅的详细信息，请使用 create-subscription-definition-version 命令创建新版本。

```
aws greengrass update-subscription-definition \  
--subscription-definition-id "fa81bc84-3f59-4377-a84b-5d0134da359b" \  
--name "ObsoleteSubscription"
```

此命令不生成任何输出。

有关更多信息，请参阅指南中的标题。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateSubscriptionDefinition](#) 中的。

## update-thing-runtime-configuration

以下代码示例演示了如何使用 update-thing-runtime-configuration。

### AWS CLI

在 Greengrass 内核的运行时配置中开启遥测功能

以下 update-thing-runtime-configuration 示例更新了 Greengrass 内核的运行时配置以开启遥测功能。



```
aws greengrass update-thing-runtime-configuration \  
  --thing-name SampleGreengrassCore \  
  --telemetry-configuration {"Telemetry\":"\0n\"}
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT Greengrass 开发人员指南中的[配置遥测设置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateThingRuntimeConfiguration](#)中的。

## AWS IoT Greengrass V2 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS IoT Greengrass V2。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-service-role-to-account**

以下代码示例演示了如何使用 `associate-service-role-to-account`。

AWS CLI

将 Greengrass 服务角色与您的账户关联 AWS

以下 `associate-service-role-to-account` 示例将您的账户的服务角色与 AWS IoT Greengrass 关联起来。AWS

```
aws greengrassv2 associate-service-role-to-account \  
  --service-role-name SampleGreengrassCoreRole \  
  --account-id 123456789012
```

```
--role-arn arn:aws:iam::123456789012:role/service-role/Greengrass_ServiceRole
```

输出：

```
{
  "associatedAt": "2022-01-19T19:21:53Z"
}
```

有关更多信息，请参阅《物AWS 联网 [Greengrass V2 开发者指南](#)》中的 [Greengrass 服务角色](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateServiceRoleToAccount](#)中的。

## batch-associate-client-device-with-core-device

以下代码示例演示了如何使用 batch-associate-client-device-with-core-device。

AWS CLI

将客户端设备与核心设备关联

以下batch-associate-client-device-with-core-device示例将两台客户端设备与一台核心设备关联起来。

```
aws greengrassv2 batch-associate-client-device-with-core-device \
  --core-device-thing-name MyGreengrassCore \
  --entries thingName=MyClientDevice1 thingName=MyClientDevice2
```

输出：

```
{
  "errorEntries": []
}
```

有关更多信息，请参阅 [IoT Greengrass V2 开发者指南](#)中的[与本地AWS 物联网设备交互](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchAssociateClientDeviceWithCoreDevice](#)中的。

## batch-disassociate-client-device-from-core-device

以下代码示例演示了如何使用 batch-disassociate-client-device-from-core-device。

## AWS CLI

### 取消客户端设备与核心设备的关联

以下batch-disassociate-client-device-from-core-device示例取消两台客户端设备与核心设备的关联。

```
aws greengrassv2 batch-disassociate-client-device-from-core-device \  
  --core-device-thing-name MyGreengrassCore \  
  --entries thingName=MyClientDevice1 thingName=MyClientDevice2
```

输出：

```
{  
  "errorEntries": []  
}
```

有关更多信息，请参阅 [IoT Greengrass V2 开发者指南中的与本地AWS 物联网设备交互](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考 [BatchDisassociateClientDeviceFromCoreDevice](#) 中的。

## cancel-deployment

以下代码示例演示了如何使用 cancel-deployment。

### AWS CLI

#### 取消部署

以下cancel-deployment示例停止了对事物组的持续部署。

```
aws greengrassv2 cancel-deployment \  
  --deployment-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "message": "SUCCESS"  
}
```

有关更多信息，请参阅 AWS IoT Greengrass V2 开发者指南中的[取消部署](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CancelDeployment](#)中的。

## create-component-version

以下代码示例演示了如何使用 create-component-version。

### AWS CLI

示例 1：根据配方创建组件版本

以下 create-component-version 示例根据配方文件创建 Hello World 组件的版本。

```
aws greengrassv2 create-component-version \  
  --inline-recipe fileb://com.example.HelloWorld-1.0.0.json
```

com.example.HelloWorld-1.0.0.json 的内容：

```
{  
  "RecipeFormatVersion": "2020-01-25",  
  "ComponentName": "com.example.HelloWorld",  
  "ComponentVersion": "1.0.0",  
  "ComponentDescription": "My first AWS IoT Greengrass component.",  
  "ComponentPublisher": "Amazon",  
  "ComponentConfiguration": {  
    "DefaultConfiguration": {  
      "Message": "world"  
    }  
  },  
  "Manifests": [  
    {  
      "Platform": {  
        "os": "linux"  
      },  
      "Lifecycle": {  
        "Run": "echo 'Hello {configuration:/Message}'"  
      }  
    }  
  ]  
}
```

输出：

```
{
  "arn": "arn:aws:greengrass:us-
west-2:123456789012:components:com.example.HelloWorld:versions:1.0.0",
  "componentName": "com.example.HelloWorld",
  "componentVersion": "1.0.0",
  "creationTimestamp": "2021-01-07T16:24:33.650000-08:00",
  "status": {
    "componentState": "REQUESTED",
    "message": "NONE",
    "errors": {}
  }
}
```

有关更多信息，请参阅《AWS IoT Greengrass V2 开发者指南》中的[创建自定义组件和上传要部署的组件](#)。

示例 2：通过 AWS Lambda 函数创建组件版本

以下 `create-component-version` 示例通过 AWS Lambda 函数创建 Hello World 组件的一个版本。

```
aws greengrassv2 create-component-version \
  --cli-input-json file://lambda-function-component.json
```

`lambda-function-component.json` 的内容：

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:us-
west-2:123456789012:function:HelloWorldPythonLambda:1",
    "componentName": "com.example.HelloWorld",
    "componentVersion": "1.0.0",
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ]
    }
  }
}
```

输出：

```
{
  "arn": "arn:aws:greengrass:us-west-2:123456789012:components:com.example.HelloWorld:versions:1.0.0",
  "componentName": "com.example.HelloWorld",
  "componentVersion": "1.0.0",
  "creationTimestamp": "2021-01-07T17:05:27.347000-08:00",
  "status": {
    "componentState": "REQUESTED",
    "message": "NONE",
    "errors": {}
  }
}
```

有关更多信息，请参阅物联网 AWS Greengrass AWS [V2 开发人员指南中的运行 Lambda 函数](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateComponentVersion](#) 中的。

## create-deployment

以下代码示例演示了如何使用 create-deployment。

### AWS CLI

#### 示例 1：创建部署

以下 create-deployment 示例将 AWS IoT Greengrass 命令行接口部署到核心设备。

```
aws greengrassv2 create-deployment \
  --cli-input-json file://cli-deployment.json
```

cli-deployment.json 的内容：

```
{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "deploymentName": "Deployment for MyGreengrassCore",
  "components": {
    "aws.greengrass.Cli": {
      "componentVersion": "2.0.3"
    }
  },
}
```

```

"deploymentPolicies": {
  "failureHandlingPolicy": "DO_NOTHING",
  "componentUpdatePolicy": {
    "timeoutInSeconds": 60,
    "action": "NOTIFY_COMPONENTS"
  },
  "configurationValidationPolicy": {
    "timeoutInSeconds": 60
  }
},
"iotJobConfiguration": {}
}

```

输出：

```

{
  "deploymentId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}

```

有关更多信息，请参阅 AWS IoT Greengrass V2 开发人员指南中的[创建部署](#)。

#### 示例 2：创建更新组件配置的部署

以下 create-deployment 示例将 AWS IoT Greengrass 核心组件部署到一组核心设备。此部署对 nucleus 组件应用了以下配置更新：

将目标设备的代理设置重置为默认的无代理设置。将目标设备的 MQTT 设置重置为默认值。为 nucleus 的 jvm 设置 JVM 选项。设置 nucleus 的日志级别。

```

aws greengrassv2 create-deployment \
  --cli-input-json file://nucleus-deployment.json

```

nucleus-deployment.json 的内容：

```

{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/MyGreengrassCoreGroup",
  "deploymentName": "Deployment for MyGreengrassCoreGroup",
  "components": {
    "aws.greengrass.Nucleus": {
      "componentVersion": "2.0.3",

```

```

        "configurationUpdate": {
            "reset": [
                "/networkProxy",
                "/mqtt"
            ],
            "merge": "{\"jvmOptions\": \"-Xmx64m\", \"logging\": {\"level\": \"WARN
\\\"}}\"
        }
    },
    "deploymentPolicies": {
        "failureHandlingPolicy": "ROLLBACK",
        "componentUpdatePolicy": {
            "timeoutInSeconds": 60,
            "action": "NOTIFY_COMPONENTS"
        },
        "configurationValidationPolicy": {
            "timeoutInSeconds": 60
        }
    },
    "iotJobConfiguration": {}
}

```

输出：

```

{
  "deploymentId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "iotJobId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "iotJobArn": "arn:aws:iot:us-west-2:123456789012:job/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}

```

有关更多信息，请参阅 [AWS IoT Greengrass V2 开发人员指南中的创建部署和更新组件配置](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 CreateDeployment](#) 中的。

## delete-component

以下代码示例演示了如何使用 delete-component。

### AWS CLI

#### 删除组件版本



以下delete-component示例删除了 Hello World 组件。

```
aws greengrassv2 delete-component \  
    --arn arn:aws:greengrass:us-  
west-2:123456789012:components:com.example>HelloWorld:versions:1.0.0
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 Greengrass V2 开发者指南》中的[管理组件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteComponent](#)中的。

## delete-core-device

以下代码示例演示了如何使用 delete-core-device。

### AWS CLI

删除核心设备

以下delete-core-device示例删除了 AWS 物联网 Greengrass 核心设备。

```
aws greengrassv2 delete-core-device \  
    --core-device-thing-name MyGreengrassCore
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网 Greengrass V2 开发者指南](#)》中的[卸载 lo AWS T Greengrass Core](#) 软件。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCoreDevice](#)中的。

## describe-component

以下代码示例演示了如何使用 describe-component。

### AWS CLI

描述组件版本

以下describe-component示例描述了 Hello World 组件。

```
aws greengrassv2 describe-component \  
  --arn arn:aws:greengrass:us-  
west-2:123456789012:components:com.example.HelloWorld:versions:1.0.0
```

输出：

```
{  
  "arn": "arn:aws:greengrass:us-  
west-2:123456789012:components:com.example.HelloWorld:versions:1.0.0",  
  "componentName": "com.example.HelloWorld",  
  "componentVersion": "1.0.0",  
  "creationTimestamp": "2021-01-07T17:12:11.133000-08:00",  
  "publisher": "Amazon",  
  "description": "My first AWS IoT Greengrass component.",  
  "status": {  
    "componentState": "DEPLOYABLE",  
    "message": "NONE",  
    "errors": {}  
  },  
  "platforms": [  
    {  
      "attributes": {  
        "os": "linux"  
      }  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS 物联网 Greengrass V2 开发者指南》中的[管理组件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeComponent](#) 中的。

## disassociate-service-role-from-account

以下代码示例演示了如何使用 disassociate-service-role-from-account。

### AWS CLI

解除 Greengrass 服务角色与您的账户的关联 AWS

以下 disassociate-service-role-from-account 示例取消您账户的 Greengrass 服务角色与 AWS IoT Greengrass 的关联。AWS

```
aws greengrassv2 disassociate-service-role-from-account
```

输出：

```
{
  "disassociatedAt": "2022-01-19T19:26:09Z"
}
```

有关更多信息，请参阅《物AWS 联网 [Greengrass V2 开发者指南](#)》中的 [Greengrass 服务角色](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateServiceRoleFromAccount](#)中的。

## get-component-version-artifact

以下代码示例演示了如何使用 `get-component-version-artifact`。

### AWS CLI

获取用于下载组件工件的 URL

以下`get-component-version-artifact`示例获取了下载本地调试控制台组件的 JAR 文件的 URL。

```
aws greengrassv2 get-component-version-artifact \
  --arn arn:aws:greengrass:us-
west-2:aws:components:aws.greengrass.LocalDebugConsole:versions:2.0.3 \
  --artifact-name "Uvt6ZEzQ9TKiAuLbfXBX_APdY0TWks3uc46tHFHTzBM=/
aws.greengrass.LocalDebugConsole.jar"
```

输出：

```
{
  "preSignedUrl": "https://evergreencomponentmanageme-
artifactbucket7410c9ef-g18n1iya8kwr.s3.us-west-2.amazonaws.com/public/
aws.greengrass.LocalDebugConsole/2.0.3/s3/ggv2-component-releases-prod-pdx/
EvergreenHttpDebugView/2ffc496ba41b39568968b22c582b4714a937193ee7687a45527238e696672521/
aws.greengrass.LocalDebugConsole/aws.greengrass.LocalDebugConsole.jar?X-Amz-
Security-Token=KwflKSdEXAMPLE..."
}
```

有关更多信息，请参阅《AWS 物联网 Greengrass V2 开发者指南》中的[管理组件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetComponentVersionArtifact](#)中的。

## get-component

以下代码示例演示了如何使用 get-component。

### AWS CLI

示例 1：以 YAML 格式 (Linux、macOS 或 Unix) 下载组件的配方

以下get-component示例将 Hello World 组件的配方下载到 YAML 格式的文件中。此命令执行以下操作：

使用--output和--query参数来控制命令的输出。这些参数从命令的输出中提取配方 blob。有关控制输出的更多信息，请参阅《[命令行界面用户指南](#)》中的[控制AWS 命令输出](#)。使用该base64实用程序。此实用程序将提取的 blob 解码为原始文本。成功的get-component命令返回的 blob 是 base64 编码的文本。必须解码此 blob 才能获得原始文本。将解码后的文本保存到文件中。命令的最后一部分 (> com.example.HelloWorld-1.0.0.json) 将解码后的文本保存到文件中。

```
aws greengrassv2 get-component \  
  --arn arn:aws:greengrass:us-  
west-2:123456789012:components:com.example.HelloWorld:versions:1.0.0 \  
  --recipe-output-format YAML \  
  --query recipe \  
  --output text | base64 --decode > com.example.HelloWorld-1.0.0.json
```

有关更多信息，请参阅《AWS 物联网 Greengrass V2 开发者指南》中的[管理组件](#)。

示例 2：以 YAML 格式下载组件的配方 (Windows CMD)

以下get-component示例将 Hello World 组件的配方下载到 YAML 格式的文件中。此命令使用该certutil实用程序。

```
aws greengrassv2 get-component ^  
  --arn arn:aws:greengrass:us-  
west-2:675946970638:components:com.example.HelloWorld:versions:1.0.0 ^  
  --recipe-output-format YAML ^  
  --query recipe ^  
  --output text > com.example.HelloWorld-1.0.0.yaml.b64
```

```
certutil -decode com.example.HelloWorld-1.0.0.yaml.b64
com.example.HelloWorld-1.0.0.yaml
```

有关更多信息，请参阅《AWS 物联网 Greengrass V2 开发者指南》中的[管理组件](#)。

示例 3：以 YAML 格式下载组件的配方 (Windows PowerShell)

以下 `get-component` 示例将 Hello World 组件的配方下载到 YAML 格式的文件中。此命令使用该 `certutil` 实用程序。

```
aws greengrassv2 get-component `
  --arn arn:aws:greengrass:us-
west-2:675946970638:components:com.example.HelloWorld:versions:1.0.0 `
  --recipe-output-format YAML `
  --query recipe `
  --output text > com.example.HelloWorld-1.0.0.yaml.b64

certutil -decode com.example.HelloWorld-1.0.0.yaml.b64
com.example.HelloWorld-1.0.0.yaml
```

有关更多信息，请参阅《AWS 物联网 Greengrass V2 开发者指南》中的[管理组件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetComponent](#) 中的。

## get-connectivity-info

以下代码示例演示了如何使用 `get-connectivity-info`。

### AWS CLI

获取 Greengrass 核心设备的连接信息

以下 `get-connectivity-info` 示例获取了 Greengrass 核心设备的连接信息。客户端设备使用此信息连接到在此核心设备上运行的 MQTT 代理。

```
aws greengrassv2 get-connectivity-info \
  --thing-name MyGreengrassCore
```

输出：

```
{
  "connectivityInfo": [
    {
```

```
        "id": "localIP_192.0.2.0",
        "hostAddress": "192.0.2.0",
        "portNumber": 8883
      }
    ]
  }
```

有关更多信息，请参阅 AWS IoT Greengrass V2 开发者指南中的[管理核心设备端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetConnectivityInfo](#)中的。

## get-core-device

以下代码示例演示了如何使用 get-core-device。

### AWS CLI

#### 获取核心设备

以下 get-core-device 示例获取有关 AWS 物联网 Greengrass 核心设备的信息。

```
aws greengrassv2 get-core-device \
  --core-device-thing-name MyGreengrassCore
```

#### 输出：

```
{
  "coreDeviceThingName": "MyGreengrassCore",
  "coreVersion": "2.0.3",
  "platform": "linux",
  "architecture": "amd64",
  "status": "HEALTHY",
  "lastStatusUpdateTimestamp": "2021-01-08T04:57:58.838000-08:00",
  "tags": {}
}
```

有关更多信息，请参阅 AWS IoT Greengrass V2 开发者指南中的[检查核心设备状态](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetCoreDevice](#)中的。

## get-deployment

以下代码示例演示了如何使用 get-deployment。

## AWS CLI

### 要进行部署

以下`get-deployment`示例获取有关将 AWS 物联网 Greengrass nucleus 组件部署到一组核心设备的信息。

```
aws greengrassv2 get-deployment \  
  --deployment-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

### 输出：

```
{  
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/  
MyGreengrassCoreGroup",  
  "revisionId": "14",  
  "deploymentId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
  "deploymentName": "Deployment for MyGreengrassCoreGroup",  
  "deploymentStatus": "ACTIVE",  
  "iotJobId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
  "iotJobArn": "arn:aws:iot:us-west-2:123456789012:job/a1b2c3d4-5678-90ab-cdef-  
EXAMPLE22222",  
  "components": {  
    "aws.greengrass.Nucleus": {  
      "componentVersion": "2.0.3",  
      "configurationUpdate": {  
        "merge": "{\"jvmOptions\": \"-Xmx64m\", \"logging\": {\"level\": \"WARN  
\"}}\",  
        "reset": [  
          "/networkProxy",  
          "/mqtt"  
        ]  
      }  
    }  
  },  
  "deploymentPolicies": {  
    "failureHandlingPolicy": "ROLLBACK",  
    "componentUpdatePolicy": {  
      "timeoutInSeconds": 60,  
      "action": "NOTIFY_COMPONENTS"  
    },  
    "configurationValidationPolicy": {  
      "timeoutInSeconds": 60
```

```
    }  
  },  
  "iotJobConfiguration": {},  
  "creationTimestamp": "2021-01-07T17:21:20.691000-08:00",  
  "isLatestForTarget": false,  
  "tags": {}  
}
```

有关更多信息，请参阅 AWS IoT Greengrass V2 开发人员指南中的[将组件部署到设备](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetDeployment](#)中的。

## get-service-role-for-account

以下代码示例演示了如何使用 `get-service-role-for-account`。

### AWS CLI

为你的账户获取 Greengrass 服务角色 AWS

以下 `get-service-role-for-account` 示例为您的账户获取与 AWS IoT Greengrass 关联的服务角色。AWS

```
aws greengrassv2 get-service-role-for-account
```

输出：

```
{  
  "associatedAt": "2022-01-19T19:21:53Z",  
  "roleArn": "arn:aws:iam::123456789012:role/service-role/Greengrass_ServiceRole"  
}
```

有关更多信息，请参阅《物AWS 联网 [Greengrass V2 开发者指南](#)》中的 [Greengrass 服务角色](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetServiceRoleForAccount](#)中的。

## list-client-devices-associated-with-core-device

以下代码示例演示了如何使用 `list-client-devices-associated-with-core-device`。

### AWS CLI

列出与核心设备关联的客户端设备



以下`list-client-devices-associated-with-core-device`示例列出了与核心设备关联的所有客户端设备。

```
aws greengrassv2 list-client-devices-associated-with-core-device \  
  --core-device-thing-name MyTestGreengrassCore
```

输出：

```
{  
  "associatedClientDevices": [  
    {  
      "thingName": "MyClientDevice2",  
      "associationTimestamp": "2021-07-12T16:33:55.843000-07:00"  
    },  
    {  
      "thingName": "MyClientDevice1",  
      "associationTimestamp": "2021-07-12T16:33:55.843000-07:00"  
    }  
  ]  
}
```

有关更多信息，请参阅 [IoT Greengrass V2 开发者指南中的与本地AWS 物联网设备交互](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListClientDevicesAssociatedWithCoreDevice](#)中的。

## list-component-versions

以下代码示例演示了如何使用 `list-component-versions`。

AWS CLI

列出组件的版本

以下`list-component-versions`示例列出了 Hello World 组件的所有版本。

```
aws greengrassv2 list-component-versions \  
  --arn arn:aws:greengrass:us-  
west-2:123456789012:components:com.example.HelloWorld
```

输出：

```
{
  "componentVersions": [
    {
      "componentName": "com.example.HelloWorld",
      "componentVersion": "1.0.1",
      "arn": "arn:aws:greengrass:us-west-2:123456789012:components:com.example.HelloWorld:versions:1.0.1"
    },
    {
      "componentName": "com.example.HelloWorld",
      "componentVersion": "1.0.0",
      "arn": "arn:aws:greengrass:us-west-2:123456789012:components:com.example.HelloWorld:versions:1.0.0"
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网 Greengrass V2 开发者指南》中的[管理组件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListComponentVersions](#)中的。

## list-components

以下代码示例演示了如何使用 list-components。

### AWS CLI

#### 列出组件

以下 list-components 示例列出了您在当前区域的 AWS 账户中定义的每个组件及其最新版本。

```
aws greengrassv2 list-components
```

#### 输出：

```
{
  "components": [
    {
      "arn": "arn:aws:greengrass:us-west-2:123456789012:components:com.example.HelloWorld",
      "componentName": "com.example.HelloWorld",
      "latestVersion": {
```

```
    "arn": "arn:aws:greengrass:us-west-2:123456789012:components:com.example.HelloWorld:versions:1.0.1",
    "componentVersion": "1.0.1",
    "creationTimestamp": "2021-01-08T16:51:07.352000-08:00",
    "description": "My first AWS IoT Greengrass component.",
    "publisher": "Amazon",
    "platforms": [
      {
        "attributes": {
          "os": "linux"
        }
      }
    ]
  }
]
```

有关更多信息，请参阅《AWS 物联网 Greengrass V2 开发者指南》中的[管理组件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListComponents](#)中的。

## list-core-devices

以下代码示例演示了如何使用 list-core-devices。

### AWS CLI

列出核心设备

以下 list-core-devices 示例列出了 AWS 您账户中 AWS 当前区域的 IoT Greengrass 核心设备。

```
aws greengrassv2 list-core-devices
```

输出：

```
{
  "coreDevices": [
    {
      "coreDeviceThingName": "MyGreengrassCore",
      "status": "HEALTHY",
```

```
        "lastStatusUpdateTimestamp": "2021-01-08T04:57:58.838000-08:00"
      }
    ]
  }
```

有关更多信息，请参阅 AWS IoT Greengrass V2 开发者指南中的[检查核心设备状态](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListCoreDevices](#)中的。

## list-deployments

以下代码示例演示了如何使用 list-deployments。

### AWS CLI

#### 列出部署

以下 list-deployments 示例列出了您在当前区域的 AWS 账户中定义的每个部署的最新版本。

```
aws greengrassv2 list-deployments
```

输出：

```
{
  "deployments": [
    {
      "targetArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/MyGreengrassCoreGroup",
      "revisionId": "14",
      "deploymentId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "deploymentName": "Deployment for MyGreengrassCoreGroup",
      "creationTimestamp": "2021-01-07T17:21:20.691000-08:00",
      "deploymentStatus": "ACTIVE",
      "isLatestForTarget": false
    },
    {
      "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
      "revisionId": "1",
      "deploymentId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "deploymentName": "Deployment for MyGreengrassCore",
      "creationTimestamp": "2021-01-06T16:10:42.407000-08:00",

```

```
        "deploymentStatus": "COMPLETED",
        "isLatestForTarget": false
      }
    ]
  }
}
```

有关更多信息，请参阅 AWS IoT Greengrass V2 开发人员指南中的将[组件部署到设备](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListDeployments](#)中的。

## list-effective-deployments

以下代码示例演示了如何使用 list-effective-deployments。

### AWS CLI

#### 列出部署任务

以下 list-effective-deployments 示例列出了适用于 AWS 物联网 Greengrass 核心设备的部署。

```
aws greengrassv2 list-effective-deployments \
  --core-device-thing-name MyGreengrassCore
```

输出：

```
{
  "effectiveDeployments": [
    {
      "deploymentId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "deploymentName": "Deployment for MyGreengrassCore",
      "iotJobId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
      "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
      "coreDeviceExecutionStatus": "COMPLETED",
      "reason": "SUCCESSFUL",
      "creationTimestamp": "2021-01-06T16:10:42.442000-08:00",
      "modifiedTimestamp": "2021-01-08T17:21:27.830000-08:00"
    },
    {
      "deploymentId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "deploymentName": "Deployment for MyGreengrassCoreGroup",

```

```

        "iotJobId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
        "iotJobArn": "arn:aws:iot:us-west-2:123456789012:job/a1b2c3d4-5678-90ab-
cdef-EXAMPLE44444",
        "targetArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
MyGreengrassCoreGroup",
        "coreDeviceExecutionStatus": "SUCCEEDED",
        "reason": "SUCCESSFUL",
        "creationTimestamp": "2021-01-07T17:19:20.394000-08:00",
        "modifiedTimestamp": "2021-01-07T17:21:20.721000-08:00"
    }
]
}

```

有关更多信息，请参阅 AWS IoT Greengrass V2 开发者指南中的[检查核心设备状态](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListEffectiveDeployments](#)中的。

## list-installed-components

以下代码示例演示了如何使用 list-installed-components。

### AWS CLI

列出安装在核心设备上的组件

以下 list-installed-components 示例列出了安装在 AWS 物联网 Greengrass 核心设备上的组件。

```
aws greengrassv2 list-installed-components \
  --core-device-thing-name MyGreengrassCore
```

输出：

```

{
  "installedComponents": [
    {
      "componentName": "aws.greengrass.Cli",
      "componentVersion": "2.0.3",
      "lifecycleState": "RUNNING",
      "isRoot": true
    },
    {

```

```
        "componentName": "aws.greengrass.Nucleus",
        "componentVersion": "2.0.3",
        "lifecycleState": "FINISHED",
        "isRoot": true
    }
]
```

有关更多信息，请参阅 AWS IoT Greengrass V2 开发者指南中的[检查核心设备状态](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListInstalledComponents](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出资源的标签

以下 `list-tags-for-resource` 示例列出了 AWS 物联网 Greengrass 核心设备的所有标签。

```
aws greengrassv2 list-tags-for-resource \
  --resource-arn arn:aws:greengrass:us-
west-2:123456789012:coreDevices:MyGreengrassCore
```

输出：

```
{
  "tags": {
    "Owner": "richard-roe"
  }
}
```

有关更多信息，请参阅 AWS IoT Greengrass V2 开发者指南中的[为资源添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListTagsForResource](#)中的。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

## AWS CLI

要将标签添加到资源中

以下tag-resource示例向 AWS 物联网 Greengrass 核心设备添加所有者标签。您可以使用此标签根据谁拥有核心设备来控制对核心设备的访问权限。

```
aws greengrassv2 tag-resource \  
  --resource-arn arn:aws:greengrass:us-  
west-2:123456789012:coreDevices:MyGreengrassCore \  
  --tags Owner=richard-roe
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT Greengrass V2 开发者指南中的为[资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

## AWS CLI

从资源中移除标签

以下untag-resource示例从 AWS 物联网 Greengrass 核心设备中移除所有者标签。

```
aws iotsitewise untag-resource \  
  --resource-arn arn:aws:greengrass:us-  
west-2:123456789012:coreDevices:MyGreengrassCore \  
  --tag-keys Owner
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT Greengrass V2 开发者指南中的为[资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-connectivity-info

以下代码示例演示了如何使用 update-connectivity-info。



## AWS CLI

### 更新 Greengrass 核心设备的连接信息

以下update-connectivity-info示例获取了 Greengrass 核心设备的连接信息。客户端设备使用此信息连接到在此核心设备上运行的 MQTT 代理。

```
aws greengrassv2 update-connectivity-info \  
  --thing-name MyGreengrassCore \  
  --cli-input-json file://core-device-connectivity-info.json
```

core-device-connectivity-info.json 的内容：

```
{  
  "connectivityInfo": [  
    {  
      "hostAddress": "192.0.2.0",  
      "portNumber": 8883,  
      "id": "localIP_192.0.2.0"  
    }  
  ]  
}
```

输出：

```
{  
  "version": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"  
}
```

有关更多信息，请参阅 AWS IoT Greengrass V2 开发者指南中的[管理核心设备端点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateConnectivityInfo](#)中的。

## AWS IoT Jobs SDK release 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS IoT Jobs SDK release。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### **describe-job-execution**

以下代码示例演示了如何使用 describe-job-execution。

#### AWS CLI

获取任务执行的详细信息

以下describe-job-execution示例检索指定任务和事物最新执行的详细信息。

```
aws iot-jobs-data describe-job-execution \  
  --job-id SampleJob \  
  --thing-name MotionSensor1 \  
  --endpoint-url https://1234567890abcd.jobs.iot.us-west-2.amazonaws.com
```

输出：

```
{  
  "execution": {  
    "approximateSecondsBeforeTimedOut": 88,  
    "executionNumber": 2939653338,  
    "jobId": "SampleJob",  
    "lastUpdatedAt": 1567701875.743,  
    "queuedAt": 1567701902.444,  
    "status": "QUEUED",  
    "thingName": "MotionSensor1 ",  
    "versionNumber": 3  
  }  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[设备和作业](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeJobExecution](#)中的。

## get-pending-job-executions

以下代码示例演示了如何使用 `get-pending-job-executions`。

### AWS CLI

获取某件事未处于终端状态的所有任务的列表

以下`get-pending-job-executions`示例显示了指定事物未处于终止状态的所有任务的列表。

```
aws iot-jobs-data get-pending-job-executions \  
  --thing-name MotionSensor1 \  
  --endpoint-url https://1234567890abcd.jobs.iot.us-west-2.amazonaws.com
```

输出：

```
{  
  "InProgressJobs": [  
  ],  
  "queuedJobs": [  
    {  
      "executionNumber": 2939653338,  
      "jobId": "SampleJob",  
      "lastUpdatedAt": 1567701875.743,  
      "queuedAt": 1567701902.444,  
      "versionNumber": 3  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[设备和作业](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPendingJobExecutions](#)中的。

## start-next-pending-job-execution

以下代码示例演示了如何使用 `start-next-pending-job-execution`。

### AWS CLI

开始执行某件事的下一个待处理任务

以下`start-next-pending-job-execution`示例检索并启动下一个任务执行，其状态为 `IN_PROGRESS` 或 `QUEUED` 用于指定事物。

```
aws iot-jobs-data start-next-pending-job-execution \  
  --thing-name MotionSensor1 \  
  --endpoint-url https://1234567890abcd.jobs.iot.us-west-2.amazonaws.com
```

输出：

```
{  
  "execution": {  
    "approximateSecondsBeforeTimedOut": 88,  
    "executionNumber": 2939653338,  
    "jobId": "SampleJob",  
    "lastUpdatedAt": 1567714853.743,  
    "queuedAt": 1567701902.444,  
    "startedAt": 1567714871.690,  
    "status": "IN_PROGRESS",  
    "thingName": "MotionSensor1 ",  
    "versionNumber": 3  
  }  
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[设备和作业](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartNextPendingJobExecution](#)中的。

## update-job-execution

以下代码示例演示了如何使用 `update-job-execution`。

### AWS CLI

更新任务执行的状态

以下`update-job-execution`示例更新了指定任务和事物的状态。

```
aws iot-jobs-data update-job-execution \  
  --job-id SampleJob \  
  --thing-name MotionSensor1 \  
  --status REMOVED \  
  --endpoint-url https://1234567890abcd.jobs.iot.us-west-2.amazonaws.com
```

```
--endpoint-url https://1234567890abcd.jobs.iot.us-west-2.amazonaws.com
```

输出：

```
{
  "executionState": {
    "status": "REMOVED",
    "versionNumber": 3
  },
}
```

有关更多信息，请参阅《AWS 物联网开发者指南》中的[设备和作业](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateJobExecution](#) 中的。

## AWS IoT SiteWise 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS IoT SiteWise。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-assets**

以下代码示例演示了如何使用 `associate-assets`。

AWS CLI

将子资产与父项资产相关联

以下associate-assets示例将风力涡轮机资产与风力发电场资产相关联，其中风力涡轮机资产模型作为层次结构存在于风电场资产模型中。

```
aws iotsitewise associate-assets \  
  --asset-id a1b2c3d4-5678-90ab-cdef-44444EXAMPLE \  
  --hierarchy-id a1b2c3d4-5678-90ab-cdef-77777EXAMPLE \  
  --child-asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[关联资产](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateAssets](#)中的。

## batch-associate-project-assets

以下代码示例演示了如何使用 batch-associate-project-assets。

### AWS CLI

将资产与项目关联

以下batch-associate-project-assets示例将风力发电场资产与项目相关联。

```
aws iotsitewise batch-associate-project-assets \  
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeeeEXAMPLE \  
  --asset-ids a1b2c3d4-5678-90ab-cdef-44444EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 监控器应用指南》中的[向项目添加资产](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchAssociateProjectAssets](#)中的。

## batch-disassociate-project-assets

以下代码示例演示了如何使用 batch-disassociate-project-assets。

### AWS CLI

取消资产与项目的关联

以下batch-disassociate-project-assets示例将风力发电场资产与项目断开关联。

```
aws iotsitewise batch-disassociate-project-assets \  
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeEXAMPLE \  
  --asset-ids a1b2c3d4-5678-90ab-cdef-44444EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 监控器应用指南》中的[向项目添加资产](#)。

• 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchDisassociateProjectAssets](#)中的。

## batch-put-asset-property-value

以下代码示例演示了如何使用 batch-put-asset-property-value。

### AWS CLI

向资产属性发送数据

以下batch-put-asset-property-value示例将功率和温度数据发送到由属性别名标识的资产属性。

```
aws iotsitewise batch-put-asset-property-value \  
  --cli-input-json file://batch-put-asset-property-value.json
```

batch-put-asset-property-value.json 的内容：

```
{  
  "entries": [  
    {  
      "entryId": "1575691200-company-windfarm-3-turbine-7-power",  
      "propertyAlias": "company-windfarm-3-turbine-7-power",  
      "propertyValues": [  
        {  
          "value": {  
            "doubleValue": 4.92  
          },  
          "timestamp": {  
            "timeInSeconds": 1575691200  
          },  
        }  
      ]  
    }  
  ]  
}
```

```

        "quality": "GOOD"
      }
    ]
  },
  {
    "entryId": "1575691200-company-windfarm-3-turbine-7-temperature",
    "propertyAlias": "company-windfarm-3-turbine-7-temperature",
    "propertyValues": [
      {
        "value": {
          "integerValue": 38
        },
        "timestamp": {
          "timeInSeconds": 1575691200
        }
      }
    ]
  }
]
}

```

输出：

```

{
  "errorEntries": []
}

```

有关更多信息，请参阅《[物联网 SiteWise 用户指南](#)》中的[使用 SiteWise I AWS oT API 提取数据](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchPutAssetPropertyValue](#)中的。

## create-access-policy

以下代码示例演示了如何使用 create-access-policy。

### AWS CLI

示例 1：向用户授予对门户的管理权限

以下create-access-policy示例创建了一个访问策略，该策略向用户授予对风电场公司门户网站的管理访问权限。



```
aws iotsitewise create-access-policy \  
  --cli-input-json file://create-portal-administrator-access-policy.json
```

create-portal-administrator-access-policy.json 的内容：

```
{  
  "accessPolicyIdentity": {  
    "user": {  
      "id": "a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-bbbbbbEXAMPLE"  
    }  
  },  
  "accessPolicyPermission": "ADMINISTRATOR",  
  "accessPolicyResource": {  
    "portal": {  
      "id": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE"  
    }  
  }  
}
```

输出：

```
{  
  "accessPolicyId": "a1b2c3d4-5678-90ab-cdef-cccccEXAMPLE",  
  "accessPolicyArn": "arn:aws:iotsitewise:us-west-2:123456789012:access-policy/  
a1b2c3d4-5678-90ab-cdef-cccccEXAMPLE"  
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[添加或移除门户管理员](#)。

示例 2：授予用户对项目的只读访问权限

以下 create-access-policy 示例创建了一个访问策略，该策略向用户授予对风力发电场项目的只读访问权限。

```
aws iotsitewise create-access-policy \  
  --cli-input-json file://create-project-viewer-access-policy.json
```

create-project-viewer-access-policy.json 的内容：

```
{
```

```

    "accessPolicyIdentity": {
      "user": {
        "id": "a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-bbbbbEXAMPLE"
      }
    },
    "accessPolicyPermission": "VIEWER",
    "accessPolicyResource": {
      "project": {
        "id": "a1b2c3d4-5678-90ab-cdef-eeeeEXAMPLE"
      }
    }
  }
}

```

输出：

```

{
  "accessPolicyId": "a1b2c3d4-5678-90ab-cdef-dddddEXAMPLE",
  "accessPolicyArn": "arn:aws:iotsitewise:us-west-2:123456789012:access-policy/a1b2c3d4-5678-90ab-cdef-dddddEXAMPLE"
}

```

有关更多信息，请参阅《AWS 物联网 SiteWise 监控器应用指南》中的[分配项目查看者](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAccessPolicy](#)中的。

## create-asset-model

以下代码示例演示了如何使用 create-asset-model。

### AWS CLI

#### 创建资产模型

以下create-asset-model示例创建了一个资产模型，该模型定义了具有以下属性的风力涡轮机：

序列号-风力涡轮机的序列号  
发电量-风力涡轮机产生的功率数据流  
温度C-风力涡轮机的温度数据流  
以 CelsiusTemperature F 为单位-映射的温度数据点从摄氏度到华氏度不等

```

aws iotsitewise create-asset-model \
  --cli-input-json file://create-wind-turbine-model.json

```

create-wind-turbine-model.json 的内容：

```
{
  "assetModelName": "Wind Turbine Model",
  "assetModelDescription": "Represents a wind turbine",
  "assetModelProperties": [
    {
      "name": "Serial Number",
      "dataType": "STRING",
      "type": {
        "attribute": {}
      }
    },
    {
      "name": "Generated Power",
      "dataType": "DOUBLE",
      "unit": "kW",
      "type": {
        "measurement": {}
      }
    },
    {
      "name": "Temperature C",
      "dataType": "DOUBLE",
      "unit": "Celsius",
      "type": {
        "measurement": {}
      }
    },
    {
      "name": "Temperature F",
      "dataType": "DOUBLE",
      "unit": "Fahrenheit",
      "type": {
        "transform": {
          "expression": "temp_c * 9 / 5 + 32",
          "variables": [
            {
              "name": "temp_c",
              "value": {
                "propertyId": "Temperature C"
              }
            }
          ]
        }
      }
    }
  ]
}
```

```

    }
  },
  {
    "name": "Total Generated Power",
    "dataType": "DOUBLE",
    "unit": "kW",
    "type": {
      "metric": {
        "expression": "sum(power)",
        "variables": [
          {
            "name": "power",
            "value": {
              "propertyId": "Generated Power"
            }
          }
        ],
        "window": {
          "tumbling": {
            "interval": "1h"
          }
        }
      }
    }
  }
]
}

```

输出：

```

{
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "assetModelStatus": {
    "state": "CREATING"
  }
}

```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[定义资产模型](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateAssetModel](#) 中的。

## create-asset

以下代码示例演示了如何使用 create-asset。

### AWS CLI

#### 创建资产

以下create-asset示例根据风力涡轮机资产模型创建风力涡轮机资产。

```
aws iotsitewise create-asset \  
  --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \  
  --asset-name "Wind Turbine 1"
```

输出：

```
{  
  "assetId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",  
  "assetArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset/  
a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",  
  "assetStatus": {  
    "state": "CREATING"  
  }  
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[创建资产](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAsset](#)中的。

## create-dashboard

以下代码示例演示了如何使用 create-dashboard。

### AWS CLI

#### 创建仪表板

以下create-dashboard示例创建了一个带有折线图的仪表板，该仪表板显示了风力发电场的总发电量。

```
aws iotsitewise create-dashboard \  
  --asset-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \  
  --dashboard-name "Wind Turbine 1 Dashboard"
```

```
--project-id a1b2c3d4-5678-90ab-cdef-eeeeEXAMPLE \  
--dashboard-name "Wind Farm" \  
--dashboard-definition file://create-wind-farm-dashboard.json
```

create-wind-farm-dashboard.json 的内容：

```
{  
  "widgets": [  
    {  
      "type": "monitor-line-chart",  
      "title": "Generated Power",  
      "x": 0,  
      "y": 0,  
      "height": 3,  
      "width": 3,  
      "metrics": [  
        {  
          "label": "Power",  
          "type": "iotsitewise",  
          "assetId": "a1b2c3d4-5678-90ab-cdef-44444EXAMPLE",  
          "propertyId": "a1b2c3d4-5678-90ab-cdef-99999EXAMPLE"  
        }  
      ]  
    }  
  ]  
}
```

输出：

```
{  
  "dashboardId": "a1b2c3d4-5678-90ab-cdef-fffffEXAMPLE",  
  "dashboardArn": "arn:aws:iotsitewise:us-west-2:123456789012:dashboard/  
a1b2c3d4-5678-90ab-cdef-fffffEXAMPLE"  
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[创建仪表板 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDashboard](#)中的。

## create-gateway

以下代码示例演示了如何使用 create-gateway。

## AWS CLI

### 创建网关

以下`create-gateway`示例创建了一个在 IoT Greengrass 上运行的 AWS 网关。

```
aws iotsitewise create-gateway \  
  --gateway-name ExampleCorpGateway \  
  --gateway-platform greengrass={groupArn=arn:aws:greengrass:us-  
west-2:123456789012:/greengrass/groups/a1b2c3d4-5678-90ab-cdef-1b1b1EXAMPLE}
```

输出：

```
{  
  "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",  
  "gatewayArn": "arn:aws:iotsitewise:us-west-2:123456789012:gateway/  
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"  
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[配置网关](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateGateway](#)中的。

## create-portal

以下代码示例演示了如何使用 `create-portal`。

## AWS CLI

### 创建门户

以下`create-portal`示例为一家风电场公司创建了一个门户网站。您只能在启用 AWS 单点登录的同一地区创建门户。

```
aws iotsitewise create-portal \  
  --portal-name WindFarmPortal \  
  --portal-description "A portal that contains wind farm projects for Example  
Corp." \  
  --portal-contact-email support@example.com \  
  --role-arn arn:aws:iam::123456789012:role/service-role/  
MySiteWiseMonitorServiceRole
```

输出：

```
{
  "portalId": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
  "portalArn": "arn:aws:iotsitewise:us-west-2:123456789012:portal/
a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
  "portalStartUrl": "https://a1b2c3d4-5678-90ab-cdef-
aaaaaEXAMPLE.app.iotsitewise.aws",
  "portalStatus": {
    "state": "CREATING"
  },
  "ssoApplicationId": "ins-a1b2c3d4-EXAMPLE"
}
```

有关更多信息，请参阅《物联网用户指南》中的“[AWS 物联网 SiteWise 监控器入门](#)”和“AWS 物联网 SiteWise 用户指南”中的“[AWS 启用 AWS SSO](#)”。SiteWise

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePortal](#)中的。

## create-project

以下代码示例演示了如何使用 create-project。

### AWS CLI

#### 创建项目

以下create-project示例创建了一个风力发电场项目。

```
aws iotsitewise create-project \
  --portal-id a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE \
  --project-name "Wind Farm 1" \
  --project-description "Contains asset visualizations for Wind Farm #1 for
Example Corp."
```

输出：

```
{
  "projectId": "a1b2c3d4-5678-90ab-cdef-eeeeeeEXAMPLE",
  "projectArn": "arn:aws:iotsitewise:us-west-2:123456789012:project/
a1b2c3d4-5678-90ab-cdef-eeeeeeEXAMPLE"
```



```
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 监控器应用指南》中的[创建项目](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateProject](#)中的。

## delete-access-policy

以下代码示例演示了如何使用 delete-access-policy。

### AWS CLI

撤消用户对项目或门户的访问权限

以下delete-access-policy示例删除了授予用户对门户的管理访问权限的访问策略。

```
aws iotsitewise delete-access-policy \  
  --access-policy-id a1b2c3d4-5678-90ab-cdef-ccccEXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[添加或移除门户管理员](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAccessPolicy](#)中的。

## delete-asset-model

以下代码示例演示了如何使用 delete-asset-model。

### AWS CLI

删除资产模型

以下delete-asset-model示例删除了风力涡轮机资产模型。

```
aws iotsitewise delete-asset-model \  
  --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
```

输出：

```
{
```

```
"assetModelStatus": {
  "state": "DELETING"
}
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[删除资产模型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAssetModel](#)中的。

## delete-asset

以下代码示例演示了如何使用 delete-asset。

### AWS CLI

#### 删除资产

以下delete-asset示例删除风力涡轮机资产。

```
aws iotsitewise delete-asset \
  --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE
```

输出：

```
{
  "assetStatus": {
    "state": "DELETING"
  }
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[删除资产](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAsset](#)中的。

## delete-dashboard

以下代码示例演示了如何使用 delete-dashboard。

### AWS CLI

#### 删除控制面板

以下delete-dashboard示例删除风力涡轮机仪表板。

```
aws iotsitewise delete-dashboard \  
  --dashboard-id a1b2c3d4-5678-90ab-cdef-fffffEXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 监控器应用指南》中的[删除仪表板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDashboard](#)中的。

## delete-gateway

以下代码示例演示了如何使用 delete-gateway。

### AWS CLI

#### 删除网关

以下delete-gateway示例删除网关。

```
aws iotsitewise delete-gateway \  
  --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《物AWS 联网 SiteWise 用户指南》中的[使用网关摄取数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteGateway](#)中的。

## delete-portal

以下代码示例演示了如何使用 delete-portal。

### AWS CLI

#### 删除入口

以下delete-portal示例删除了一家风电场公司的 Web 门户。

```
aws iotsitewise delete-portal \  
  --portal-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

```
--portal-id a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE
```

输出：

```
{
  "portalStatus": {
    "state": "DELETING"
  }
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[删除门户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePortal](#)中的。

## delete-project

以下代码示例演示了如何使用 delete-project。

### AWS CLI

#### 删除项目

以下delete-project示例删除了一个风力发电场项目。

```
aws iotsitewise delete-project \
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeeeEXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 监控器应用指南》中的[删除项目](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteProject](#)中的。

## describe-access-policy

以下代码示例演示了如何使用 describe-access-policy。

### AWS CLI

#### 描述访问策略

以下describe-access-policy示例描述了一种访问策略，该策略授予用户对风电场公司门户网站的管理访问权限。

```
aws iotsitewise describe-access-policy \  
  --access-policy-id a1b2c3d4-5678-90ab-cdef-ccccEXAMPLE
```

输出：

```
{  
  "accessPolicyId": "a1b2c3d4-5678-90ab-cdef-ccccEXAMPLE",  
  "accessPolicyArn": "arn:aws:iotsitewise:us-west-2:123456789012:access-policy/  
a1b2c3d4-5678-90ab-cdef-ccccEXAMPLE",  
  "accessPolicyIdentity": {  
    "user": {  
      "id": "a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-bbbbbbEXAMPLE"  
    }  
  },  
  "accessPolicyResource": {  
    "portal": {  
      "id": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE"  
    }  
  },  
  "accessPolicyPermission": "ADMINISTRATOR",  
  "accessPolicyCreationDate": "2020-02-20T22:35:15.552880124Z",  
  "accessPolicyLastUpdateDate": "2020-02-20T22:35:15.552880124Z"  
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[添加或移除门户管理员](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAccessPolicy](#)中的。

## describe-asset-model

以下代码示例演示了如何使用 describe-asset-model。

### AWS CLI

#### 描述资产模型

以下describe-asset-model示例描述了风力发电场的资产模型。

```
aws iotsitewise describe-asset-model \  
  --asset-model-id a1b2c3d4-5678-90ab-cdef-ccccEXAMPLE
```

```
--asset-model-id a1b2c3d4-5678-90ab-cdef-22222EXAMPLE
```

输出：

```
{
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
  "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
  "assetModelName": "Wind Farm Model",
  "assetModelDescription": "Represents a wind farm that comprises many wind
turbines",
  "assetModelProperties": [
    {
      "id": "a1b2c3d4-5678-90ab-cdef-99999EXAMPLE",
      "name": "Total Generated Power",
      "dataType": "DOUBLE",
      "unit": "kW",
      "type": {
        "metric": {
          "expression": "sum(power)",
          "variables": [
            {
              "name": "power",
              "value": {
                "propertyId": "a1b2c3d4-5678-90ab-
cdef-66666EXAMPLE",
                "hierarchyId": "a1b2c3d4-5678-90ab-
cdef-77777EXAMPLE"
              }
            }
          ],
          "window": {
            "tumbling": {
              "interval": "1h"
            }
          }
        }
      }
    },
    {
      "id": "a1b2c3d4-5678-90ab-cdef-88888EXAMPLE",
      "name": "Region",
      "dataType": "STRING",
```

```
        "type": {
            "attribute": {
                "defaultValue": " "
            }
        }
    ],
    "assetModelHierarchies": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-77777EXAMPLE",
            "name": "Wind Turbines",
            "childAssetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
        }
    ],
    "assetModelCreationDate": 1575671284.0,
    "assetModelLastUpdateDate": 1575671988.0,
    "assetModelStatus": {
        "state": "ACTIVE"
    }
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[描述特定资产模型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAssetModel](#)中的。

## describe-asset-property

以下代码示例演示了如何使用 describe-asset-property。

### AWS CLI

#### 描述资产属性

以下describe-asset-property示例描述了风力发电场资产的总发电量。

```
aws iotsitewise describe-asset-property \
  --asset-id a1b2c3d4-5678-90ab-cdef-44444EXAMPLE \
  --property-id a1b2c3d4-5678-90ab-cdef-99999EXAMPLE
```

输出：

```
{
```

```

"assetId": "a1b2c3d4-5678-90ab-cdef-44444EXAMPLE",
"assetName": "Wind Farm 1",
"assetModelId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
"assetProperty": {
  "id": "a1b2c3d4-5678-90ab-cdef-99999EXAMPLE",
  "name": "Total Generated Power",
  "notification": {
    "topic": "$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-
cdef-22222EXAMPLE/assets/a1b2c3d4-5678-90ab-cdef-44444EXAMPLE/properties/
a1b2c3d4-5678-90ab-cdef-99999EXAMPLE",
    "state": "DISABLED"
  },
  "dataType": "DOUBLE",
  "unit": "kW",
  "type": {
    "metric": {
      "expression": "sum(power)",
      "variables": [
        {
          "name": "power",
          "value": {
            "propertyId": "a1b2c3d4-5678-90ab-cdef-66666EXAMPLE",
            "hierarchyId": "a1b2c3d4-5678-90ab-cdef-77777EXAMPLE"
          }
        }
      ],
      "window": {
        "tumbling": {
          "interval": "1h"
        }
      }
    }
  }
}
}
}
}

```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[描述特定资产属性](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAssetProperty](#)中的。

## describe-asset

以下代码示例演示了如何使用 describe-asset。



## AWS CLI

### 描述资产

以下describe-asset示例描述了风力发电场资产。

```
aws iotsitewise describe-asset \  
  --asset-id a1b2c3d4-5678-90ab-cdef-44444EXAMPLE
```

输出：

```
{  
  "assetId": "a1b2c3d4-5678-90ab-cdef-44444EXAMPLE",  
  "assetArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset/  
a1b2c3d4-5678-90ab-cdef-44444EXAMPLE",  
  "assetName": "Wind Farm 1",  
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",  
  "assetProperties": [  
    {  
      "id": "a1b2c3d4-5678-90ab-cdef-88888EXAMPLE",  
      "name": "Region",  
      "dataType": "STRING"  
    },  
    {  
      "id": "a1b2c3d4-5678-90ab-cdef-99999EXAMPLE",  
      "name": "Total Generated Power",  
      "dataType": "DOUBLE",  
      "unit": "kW"  
    }  
  ],  
  "assetHierarchies": [  
    {  
      "id": "a1b2c3d4-5678-90ab-cdef-77777EXAMPLE",  
      "name": "Wind Turbines"  
    }  
  ],  
  "assetCreationDate": 1575672453.0,  
  "assetLastUpdateDate": 1575672453.0,  
  "assetStatus": {  
    "state": "ACTIVE"  
  }  
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[描述特定资产](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAsset](#)中的。

## describe-dashboard

以下代码示例演示了如何使用 describe-dashboard。

### AWS CLI

#### 描述仪表板

以下describe-dashboard示例描述了指定的风力发电场仪表板。

```
aws iotsitewise describe-dashboard \  
  --dashboard-id a1b2c3d4-5678-90ab-cdef-fffffEXAMPLE
```

输出：

```
{  
  "dashboardId": "a1b2c3d4-5678-90ab-cdef-fffffEXAMPLE",  
  "dashboardArn": "arn:aws:iotsitewise:us-west-2:123456789012:dashboard/  
a1b2c3d4-5678-90ab-cdef-fffffEXAMPLE",  
  "dashboardName": "Wind Farm",  
  "projectId": "a1b2c3d4-5678-90ab-cdef-eeeeeeEXAMPLE",  
  "dashboardDefinition": "{\"widgets\": [{\"type\": \"monitor-line-chart\", \"title  
\": \"Generated Power\", \"x\": 0, \"y\": 0, \"height\": 3, \"width\": 3, \"metrics\":  
[ {\"label\": \"Power\", \"type\": \"iotsitewise\", \"assetId\": \"a1b2c3d4-5678-90ab-  
cdef-44444EXAMPLE\", \"propertyId\": \"a1b2c3d4-5678-90ab-cdef-99999EXAMPLE\" } ] } ] }",  
  "dashboardCreationDate": "2020-05-01T20:32:12.228476348Z",  
  "dashboardLastUpdateDate": "2020-05-01T20:32:12.228476348Z"  
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 监控器应用指南》中的[查看仪表板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDashboard](#)中的。

## describe-gateway-capability-configuration

以下代码示例演示了如何使用 describe-gateway-capability-configuration。

## AWS CLI

### 描述网关功能

以下describe-gateway-capability-configuration示例描述了 OPC-UA 源功能。

```
aws iotsitewise describe-gateway-capability-configuration \  
  --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE \  
  --capability-namespace "iotsitewise:opcuacollector:1"
```

输出：

```
{  
  "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",  
  "capabilityNamespace": "iotsitewise:opcuacollector:1",  
  "capabilityConfiguration": "{\"sources\": [{\"name\": \"Wind Farm #1\",  
  \"endpoint\": {\"certificateTrust\": {\"type\": \"TrustAny\"}, \"endpointUri\": \"opc.tcp://203.0.113.0:49320\", \"securityPolicy\": \"BASIC256\",  
  \"messageSecurityMode\": \"SIGN_AND_ENCRYPT\", \"identityProvider\": {\"type\": \"Username\", \"usernameSecretArn\": \"arn:aws:secretsmanager:us-east-1:123456789012:secret:green-grass-factory1-auth-3QNDmM\"}, \"nodeFilterRules\": []}, \"measurementDataStreamPrefix\": \"\"}]}",  
  "capabilitySyncStatus": "IN_SYNC"  
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[配置数据源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeGatewayCapabilityConfiguration](#)中的。

## describe-gateway

以下代码示例演示了如何使用 describe-gateway。

### AWS CLI

#### 描述网关

以下describe-gateway示例描述了网关。

```
aws iotsitewise describe-gateway \  
  --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

```
--gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

输出：

```
{
  "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
  "gatewayName": "ExampleCorpGateway",
  "gatewayArn": "arn:aws:iotsitewise:us-west-2:123456789012:gateway/a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
  "gatewayPlatform": {
    "greengrass": {
      "groupArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/groups/a1b2c3d4-5678-90ab-cdef-1b1b1EXAMPLE"
    }
  },
  "gatewayCapabilitySummaries": [
    {
      "capabilityNamespace": "iotsitewise:opcuacollector:1",
      "capabilitySyncStatus": "IN_SYNC"
    }
  ],
  "creationDate": 1588369971.457,
  "lastUpdateDate": 1588369971.457
}
```

有关更多信息，请参阅《物AWS 联网 SiteWise 用户指南》中的[使用网关摄取数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeGateway](#)中的。

## describe-logging-options

以下代码示例演示了如何使用 describe-logging-options。

### AWS CLI

检索当前 AWS IoT SiteWise 日志选项

以下describe-logging-options示例检索当前区域中您的 AWS 账户的当前 AWS IoT SiteWise 日志选项。

```
aws iotsitewise describe-logging-options
```

输出：

```
{
  "loggingOptions": {
    "level": "INFO"
  }
}
```

有关更多信息，请参阅[AWS 物联网 SiteWise 用户指南中的 SiteWise 使用 Amazon CloudWatch 日志监控AWS物联网](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoggingOptions](#)中的。

## describe-portal

以下代码示例演示了如何使用 describe-portal。

### AWS CLI

描述门户

以下describe-portal示例描述了一家风电场公司的 Web 门户。

```
aws iotsitewise describe-portal \
  --portal-id a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE
```

输出：

```
{
  "portalId": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
  "portalArn": "arn:aws:iotsitewise:us-west-2:123456789012:portal/a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
  "portalName": "WindFarmPortal",
  "portalDescription": "A portal that contains wind farm projects for Example Corp.",
  "portalClientId": "E-a1b2c3d4e5f6_a1b2c3d4e5f6EXAMPLE",
  "portalStartUrl": "https://a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE.app.iotsitewise.aws",
  "portalContactEmail": "support@example.com",
  "portalStatus": {
    "state": "ACTIVE"
  },
  "portalCreationDate": "2020-02-04T23:01:52.90248068Z",
}
```

```
"portalLastUpdateDate": "2020-02-04T23:01:52.90248078Z",  
"roleArn": "arn:aws:iam::123456789012:role/MySiteWiseMonitorServiceRole"  
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[管理门户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePortal](#)中的。

## describe-project

以下代码示例演示了如何使用 describe-project。

### AWS CLI

#### 描述项目

以下describe-project示例描述了一个风力发电场项目。

```
aws iotsitewise describe-project \  
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeEXAMPLE
```

输出：

```
{  
  "projectId": "a1b2c3d4-5678-90ab-cdef-eeeeEXAMPLE",  
  "projectArn": "arn:aws:iotsitewise:us-west-2:123456789012:project/  
a1b2c3d4-5678-90ab-cdef-eeeeEXAMPLE",  
  "projectName": "Wind Farm 1",  
  "portalId": "a1b2c3d4-5678-90ab-cdef-aaaaEXAMPLE",  
  "projectDescription": "Contains asset visualizations for Wind Farm #1 for  
Example Corp.",  
  "projectCreationDate": "2020-02-20T21:58:43.362246001Z",  
  "projectLastUpdateDate": "2020-02-20T21:58:43.362246095Z"  
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 监控器应用指南》中的[查看项目详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeProject](#)中的。

## disassociate-assets

以下代码示例演示了如何使用 disassociate-assets。

## AWS CLI

### 取消子资产与父资产的关联

以下disassociate-assets示例将风力涡轮机资产与风力发电场资产断开关联。

```
aws iotsitewise disassociate-assets \  
  --asset-id a1b2c3d4-5678-90ab-cdef-44444EXAMPLE \  
  --hierarchy-id a1b2c3d4-5678-90ab-cdef-77777EXAMPLE \  
  --child-asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[关联资产](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateAssets](#)中的。

## get-asset-property-aggregates

以下代码示例演示了如何使用 get-asset-property-aggregates。

### AWS CLI

#### 检索资产属性的汇总平均值和计数值

以下get-asset-property-aggregates示例检索风力涡轮机资产在 1 小时内的平均总功率和总功率数据点计数。

```
aws iotsitewise get-asset-property-aggregates \  
  --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE \  
  --property-id a1b2c3d4-5678-90ab-cdef-66666EXAMPLE \  
  --start-date 1580849400 \  
  --end-date 1580853000 \  
  --aggregate-types AVERAGE COUNT \  
  --resolution 1h
```

输出：

```
{  
  "aggregatedValues": [  
    {  
      "timestamp": 1580850000.0,  
      "quality": "GOOD",
```

```

        "value": {
            "average": 8723.46538886233,
            "count": 12.0
        }
    ]
}

```

有关更多信息，请参阅《物AWS 联网 SiteWise 用户指南》中的[查询资产属性聚合](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAssetPropertyAggregates](#)中的。

## get-asset-property-value-history

以下代码示例演示了如何使用 get-asset-property-value-history。

### AWS CLI

#### 检索资产属性的历史价值

以下get-asset-property-value-history示例检索风力涡轮机资产在 20 分钟内的总功率值。

```

aws iotsitewise get-asset-property-value-history \
  --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE \
  --property-id a1b2c3d4-5678-90ab-cdef-66666EXAMPLE \
  --start-date 1580851800 \
  --end-date 1580853000

```

输出：

```

{
  "assetPropertyValueHistory": [
    {
      "value": {
        "doubleValue": 7217.787046814844
      },
      "timestamp": {
        "timeInSeconds": 1580852100,
        "offsetInNanos": 0
      },
      "quality": "GOOD"
    },
  ],
}

```



```
{
  "value": {
    "doubleValue": 6941.242811875451
  },
  "timestamp": {
    "timeInSeconds": 1580852400,
    "offsetInNanos": 0
  },
  "quality": "GOOD"
},
{
  "value": {
    "doubleValue": 6976.797662266717
  },
  "timestamp": {
    "timeInSeconds": 1580852700,
    "offsetInNanos": 0
  },
  "quality": "GOOD"
},
{
  "value": {
    "doubleValue": 6890.8677520453875
  },
  "timestamp": {
    "timeInSeconds": 1580853000,
    "offsetInNanos": 0
  },
  "quality": "GOOD"
}
]
```

有关更多信息，请参阅《物AWS 联网 SiteWise 用户指南》中的[查询历史资产属性值](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAssetPropertyValueHistory](#)中的。

## get-asset-property-value

以下代码示例演示了如何使用 get-asset-property-value。

### AWS CLI

检索资产属性的当前值

以下`get-asset-property-value`示例检索风力涡轮机资产的当前总功率。

```
aws iotsitewise get-asset-property-value \  
  --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE \  
  --property-id a1b2c3d4-5678-90ab-cdef-66666EXAMPLE
```

输出：

```
{  
  "propertyValue": {  
    "value": {  
      "doubleValue": 6890.8677520453875  
    },  
    "timestamp": {  
      "timeInSeconds": 1580853000,  
      "offsetInNanos": 0  
    },  
    "quality": "GOOD"  
  }  
}
```

有关更多信息，请参阅《物AWS 联网 SiteWise 用户指南》中的[查询当前资产属性值](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAssetPropertyValue](#)中的。

## list-access-policies

以下代码示例演示了如何使用 `list-access-policies`。

AWS CLI

列出所有访问策略

以下`list-access-policies`示例列出了门户管理员用户的所有访问策略。

```
aws iotsitewise list-access-policies \  
  --identity-type USER \  
  --identity-id a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-bbbbbbEXAMPLE
```

输出：

```
{
```

```
"accessPolicySummaries": [
  {
    "id": "a1b2c3d4-5678-90ab-cdef-ccccEXAMPLE",
    "identity": {
      "user": {
        "id": "a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-bbbbbbEXAMPLE"
      }
    },
    "resource": {
      "portal": {
        "id": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE"
      }
    },
    "permission": "ADMINISTRATOR"
  }
]
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[管理门户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAccessPolicies](#)中的。

## list-asset-models

以下代码示例演示了如何使用 list-asset-models。

### AWS CLI

列出所有资产模型

以下list-asset-models示例列出了您在当前地区的 AWS 账户中定义的所有资产模型。

```
aws iotsitewise list-asset-models
```

输出：

```
{
  "assetModelSummaries": [
    {
      "id": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "arn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",

```

```
    "name": "Wind Farm Model",
    "description": "Represents a wind farm that comprises many wind
turbines",
    "creationDate": 1575671284.0,
    "lastUpdateDate": 1575671988.0,
    "status": {
      "state": "ACTIVE"
    }
  },
  {
    "id": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "arn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "name": "Wind Turbine Model",
    "description": "Represents a wind turbine manufactured by Example Corp",
    "creationDate": 1575671207.0,
    "lastUpdateDate": 1575686273.0,
    "status": {
      "state": "ACTIVE"
    }
  }
]
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[列出所有资产模型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAssetModels](#)中的。

## list-assets

以下代码示例演示了如何使用 list-assets。

### AWS CLI

#### 示例 1：列出所有顶级资产

以下list-assets示例列出了在资产层次结构树中处于顶层并在当前区域的 AWS 账户中定义的所有资产。

```
aws iotsitewise list-assets \
  --filter TOP_LEVEL
```

输出：

```
{
  "assetSummaries": [
    {
      "id": "a1b2c3d4-5678-90ab-cdef-44444EXAMPLE",
      "arn": "arn:aws:iotsitewise:us-west-2:123456789012:asset/a1b2c3d4-5678-90ab-cdef-44444EXAMPLE",
      "name": "Wind Farm 1",
      "assetModelId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "creationDate": 1575672453.0,
      "lastUpdateDate": 1575672453.0,
      "status": {
        "state": "ACTIVE"
      },
      "hierarchies": [
        {
          "id": "a1b2c3d4-5678-90ab-cdef-77777EXAMPLE",
          "name": "Wind Turbines"
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[列出资产](#)。

示例 2：根据资产模型列出所有资产

以下list-assets示例列出了基于资产模型并在当前地区的 AWS 账户中定义的所有资产。

```
aws iotsitewise list-assets \
  --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
```

输出：

```
{
  "assetSummaries": [
    {
      "id": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
      "arn": "arn:aws:iotsitewise:us-west-2:123456789012:asset/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
      "name": "Wind Turbine 1",
      "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
```

```

        "creationDate": 1575671550.0,
        "lastUpdateDate": 1575686308.0,
        "status": {
            "state": "ACTIVE"
        },
        "hierarchies": []
    }
]
}

```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[列出资产](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListAssets](#)中的。

## list-associated-assets

以下代码示例演示了如何使用 list-associated-assets。

### AWS CLI

列出与特定层次结构中的资产关联的所有资产

以下 list-associated-assets 示例列出了与指定风电场资产关联的所有风力涡轮机资产。

```

aws iotsitewise list-associated-assets \
  --asset-id a1b2c3d4-5678-90ab-cdef-44444EXAMPLE \
  --hierarchy-id a1b2c3d4-5678-90ab-cdef-77777EXAMPLE

```

输出：

```

{
  "assetSummaries": [
    {
      "id": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
      "arn": "arn:aws:iotsitewise:us-west-2:123456789012:asset/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
      "name": "Wind Turbine 1",
      "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "creationDate": 1575671550.0,
      "lastUpdateDate": 1575686308.0,
      "status": {
        "state": "ACTIVE"
      },
    },
  ],
}

```

```
        "hierarchies": []
      }
    ]
  }
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中[列出与特定资产关联的资产](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAssociatedAssets](#)中的。

## list-dashboards

以下代码示例演示了如何使用 list-dashboards。

### AWS CLI

列出项目中的所有仪表板

以下list-dashboards示例列出了在项目中定义的所有仪表板。

```
aws iotsitewise list-dashboards \
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeEXAMPLE
```

输出：

```
{
  "dashboardSummaries": [
    {
      "id": "a1b2c3d4-5678-90ab-cdef-ffffEXAMPLE",
      "name": "Wind Farm",
      "creationDate": "2020-05-01T20:32:12.228476348Z",
      "lastUpdateDate": "2020-05-01T20:32:12.228476348Z"
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 监控器应用指南》中的[查看仪表板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDashboards](#)中的。

## list-gateways

以下代码示例演示了如何使用 list-gateways。

## AWS CLI

### 列出所有网关

以下`list-gateways`示例列出了您在当前区域的 AWS 账户中定义的所有网关。

```
aws iotsitewise list-gateways
```

输出：

```
{
  "gatewaySummaries": [
    {
      "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
      "gatewayName": "ExampleCorpGateway",
      "gatewayCapabilitySummaries": [
        {
          "capabilityNamespace": "iotsitewise:opcuacollector:1",
          "capabilitySyncStatus": "IN_SYNC"
        }
      ],
      "creationDate": 1588369971.457,
      "lastUpdateDate": 1588369971.457
    }
  ]
}
```

有关更多信息，请参阅《物AWS 联网 SiteWise 用户指南》中的[使用网关摄取数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListGateways](#)中的。

## list-portals

以下代码示例演示了如何使用 `list-portals`。

### AWS CLI

#### 列出所有传送门

以下`list-portals`示例列出了您在当前区域的 AWS 账户中定义的所有门户。

```
aws iotsitewise list-portals
```



输出：

```
{
  "portalSummaries": [
    {
      "id": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
      "name": "WindFarmPortal",
      "description": "A portal that contains wind farm projects for Example Corp.",
      "startUrl": "https://a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE.app.iotsitewise.aws",
      "creationDate": "2020-02-04T23:01:52.90248068Z",
      "lastUpdateDate": "2020-02-04T23:01:52.90248078Z",
      "roleArn": "arn:aws:iam::123456789012:role/service-role/MySiteWiseMonitorServiceRole"
    }
  ]
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[管理门户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPortals](#)中的。

## list-project-assets

以下代码示例演示了如何使用 list-project-assets。

AWS CLI

列出与项目关联的所有资产

以下list-project-assets示例列出了与风电场项目关联的所有资产。

```
aws iotsitewise list-projects \
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeeeEXAMPLE
```

输出：

```
{
  "assetIds": [
    "a1b2c3d4-5678-90ab-cdef-44444EXAMPLE"
  ]
}
```

```
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 监控器应用指南》中的[向项目添加资产](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListProjectAssets](#)中的。

## list-projects

以下代码示例演示了如何使用 list-projects。

### AWS CLI

列出入口中的所有项目

以下list-projects示例列出了在入口中定义的所有项目。

```
aws iotsitewise list-projects \  
  --portal-id a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE
```

输出：

```
{  
  "projectSummaries": [  
    {  
      "id": "a1b2c3d4-5678-90ab-cdef-eeeeeeEXAMPLE",  
      "name": "Wind Farm 1",  
      "description": "Contains asset visualizations for Wind Farm #1 for  
Example Corp.",  
      "creationDate": "2020-02-20T21:58:43.362246001Z",  
      "lastUpdateDate": "2020-02-20T21:58:43.362246095Z"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 监控器应用指南》中的[查看项目详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListProjects](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 list-tags-for-resource。

## AWS CLI

### 列出资源的所有标签

以下`list-tags-for-resource`示例列出了风力涡轮机资产的所有标签。

```
aws iotsitewise list-tags-for-resource \  
  --resource-arn arn:aws:iotsitewise:us-west-2:123456789012:asset/  
a1b2c3d4-5678-90ab-cdef-33333EXAMPLE
```

输出：

```
{  
  "tags": {  
    "Owner": "richard-roe"  
  }  
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的为[资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## put-logging-options

以下代码示例演示了如何使用 `put-logging-options`。

### AWS CLI

#### 指定日志级别

以下`put-logging-options`示例在 AWS IoT 中启用INFO关卡日志 SiteWise。其他级别包括DEBUG和OFF。

```
aws iotsitewise put-logging-options \  
  --logging-options level=INFO
```

此命令不生成任何输出。

有关更多信息，请参阅[AWS 物联网 SiteWise 用户指南中的 SiteWise 使用 Amazon CloudWatch 日志监控AWS物联网](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutLoggingOptions](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

要将标签添加到资源中

以下tag-resource示例为风力涡轮机资产添加所有者标签。这使您可以根据谁拥有资产来控制对资产的访问权限。

```
aws iotsitewise tag-resource \  
  --resource-arn arn:aws:iotsitewise:us-west-2:123456789012:asset/  
a1b2c3d4-5678-90ab-cdef-33333EXAMPLE \  
  --tags Owner=richard-roe
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的为[资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从资源中移除标签

以下untag-resource示例从风力涡轮机资产中移除所有者标签。

```
aws iotsitewise untag-resource \  
  --resource-arn arn:aws:iotsitewise:us-west-2:123456789012:asset/  
a1b2c3d4-5678-90ab-cdef-33333EXAMPLE \  
  --tag-keys Owner
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的为[资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-access-policy

以下代码示例演示了如何使用 `update-access-policy`。

### AWS CLI

授予项目查看者对项目的所有权

以下 `update-access-policy` 示例更新了向项目查看者授予项目所有权的访问策略。

```
aws iotsitewise update-access-policy \  
  --access-policy-id a1b2c3d4-5678-90ab-cdef-dddddEXAMPLE \  
  --cli-input-json file://update-project-viewer-access-policy.json
```

`update-project-viewer-access-policy.json` 的内容：

```
{  
  "accessPolicyIdentity": {  
    "user": {  
      "id": "a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-bbbbbbEXAMPLE"  
    }  
  },  
  "accessPolicyPermission": "ADMINISTRATOR",  
  "accessPolicyResource": {  
    "project": {  
      "id": "a1b2c3d4-5678-90ab-cdef-eeeeeeEXAMPLE"  
    }  
  }  
}
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 监控器应用指南》中的[分配项目所有者](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateAccessPolicy](#) 中的。

## update-asset-model

以下代码示例演示了如何使用 `update-asset-model`。

### AWS CLI

更新资产模型

以下update-asset-model示例更新了风电场资产模型的描述。此示例包括模型的现有 ID 和定义，因为新模型update-asset-model会覆盖现有模型。

```
aws iotsitewise update-asset-model \  
  --cli-input-json file://update-wind-farm-model.json
```

update-wind-farm-model.json 的内容：

```
{  
  "assetModelName": "Wind Farm Model",  
  "assetModelDescription": "Represents a wind farm that comprises many wind  
turbines",  
  "assetModelProperties": [  
    {  
      "id": "a1b2c3d4-5678-90ab-cdef-88888EXAMPLE",  
      "name": "Region",  
      "dataType": "STRING",  
      "type": {  
        "attribute": {}  
      }  
    },  
    {  
      "id": "a1b2c3d4-5678-90ab-cdef-99999EXAMPLE",  
      "name": "Total Generated Power",  
      "dataType": "DOUBLE",  
      "unit": "kW",  
      "type": {  
        "metric": {  
          "expression": "sum(power)",  
          "variables": [  
            {  
              "name": "power",  
              "value": {  
                "hierarchyId": "a1b2c3d4-5678-90ab-  
cdef-77777EXAMPLE",  
                "propertyId": "a1b2c3d4-5678-90ab-cdef-66666EXAMPLE"  
              }  
            }  
          ],  
          "window": {  
            "tumbling": {  
              "interval": "1h"  
            }  
          }  
        }  
      }  
    }  
  ]  
}
```

```

    }
  }
}
],
"assetModelHierarchies": [
  {
    "id": "a1b2c3d4-5678-90ab-cdef-77777EXAMPLE",
    "name": "Wind Turbines",
    "childAssetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
  }
]
}

```

输出：

```

{
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
  "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
  "assetModelStatus": {
    "state": "CREATING"
  }
}

```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[更新资产模型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAssetModel](#)中的。

## update-asset-property

以下代码示例演示了如何使用 update-asset-property。

### AWS CLI

示例 1：更新资产属性的别名

以下update-asset-property示例更新了风力涡轮机资产的功率属性别名。

```

aws iotsitewise update-asset-property \
  --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE \
  --property-id a1b2c3d4-5678-90ab-cdef-55555EXAMPLE \

```

```
--property-alias "/examplecorp/windfarm/1/turbine/1/power" \  
--property-notification-state DISABLED
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的将[工业数据流映射到资产属性](#)。

#### 示例 2：启用资产属性通知

以下update-asset-property示例为风力涡轮机资产的功率属性启用资产属性更新通知。属性值更新将发布到 MQTT 主题\$aws/sitewise/asset-models/<assetModelId>/assets/<assetId>/properties/<propertyId>，其中每个 ID 都替换为资产属性的属性、资产和模型 ID。

```
aws iotsitewise update-asset-property \  
  --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE \  
  --property-id a1b2c3d4-5678-90ab-cdef-66666EXAMPLE \  
  --property-notification-state ENABLED \  
  --property-alias "/examplecorp/windfarm/1/turbine/1/power"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[与其他服务交互](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAssetProperty](#)中的。

## update-asset

以下代码示例演示了如何使用 update-asset。

### AWS CLI

#### 更新资产的名称

以下update-asset示例更新了风力涡轮机资产的名称。

```
aws iotsitewise update-asset \  
  --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE \  
  --asset-name "Wind Turbine 2"
```

输出：



```
{
  "assetStatus": {
    "state": "UPDATING"
  }
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[更新资产](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAsset](#)中的。

## update-dashboard

以下代码示例演示了如何使用 update-dashboard。

### AWS CLI

#### 更新控制面板

以下update-dashboard示例更改了显示风力发电场总发电量的仪表板折线图的标题。

```
aws iotsitewise update-dashboard \
  --project-id a1b2c3d4-5678-90ab-cdef-fffffEXAMPLE \
  --dashboard-name "Wind Farm" \
  --dashboard-definition file://update-wind-farm-dashboard.json
```

update-wind-farm-dashboard.json 的内容：

```
{
  "widgets": [
    {
      "type": "monitor-line-chart",
      "title": "Total Generated Power",
      "x": 0,
      "y": 0,
      "height": 3,
      "width": 3,
      "metrics": [
        {
          "label": "Power",
          "type": "iotsitewise",
          "assetId": "a1b2c3d4-5678-90ab-cdef-44444EXAMPLE",
          "propertyId": "a1b2c3d4-5678-90ab-cdef-99999EXAMPLE"
        }
      ]
    }
  ]
}
```

```

    }
  ]
}

```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[创建仪表板 \(CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDashboard](#)中的。

## update-gateway-capability-configuration

以下代码示例演示了如何使用 update-gateway-capability-configuration。

### AWS CLI

#### 更新网关功能

以下update-gateway-capability-configuration示例使用以下属性配置 OPC-UA 源：

信任任何证书。使用 Basic256 算法保护消息。使用该 SignAndEncrypt 模式保护连接。使用存储在 Secrets Manager 密钥中的身份验证凭据。 AWS

```

aws iotsitewise update-gateway-capability-configuration \
  --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE \
  --capability-namespace "iotsitewise:opcuacollector:1" \
  --capability-configuration file://opc-ua-capability-configuration.json

```

opc-ua-capability-configuration.json 的内容：

```

{
  "sources": [
    {
      "name": "Wind Farm #1",
      "endpoint": {
        "certificateTrust": {
          "type": "TrustAny"
        },
        "endpointUri": "opc.tcp://203.0.113.0:49320",
        "securityPolicy": "BASIC256",

```

```

        "messageSecurityMode": "SIGN_AND_ENCRYPT",
        "identityProvider": {
            "type": "Username",
            "usernameSecretArn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:greenrass-windfarm1-auth-1ABCDE"
        },
        "nodeFilterRules": []
    },
    "measurementDataStreamPrefix": ""
}
]
}

```

输出：

```

{
  "capabilityNamespace": "iotsitewise:opcuacollector:1",
  "capabilitySyncStatus": "OUT_OF_SYNC"
}

```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[配置数据源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateGatewayCapabilityConfiguration](#)中的。

## update-gateway

以下代码示例演示了如何使用 update-gateway。

### AWS CLI

更新网关的名称

以下update-gateway示例更新了网关的名称。

```

aws iotsitewise update-gateway \
  --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE \
  --gateway-name ExampleCorpGateway1

```

此命令不生成任何输出。

有关更多信息，请参阅《物AWS 联网 SiteWise 用户指南》中的[使用网关摄取数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateGateway](#)中的。

## update-portal

以下代码示例演示了如何使用 update-portal。

### AWS CLI

更新门户的详细信息

以下update-portal示例更新了一家风电场公司的 Web 门户。

```
aws iotsitewise update-portal \  
  --portal-id a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE \  
  --portal-name WindFarmPortal \  
  --portal-description "A portal that contains wind farm projects for Example  
Corp." \  
  --portal-contact-email support@example.com \  
  --role-arn arn:aws:iam::123456789012:role/MySiteWiseMonitorServiceRole
```

输出：

```
{  
  "portalStatus": {  
    "state": "UPDATING"  
  }  
}
```

有关更多信息，请参阅《AWS 物联网 SiteWise 用户指南》中的[管理门户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePortal](#)中的。

## update-project

以下代码示例演示了如何使用 update-project。

### AWS CLI

更新项目的详细信息

以下update-project示例更新了一个风力发电场项目。

```
aws iotsitewise update-project \  
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeeeEXAMPLE \  
  --project-name "Wind Farm 1" \  
  --role-arn arn:aws:iam::123456789012:role/MySiteWiseMonitorServiceRole
```

```
--project-description "Contains asset visualizations for Wind Farm #1 for Example Corp."
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 物联网 SiteWise 监控器应用指南》中的[更改项目详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateProject](#)中的。

## AWS IoT Things Graph 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS IoT Things Graph。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-entity-to-thing**

以下代码示例演示了如何使用 associate-entity-to-thing。

AWS CLI

将事物与设备关联

以下associate-entity-to-thing示例将事物与设备关联。该示例使用公共命名空间中的运动传感器设备。

```
aws iotthingsgraph associate-entity-to-thing \  
  --thing-name "MotionSensorName" \  
  --entity-id "urn:tdm:aws/examples:Device:HCSR501MotionSensor"
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT Things Graph 用户指南中的[创建和上传模型](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[AssociateEntityToThing](#)中的。

## create-flow-template

以下代码示例演示了如何使用 create-flow-template。

### AWS CLI

#### 创建流程

以下 create-flow-template 示例创建了一个流程（工作流程）。的值 MyFlowDefinition 是对流程进行建模的 GraphQL。

```
aws iotthingsgraph create-flow-template \  
  --definition language=GRAPHQL,text="MyFlowDefinition"
```

输出：

```
{  
  "summary": {  
    "createdAt": 1559248067.545,  
    "id": "urn:tdm:us-west-2/123456789012/default:Workflow:MyFlow",  
    "revisionNumber": 1  
  }  
}
```

有关更多信息，请参阅《AWS IoT Things Graph 用户指南》[中的使用流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateFlowTemplate](#)中的。

## create-system-instance

以下代码示例演示了如何使用 create-system-instance。

### AWS CLI

#### 创建系统实例

以下 `create-system-instance` 示例创建了一个系统实例。的值 `MySystemInstanceDefinition` 是对系统实例进行建模的 GraphQL。

```
aws iotthingsgraph create-system-instance -\
  --definition language=GRAPHQL,text="MySystemInstanceDefinition" \
  --target CLOUD \
  --flow-actions-role-arn myRoleARN
```

输出：

```
{
  "summary": {
    "id": "urn:tdm:us-west-2/123456789012/default:Deployment:Room218",
    "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:Deployment/default/Room218",
    "status": "NOT_DEPLOYED",
    "target": "CLOUD",
    "createdAt": 1559249315.208,
    "updatedAt": 1559249315.208
  }
}
```

有关更多信息，请参阅 [AWS IoT Things Graph 用户指南中的使用系统和流程配置](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `CreateSystemInstance`](#) 中的。

## `create-system-template`

以下代码示例演示了如何使用 `create-system-template`。

### AWS CLI

#### 创建系统

以下 `create-system-template` 示例创建了一个系统。的值 `MySystemDefinition` 是对系统进行建模的 GraphQL。

```
aws iotthingsgraph create-system-template \
  --definition language=GRAPHQL,text="MySystemDefinition"
```

输出：

```
{
  "summary": {
    "createdAt": 1559249776.254,
    "id": "urn:tdm:us-west-2/123456789012/default:System:MySystem",
    "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:System/default/MySystem",
    "revisionNumber": 1
  }
}
```

有关更多信息，请参阅《AWS IoT Things Graph 用户指南》中的[创建系统](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSystemTemplate](#)中的。

## delete-flow-template

以下代码示例演示了如何使用 delete-flow-template。

### AWS CLI

#### 删除流程

以下delete-flow-template示例删除流程（工作流程）。

```
aws iotthingsgraph delete-flow-template \
  --id "urn:tdm:us-west-2/123456789012/default:Workflow:MyFlow"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS IoT Things Graph 用户指南](#)》中的 [IoT Things Graph Graph 实体、流程、系统和部署的生命周期管理](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFlowTemplate](#)中的。

## delete-namespace

以下代码示例演示了如何使用 delete-namespace。

### AWS CLI

#### 删除命名空间

以下delete-namespace示例删除命名空间。



```
aws iotthingsgraph delete-namespace
```

输出：

```
{
  "namespaceArn": "arn:aws:iotthingsgraph:us-west-2:123456789012",
  "namespaceName": "us-west-2/123456789012/default"
}
```

有关更多信息，请参阅 [《AWS IoT Things Graph 用户指南》中的 IoT Things Graph Graph 实体、流程、系统和部署的生命周期管理](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNamespace](#)中的。

## delete-system-instance

以下代码示例演示了如何使用 delete-system-instance。

AWS CLI

删除系统实例

以下delete-system-instance示例删除系统实例。

```
aws iotthingsgraph delete-system-instance \
  --id "urn:tdm:us-west-2/123456789012/default:Deployment:Room218"
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS IoT Things Graph 用户指南》中的 IoT Things Graph Graph 实体、流程、系统和部署的生命周期管理](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSystemInstance](#)中的。

## delete-system-template

以下代码示例演示了如何使用 delete-system-template。

AWS CLI

删除系统

以下delete-system-template示例删除了一个系统。

```
aws iotthingsgraph delete-system-template \  
  --id "urn:tdm:us-west-2/123456789012/default:System:MySystem"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS IoT Things Graph 用户指南](#)》中的 [IoT Things Graph Graph 实体、流程、系统和部署的生命周期管理](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSystemTemplate](#)中的。

## deploy-system-instance

以下代码示例演示了如何使用 deploy-system-instance。

### AWS CLI

#### 部署系统实例

以下delete-system-template示例部署了一个系统实例。

```
aws iotthingsgraph deploy-system-instance \  
  --id "urn:tdm:us-west-2/123456789012/default:Deployment:Room218"
```

输出：

```
{  
  "summary": {  
    "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:Deployment:Room218",  
    "createdAt": 1559249776.254,  
    "id": "urn:tdm:us-west-2/123456789012/default:Deployment:Room218",  
    "status": "DEPLOYED_IN_TARGET",  
    "target": "CLOUD",  
    "updatedAt": 1559249776.254  
  }  
}
```

有关更多信息，请参阅 [AWS IoT Things Graph 用户指南](#)中的[使用系统和流程配置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeploySystemInstance](#)中的。

## deprecate-flow-template

以下代码示例演示了如何使用 `deprecate-flow-template`。

### AWS CLI

#### 弃用流程

以下`deprecate-flow-template`示例弃用了流程（工作流程）。

```
aws iotthingsgraph deprecate-flow-template \  
  --id "urn:tdm:us-west-2/123456789012/default:Workflow:MyFlow"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS IoT Things Graph 用户指南](#)》中的 [IoT Things Graph Graph 实体、流程、系统和部署的生命周期管理](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeprecateFlowTemplate](#)中的。

## deprecate-system-template

以下代码示例演示了如何使用 `deprecate-system-template`。

### AWS CLI

#### 弃用系统

以下`deprecate-system-template`示例不推荐使用系统。

```
aws iotthingsgraph deprecate-system-template \  
  --id "urn:tdm:us-west-2/123456789012/default:System:MySystem"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS IoT Things Graph 用户指南](#)》中的 [IoT Things Graph Graph 实体、流程、系统和部署的生命周期管理](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeprecateSystemTemplate](#)中的。

## describe-namespace

以下代码示例演示了如何使用 `describe-namespace`。

## AWS CLI

### 获取命名空间的描述

以下describe-namespace示例获取了您的命名空间的描述。

```
aws iotthingsgraph describe-namespace
```

输出：

```
{
  "namespaceName": "us-west-2/123456789012/default",
  "trackingNamespaceName": "aws",
  "trackingNamespaceVersion": 1,
  "namespaceVersion": 5
}
```

有关更多信息，请参阅 AWS IoT Things Graph 用户指南中的[命名空间](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeNamespace](#)中的。

## dissociate-entity-from-thing

以下代码示例演示了如何使用 dissociate-entity-from-thing。

### AWS CLI

#### 将事物与设备断开关联

以下dissociate-entity-from-thing示例将事物与设备解除关联。

```
aws iotthingsgraph dissociate-entity-from-thing \
  --thing-name "MotionSensorName" \
  --entity-type "DEVICE"
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS IoT Things Graph 用户指南中的[创建和上传模型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DissociateEntityFromThing](#)中的。

## get-entities

以下代码示例演示了如何使用 get-entities。

### AWS CLI

获取实体的定义

以下get-entities示例获取了设备型号的定义。

```
aws iotthingsgraph get-entities \  
  --ids "urn:tdm:aws/examples:DeviceModel:MotionSensor"
```

输出：

```
{  
  "descriptions": [  
    {  
      "id": "urn:tdm:aws/examples:DeviceModel:MotionSensor",  
      "type": "DEVICE_MODEL",  
      "createdAt": 1559256190.599,  
      "definition": {  
        "language": "GRAPHQL",  
        "text": "##\n# Specification of motion sensor devices interface.\n##  
\n#type MotionSensor @deviceModel(id: \"urn:tdm:aws/examples:deviceModel:MotionSensor  
\",\n      capability: \"urn:tdm:aws/examples:capability:MotionSensorCapability\")  
      {ignore:void}"  
      }  
    }  
  ]  
}
```

有关更多信息，请参阅 AWS IoT Things Graph 用户指南中的[创建和上传模型](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetEntities](#)中的。

## get-flow-template-revisions

以下代码示例演示了如何使用 get-flow-template-revisions。

### AWS CLI

获取有关流程的修订信息

以下`get-flow-template-revisions`示例获取有关流程（工作流程）的修订信息。

```
aws iotthingsgraph get-flow-template-revisions \  
  --id urn:tdm:us-west-2/123456789012/default:Workflow:MyFlow
```

输出：

```
{  
  "summaries": [  
    {  
      "id": "urn:tdm:us-west-2/123456789012/default:Workflow:MyFlow",  
      "revisionNumber": 1,  
      "createdAt": 1559247540.292  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS IoT Things Graph 用户指南》[中的使用流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFlowTemplateRevisions](#)中的。

## get-flow-template

以下代码示例演示了如何使用 `get-flow-template`。

AWS CLI

获取流程定义

以下`get-flow-template`示例获取流程（工作流程）的定义。

```
aws iotthingsgraph get-flow-template \  
  --id "urn:tdm:us-west-2/123456789012/default:Workflow:MyFlow"
```

输出：

```
{  
  "description": {  
    "summary": {  
      "id": "urn:tdm:us-west-2/123456789012/default:Workflow:MyFlow",  
      "revisionNumber": 1,  

```

```

      "createdAt": 1559247540.292
    },
    "definition": {
      "language": "GraphQL",
      "text": "{\nquery MyFlow($camera: string!, $screen: string!)
@workflowType(id: \"urn:tdm:us-west-2/123456789012/default:Workflow:MyFlow\")
@annotation(type: \"tgc:FlowEvent\", id: \"sledged790c1b2bcd949e09da0c9bfc077f79d
\", x: 1586, y: 653) @triggers(definition: \"{MotionSensor(description:
\\\\"\\\\"") @position(x: 1045, y: 635.6666564941406) {\n  condition(expr:
\\\\"devices[name == \\\\\"motionSensor\\\\"].events[name == \\\\\"
\\\\"StateChanged\\\\"].lastEvent\\\\"})\n  action(expr: \\\\\"\\\\"})\n
}}\n") {\n  variables {\n    cameraResult @property(id: \"urn:tdm:aws/
examples:property:CameraStateProperty\")\n  }\n  steps {\n    step(name: \"Camera
\", outEvent: [\"sledged790c1b2bcd949e09da0c9bfc077f79d\"] @position(x: 1377,
y: 638.6666564941406) {\n      DeviceActivity(deviceModel: \"urn:tdm:aws/
examples:deviceModel:Camera\", out: \"cameraResult\", deviceId: \"${camera}\")
{\n        capture\n      }\n    }\n    step(name: \"Screen\", inEvent:
[\"sledged790c1b2bcd949e09da0c9bfc077f79d\"] @position(x: 1675.6666870117188,
y: 637.9999847412109) {\n      DeviceActivity(deviceModel: \"urn:tdm:aws/
examples:deviceModel:Screen\", deviceId: \"${screen}\") {\n        display(imageUrl:
\"${cameraResult.lastClickedImage}\")\n      }\n    }\n  }\n}\n}\n}"}
    },
    "validatedNamespaceVersion": 5
  }
}

```

有关更多信息，请参阅《AWS IoT Things Graph 用户指南》中的[使用流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetFlowTemplate](#) 中的。

## get-namespace-deletion-status

以下代码示例演示了如何使用 `get-namespace-deletion-status`。

### AWS CLI

获取命名空间删除任务的状态

以下 `get-namespace-deletion-status` 示例获取命名空间删除任务的状态。

```
aws iotthingsgraph get-namespace-deletion-status
```

输出：

```
{
  "namespaceArn": "arn:aws:iotthingsgraph:us-west-2:123456789012",
  "namespaceName": "us-west-2/123456789012/default"
  "status": "SUCCEEDED "
}
```

有关更多信息，请参阅 AWS IoT Things Graph 用户指南中的[命名空间](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetNamespaceDeletionStatus](#)中的。

## get-system-instance

以下代码示例演示了如何使用 get-system-instance。

### AWS CLI

#### 获取系统实例

以下 get-system-instance 示例获取了系统实例的定义。

```
aws iotthingsgraph get-system-instance \
  --id "urn:tdm:us-west-2/123456789012/default:Deployment:Room218"
```

输出：

```
{
  "description": {
    "summary": {
      "id": "urn:tdm:us-west-2/123456789012/default:Deployment:Room218",
      "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:Deployment/default/Room218",
      "status": "NOT_DEPLOYED",
      "target": "CLOUD",
      "createdAt": 1559249315.208,
      "updatedAt": 1559249315.208
    },
    "definition": {
      "language": "GRAPHQL",
      "text": "{\r\nquery Room218 @deployment(id: \"urn:tdm:us-west-2/123456789012/default:Deployment:Room218\", systemId: \"urn:tdm:us-west-2/123456789012/default:System:SecurityFlow\") {\r\n  motionSensor(deviceId: \"MotionSensorName\")\r\n  screen(deviceId: \"ScreenName\")\r\n}
```



```
camera(deviceId: \"CameraName\") \r\n    triggers {MotionEventTrigger(description:
\"a trigger\") { \r\n    condition(expr: \"devices[name ==
'motionSensor'].events[name == 'StateChanged'].lastEvent\") \r\n    action(expr:
\"ThingsGraph.startFlow('SecurityFlow', bindings[name == 'camera'].deviceId,
bindings[name == 'screen'].deviceId)\")\r\n    }\r\n    }\r\n    }\r\n    }"
    },
    "metricsConfiguration": {
        "cloudMetricEnabled": false
    },
    "validatedNamespaceVersion": 5,
    "flowActionsRoleArn": "arn:aws:iam::123456789012:role/ThingsGraphRole"
}
}
```

有关更多信息，请参阅 [AWS IoT Things Graph 用户指南中的使用系统和流程配置](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `GetSystemInstance`](#) 中的。

## get-system-template-revisions

以下代码示例演示了如何使用 `get-system-template-revisions`。

### AWS CLI

获取有关系统的版本信息

以下 `get-system-template-revisions` 示例获取有关系统的修订信息。

```
aws iotthingsgraph get-system-template-revisions \
  --id "urn:tdm:us-west-2/123456789012/default:System:MySystem"
```

输出：

```
{
  "summaries": [
    {
      "id": "urn:tdm:us-west-2/123456789012/default:System:MySystem",
      "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:System/default/MySystem",
      "revisionNumber": 1,
      "createdAt": 1559247540.656
    }
  ]
}
```

```
}
```

有关更多信息，请参阅 [AWS IoT Things Graph 用户指南中的使用系统和流程配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetSystemTemplateRevisions](#) 中的。

## get-system-template

以下代码示例演示了如何使用 `get-system-template`。

### AWS CLI

要获得系统

以下 `get-system-template` 示例获取了系统的定义。

```
aws iotthingsgraph get-system-template \  
  --id "urn:tdm:us-west-2/123456789012/default:System:MySystem"
```

输出：

```
{  
  "description": {  
    "summary": {  
      "id": "urn:tdm:us-west-2/123456789012/default:System:MySystem",  
      "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:System/default/  
MyFlow",  
      "revisionNumber": 1,  
      "createdAt": 1559247540.656  
    },  
    "definition": {  
      "language": "GRAPHQL",  
      "text": "{\n  type MySystem @systemType(id: \"urn:tdm:us-  
west-2/123456789012/default:System:MySystem\", description: \"\") {\n    camera:  
      Camera @thing(id: \"urn:tdm:aws/examples:deviceModel:Camera\")\n    screen:  
      Screen @thing(id: \"urn:tdm:aws/examples:deviceModel:Screen\")\n    motionSensor:  
      MotionSensor @thing(id: \"urn:tdm:aws/examples:deviceModel:MotionSensor  
\")\n    MyFlow: MyFlow @workflow(id: \"urn:tdm:us-west-2/123456789012/  
default:Workflow:MyFlow\")\n  }\n}"  
    },  
      "validatedNamespaceVersion": 5  
    }  
  }  
}
```

有关更多信息，请参阅 AWS IoT Things Graph 用户指南中的[使用系统和流程配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetSystemTemplate](#)中的。

## get-upload-status

以下代码示例演示了如何使用 get-upload-status。

### AWS CLI

要获取实体的状态，请上传

以下 get-upload-status 示例获取您的实体上传操作的状态。的值 MyUploadId 是 upload-entity-definitions 操作返回的 ID 值。

```
aws iotthingsgraph get-upload-status \  
  --upload-id "MyUploadId"
```

输出：

```
{  
  "namespaceName": "us-west-2/123456789012/default",  
  "namespaceVersion": 5,  
  "uploadId": "f6294f1e-b109-4bbe-9073-f451a2dda2da",  
  "uploadStatus": "SUCCEEDED"  
}
```

有关更多信息，请参阅《AWS IoT Things Graph 用户指南》中的[建模实体](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetUploadStatus](#)中的。

## list-flow-execution-messages

以下代码示例演示了如何使用 list-flow-execution-messages。

### AWS CLI

获取有关流程执行中事件的信息

以下 list-flow-execution-messages 示例获取有关流程执行中事件的信息。

```
aws iotthingsgraph list-flow-execution-messages \  
  --namespace-name "us-west-2/123456789012/default" \  
  --namespace-version 5
```

```
--flow-execution-id "urn:tdm:us-west-2/123456789012/  
default:Workflow:SecurityFlow_2019-05-11T19:39:55.317Z_MotionSensor_69b151ad-  
a611-42f5-ac21-fe537f9868ad"
```

输出：

```
{  
  "messages": [  
    {  
      "eventType": "EXECUTION_STARTED",  
      "messageId": "f6294f1e-b109-4bbe-9073-f451a2dda2da",  
      "payload": "Flow execution started",  
      "timestamp": 1559247540.656  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS IoT Things Graph 用户指南》[中的使用流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFlowExecutionMessages](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 list-tags-for-resource。

### AWS CLI

列出资源的所有标签

以下list-tags-for-resource示例列出了 AWS IoT Things Graph 资源的所有标签。

```
aws iotthingsgraph list-tags-for-resource \  
  --resource-arn "arn:aws:iotthingsgraph:us-west-2:123456789012:Deployment/  
  default/Room218"
```

输出：

```
{  
  "tags": [  
    {  
      "key": "Type",  
      "value": "Residential"    }  
  ]  
}
```

```
    }
  ]
}
```

有关更多信息，请参阅《[AWS IoT Things Graph 用户指南](#)》中的为AWS 物联网事物图表资源添加[标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## search-entities

以下代码示例演示了如何使用 search-entities。

### AWS CLI

#### 搜索实体

以下search-entities示例搜索类型的所有实体EVENT。

```
aws iotthingsgraph search-entities \
  --entity-types "EVENT"
```

输出：

```
{
  "descriptions": [
    {
      "id": "urn:tdm:aws/examples:Event:MotionSensorEvent",
      "type": "EVENT",
      "definition": {
        "language": "GRAPHQL",
        "text": "##\n# Description of events emitted by motion
sensor.\n##\n# type MotionSensorEvent @eventType(id: \"urn:tdm:aws/
examples:event:MotionSensorEvent\", \n          payload: \"urn:tdm:aws/
examples:property:MotionSensorStateProperty\") {ignore:void}"
      }
    },
    {
      "id": "urn:tdm:us-west-2/123456789012/
default:Event:CameraClickedEventV2",
      "type": "EVENT",
      "definition": {
        "language": "GRAPHQL",
```

```

      "text": "type CameraClickedEventV2 @eventType(id: \"urn:tdm:us-
west-2/123456789012/default:event:CameraClickedEventV2\", \r\npayload:
 \"urn:tdm:aws:Property:Boolean\") {ignore:void}"
    }
  },
  {
    "id": "urn:tdm:us-west-2/123456789012/
default:Event:MotionSensorEventV2",
    "type": "EVENT",
    "definition": {
      "language": "GraphQL",
      "text": "# Event emitted by the motion sensor.\r\nrtype
MotionSensorEventV2 @eventType(id: \"urn:tdm:us-west-2/123456789012/
default:event:MotionSensorEventV2\", \r\npayload: \"urn:tdm:us-west-2/123456789012/
default:property:MotionSensorStateProperty2\") {ignore:void}"
    }
  }
],
"nextToken": "urn:tdm:us-west-2/123456789012/default:Event:MotionSensorEventV2"
}

```

有关更多信息，请参阅 [《AWS IoT Things Graph 用户指南》](#) 中的 [AWS IoT Things Graph 数据模型参考](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SearchEntities](#) 中的。

## search-flow-executions

以下代码示例演示了如何使用 search-flow-executions。

### AWS CLI

#### 搜索流程执行

以下 search-flow-executions 示例搜索指定系统实例中某个流程的所有执行。

```

aws iotthingsgraph search-flow-executions \
  --system-instance-id "urn:tdm:us-west-2/123456789012/default:Deployment:Room218"

```

输出：

```

{
  "summaries": [

```

```

    {
      "createdAt": 1559247540.656,
      "flowExecutionId": "f6294f1e-b109-4bbe-9073-f451a2dda2da",
      "flowTemplateId": "urn:tdm:us-west-2/123456789012/default:Workflow:MyFlow",
      "status": "RUNNING ",
      "systemInstanceId": "urn:tdm:us-west-2/123456789012/
default:System:MySystem",
      "updatedAt": 1559247540.656
    }
  ]
}

```

有关更多信息，请参阅 AWS IoT Things Graph 用户指南 [中的使用系统和流程配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SearchFlowExecutions](#) 中的。

## search-flow-templates

以下代码示例演示了如何使用 search-flow-templates。

### AWS CLI

#### 搜索流程 ( 或工作流程 )

以下 search-flow-templates 示例搜索包含相机设备型号的所有流程 ( 工作流程 )。

```

aws iotthingsgraph search-flow-templates \
  --filters name="DEVICE_MODEL_ID",value="urn:tdm:aws/examples:DeviceModel:Camera"

```

输出：

```

{
  "summaries": [
    {
      "id": "urn:tdm:us-west-2/123456789012/default:Workflow:MyFlow",
      "revisionNumber": 1,
      "createdAt": 1559247540.292
    },
    {
      "id": "urn:tdm:us-west-2/123456789012/default:Workflow:SecurityFlow",
      "revisionNumber": 3,
      "createdAt": 1548283099.27
    }
  ]
}

```

```
]
}
```

有关更多信息，请参阅《AWS IoT Things Graph 用户指南》中的[使用流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SearchFlowTemplates](#) 中的。

## search-system-instances

以下代码示例演示了如何使用 search-system-instances。

### AWS CLI

#### 搜索系统实例

以下 search-system-instances 示例搜索包含指定系统的所有系统实例。

```
aws iotthingsgraph search-system-instances \
  --filters name="SYSTEM_TEMPLATE_ID",value="urn:tdm:us-west-2/123456789012/
  default:System:SecurityFlow"
```

输出：

```
{
  "summaries": [
    {
      "id": "urn:tdm:us-west-2/123456789012/
      default:Deployment:DeploymentForSample",
      "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:Deployment/
      default/DeploymentForSample",
      "status": "NOT_DEPLOYED",
      "target": "GREENGRASS",
      "greengrassGroupName": "ThingsGraphGrnGr",
      "createdAt": 1555716314.707,
      "updatedAt": 1555716314.707
    },
    {
      "id": "urn:tdm:us-west-2/123456789012/
      default:Deployment:MockDeployment",
      "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:Deployment/
      default/MockDeployment",
      "status": "DELETED_IN_TARGET",
      "target": "GREENGRASS",
```



```

    "greengrassGroupName": "ThingsGraphGrnGr",
    "createdAt": 1549416462.049,
    "updatedAt": 1549416722.361,
    "greengrassGroupId": "01d04b07-2a51-467f-9d03-0c90b3cdcaaf",
    "greengrassGroupVersionId": "7365aed7-2d3e-4d13-aad8-75443d45eb05"
  },
  {
    "id": "urn:tdm:us-west-2/123456789012/
default:Deployment:MockDeployment2",
    "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:Deployment/
default/MockDeployment2",
    "status": "DEPLOYED_IN_TARGET",
    "target": "GREENGRASS",
    "greengrassGroupName": "ThingsGraphGrnGr",
    "createdAt": 1549572385.774,
    "updatedAt": 1549572418.408,
    "greengrassGroupId": "01d04b07-2a51-467f-9d03-0c90b3cdcaaf",
    "greengrassGroupVersionId": "bfa70ab3-2bf7-409c-a4d4-bc8328ae5b86"
  },
  {
    "id": "urn:tdm:us-west-2/123456789012/default:Deployment:Room215",
    "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:Deployment/
default/Room215",
    "status": "NOT_DEPLOYED",
    "target": "GREENGRASS",
    "greengrassGroupName": "ThingsGraphGG",
    "createdAt": 1547056918.413,
    "updatedAt": 1547056918.413
  },
  {
    "id": "urn:tdm:us-west-2/123456789012/default:Deployment:Room218",
    "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:Deployment/
default/Room218",
    "status": "NOT_DEPLOYED",
    "target": "CLOUD",
    "createdAt": 1559249315.208,
    "updatedAt": 1559249315.208
  }
]
}

```

有关更多信息，请参阅 [AWS IoT Things Graph 用户指南中的使用系统和流程配置](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 SearchSystemInstances](#) 中的。

## search-system-templates

以下代码示例演示了如何使用 `search-system-templates`。

### AWS CLI

#### 要搜索系统

以下 `search-system-templates` 示例搜索包含指定流程的所有系统。

```
aws iotthingsgraph search-system-templates \  
  --filters name="FLOW_TEMPLATE_ID",value="urn:tdm:us-west-2/123456789012/  
  default:Workflow:SecurityFlow"
```

输出：

```
{  
  "summaries": [  
    {  
      "id": "urn:tdm:us-west-2/123456789012/default:System:SecurityFlow",  
      "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:System/default/  
SecurityFlow",  
      "revisionNumber": 1,  
      "createdAt": 1548283099.433  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS IoT Things Graph 用户指南》中的[使用流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SearchSystemTemplates](#) 中的。

## search-things

以下代码示例演示了如何使用 `search-things`。

### AWS CLI

搜索与设备和设备型号相关的内容

以下 `search-things` 示例搜索与 HCSR501 MotionSensor 设备关联的所有内容。

```
aws iotthingsgraph search-things \  
  --filters name="FLOW_TEMPLATE_ID",value="urn:tdm:us-west-2/123456789012/  
  default:Workflow:SecurityFlow"
```

```
--entity-id "urn:tdm:aws/examples:Device:HCSR501MotionSensor"
```

输出：

```
{
  "things": [
    {
      "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MotionSensor1",
      "thingName": "MotionSensor1"
    },
    {
      "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/TG_MS",
      "thingName": "TG_MS"
    }
  ]
}
```

有关更多信息，请参阅 AWS IoT Things Graph 用户指南中的[创建和上传模型](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[Search Things](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源创建标签

以下 tag-resource 示例为指定资源创建标签。

```
aws iotthingsgraph tag-resource \
  --resource-arn "arn:aws:iotthingsgraph:us-west-2:123456789012:Deployment/
  default/Room218" \
  --tags key="Type",value="Residential"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS IoT Things Graph 用户指南](#)》中的[为 AWS 物联网事物图表资源添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[TagResource](#)中的。

## undeploy-system-instance

以下代码示例演示了如何使用 `undeploy-system-instance`。

### AWS CLI

将系统实例从其目标上取消部署

以下 `undeploy-system-instance` 示例将系统实例从其目标中移除。

```
aws iotthingsgraph undeploy-system-instance \  
  --id "urn:tdm:us-west-2/123456789012/default:Deployment:Room215"
```

输出：

```
{  
  "summary": {  
    "id": "urn:tdm:us-west-2/123456789012/default:Deployment:Room215",  
    "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:Deployment/default/  
Room215",  
    "status": "PENDING_DELETE",  
    "target": "GREENGRASS",  
    "greengrassGroupName": "ThingsGraphGrnGr",  
    "createdAt": 1553189694.255,  
    "updatedAt": 1559344549.601,  
    "greengrassGroupId": "01d04b07-2a51-467f-9d03-0c90b3cdcaaf",  
    "greengrassGroupVersionId": "731b371d-d644-4b67-ac64-3934e99b75d7"  
  }  
}
```

有关更多信息，请参阅 [《AWS IoT Things Graph 用户指南》中的 IoT Things Graph 实体、流程、系统和部署的生命周期管理](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UndeploySystemInstance](#) 中的。

## untag-resource

以下代码示例演示了如何使用 `untag-resource`。

### AWS CLI

移除资源的标签

以下 `untag-resource` 示例删除了指定资源的标签。

```
aws iotthingsgraph untag-resource \  
  --resource-arn "arn:aws:iotthingsgraph:us-west-2:123456789012:Deployment/  
default/Room218" \  
  --tag-keys "Type"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS IoT Things Graph 用户指南](#)》中的[为AWS 物联网事物图表资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-flow-template

以下代码示例演示了如何使用 `update-flow-template`。

### AWS CLI

#### 更新流程

以下 `update-flow-template` 示例更新流程（工作流程）。的值 `MyFlowDefinition` 是对流程进行建模的 GraphQL。

```
aws iotthingsgraph update-flow-template \  
  --id "urn:tdm:us-west-2/123456789012/default:Workflow:MyFlow" \  
  --definition language=GRAPHQL,text="MyFlowDefinition"
```

输出：

```
{  
  "summary": {  
    "createdAt": 1559248067.545,  
    "id": "urn:tdm:us-west-2/123456789012/default:Workflow:MyFlow",  
    "revisionNumber": 2  
  }  
}
```

有关更多信息，请参阅《[AWS IoT Things Graph 用户指南](#)》中的[使用流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateFlowTemplate](#)中的。

## update-system-template

以下代码示例演示了如何使用 `update-system-template`。

### AWS CLI

#### 更新系统

以下 `update-system-template` 示例更新系统。的值 `MySystemDefinition` 是对系统进行建模的 GraphQL。

```
aws iotthingsgraph update-system-template \  
  --id "urn:tdm:us-west-2/123456789012/default:System:MySystem" \  
  --definition language=GRAPHQL,text="MySystemDefinition"
```

输出：

```
{  
  "summary": {  
    "createdAt": 1559249776.254,  
    "id": "urn:tdm:us-west-2/123456789012/default:System:MySystem",  
    "arn": "arn:aws:iotthingsgraph:us-west-2:123456789012:System/default/  
MySystem",  
    "revisionNumber": 2  
  }  
}
```

有关更多信息，请参阅《AWS IoT Things Graph 用户指南》中的[创建系统](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateSystemTemplate](#) 中的。

## upload-entity-definitions

以下代码示例演示了如何使用 `upload-entity-definitions`。

### AWS CLI

#### 上传实体定义

以下 `upload-entity-definitions` 示例将实体定义上传到您的命名空间。的值 `MyEntityDefinitions` 是对实体进行建模的 GraphQL。

```
aws iotthingsgraph upload-entity-definitions \  
  --document language=GRAPHQL,text="MyEntityDefinitions"
```

输出：

```
{  
  "uploadId": "f6294f1e-b109-4bbe-9073-f451a2dda2da"  
}
```

有关更多信息，请参阅《AWS IoT Things Graph 用户指南》中的[建模实体](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UploadEntityDefinitions](#)中的。

## AWS IoT Wireless 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS IoT Wireless。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-aws-account-with-partner-account**

以下代码示例演示了如何使用 `associate-aws-account-with-partner-account`。

AWS CLI

将合作伙伴账户与您的 AWS 账户关联

以下 `associate-aws-account-with-partner-account` 示例将以下 Sidewalk 账户凭证与您的 AWS 账户相关联。

```
aws iotwireless associate-aws-account-with-partner-account \  
  --sidewalk  
  AmazonId="12345678901234",AppServerPrivateKey="a123b45c6d78e9f012a34cd5e6a7890b12c3d45e6f78a1b234c56d7e890a1234"
```

输出：

```
{  
  "Sidewalk": {  
    "AmazonId": "12345678901234",  
    "AppServerPrivateKey":  
    "a123b45c6d78e9f012a34cd5e6a7890b12c3d45e6f78a1b234c56d7e890a1234"  
  }  
}
```

有关更多信息，请参阅《物联网开发人员指南》中的 Amazon Sidewalk | AWS IoT Core [集成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateAwsAccountWithPartnerAccount](#) 中的。

## associate-wireless-device-with-thing

以下代码示例演示了如何使用 `associate-wireless-device-with-thing`。

### AWS CLI

将事物与无线设备关联

以下 `associate-wireless-device-with-thing` 示例将事物与具有指定 ID 的无线设备相关联。

```
aws iotwireless associate-wireless-device-with-thing \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d" \  
  --thing-arn "arn:aws:iot:us-east-1:123456789012:thing/MyIoTWirelessThing"
```

此命令不生成任何输出。

有关更多信息，请参阅《物联网开发人员指南》中的 [将您的网关和无线设备添加到 AWS IoT Core for LoRa WAN](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateWirelessDeviceWithThing](#) 中的。



## associate-wireless-gateway-with-certificate

以下代码示例演示了如何使用 `associate-wireless-gateway-with-certificate`。

### AWS CLI

将证书与无线网关关联

以下内容 `associate-wireless-gateway-with-certificate` 将无线网关与证书相关联。

```
aws iotwireless associate-wireless-gateway-with-certificate \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d" \  
  --iot-certificate-id  
  "a123b45c6d78e9f012a34cd5e6a7890b12c3d45e6f78a1b234c56d7e890a1234"
```

输出：

```
{  
  "IotCertificateId":  
  "a123b45c6d78e9f012a34cd5e6a7890b12c3d45e6f78a1b234c56d7e890a1234"  
}
```

有关更多信息，请参阅《[物联网开发人员指南](#)》中的[将您的网关和无线设备添加到 AWS IoT Core for LoRa WAN](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateWirelessGatewayWithCertificate](#) 中的。

## associate-wireless-gateway-with-thing

以下代码示例演示了如何使用 `associate-wireless-gateway-with-thing`。

### AWS CLI

将事物与无线网关关联

以下 `associate-wireless-gateway-with-thing` 示例将事物与无线网关关联。

```
aws iotwireless associate-wireless-gateway-with-thing \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d" \  
  --thing-arn "arn:aws:iot:us-east-1:123456789012:thing/MyIoTWirelessThing"
```

此命令不生成任何输出。

有关更多信息，请参阅《[物联网开发人员指南](#)》中的[将您的网关和无线设备添加到 AWS IoT Core for LoRa WAN](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateWirelessGatewayWithThing](#) 中的。

## create-destination

以下代码示例演示了如何使用 create-destination。

### AWS CLI

创建物联网无线目的地

以下 create-destination 示例创建用于将设备消息映射到 AWS 物联网规则的目标。在运行此命令之前，您必须创建一个 IAM 角色，该角色向适用于 LoRa WAN 的 AWS IoT Core 授予向 AWS 物联网规则发送数据所需的权限。

```
aws iotwireless create-destination \
  --name IoTWirelessDestination \
  --expression-type RuleName \
  --expression IoTWirelessRule \
  --role-arn arn:aws:iam::123456789012:role/IoTWirelessDestinationRole
```

输出：

```
{
  "Arn": "arn:aws:iotwireless:us-east-1:123456789012:Destination/
IoTWirelessDestination",
  "Name": "IoTWirelessDestination"
}
```

有关更多信息，请参阅《[物联网开发人员指南](#)》中的[向 AWS IoT Core for LoRa WAN 添加目标](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDestination](#) 中的。

## create-device-profile

以下代码示例演示了如何使用 create-device-profile。

## AWS CLI

### 创建新的设备配置文件

以下create-device-profile示例创建了新的物联网无线设备配置文件。

```
aws iotwireless create-device-profile
```

输出：

```
{
  "Arn": "arn:aws:iotwireless:us-east-1:123456789012:DeviceProfile/12345678-
a1b2-3c45-67d8-e90fa1b2c34d",
  "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》中的向适用于 LoRa WAN 的AWS IoT Core 添加配置文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDeviceProfile](#)中的。

## create-service-profile

以下代码示例演示了如何使用 create-service-profile。

## AWS CLI

### 创建新的服务配置文件

以下create-service-profile示例创建了新的物联网无线服务配置文件。

```
aws iotwireless create-service-profile
```

输出：

```
{
  "Arn": "arn:aws:iotwireless:us-east-1:123456789012:ServiceProfile/12345678-
a1b2-3c45-67d8-e90fa1b2c34d",
  "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [向适用于 LoRa WAN 的 AWS IoT Core 添加配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateServiceProfile](#) 中的。

## create-wireless-device

以下代码示例演示了如何使用 create-wireless-device。

### AWS CLI

#### 创建 IoT 无线设备

以下 create-wireless-device 示例创建 LoRa WAN 类型的无线设备资源。

```
aws iotwireless create-wireless-device \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{  
  "Description": "My LoRaWAN wireless device"  
  "DestinationName": "IoTWirelessDestination"  
  "LoRaWAN": {  
    "DeviceProfileId": "ab0c23d3-b001-45ef-6a01-2bc3de4f5333",  
    "ServiceProfileId": "fe98dc76-cd12-001e-2d34-5550432da100",  
    "OtaaV1_1": {  
      "AppKey": "3f4ca100e2fc675ea123f4eb12c4a012",  
      "JoinEui": "b4c231a359bc2e3d",  
      "NwkKey": "01c3f004a2d6efffe32c4eda14bcd2b4"  
    },  
    "DevEui": "ac12efc654d23fc2"  
  },  
  "Name": "SampleIoTWirelessThing"  
  "Type": "LoRaWAN"  
}
```

输出：

```
{  
  "Arn": "arn:aws:iotwireless:us-  
east-1:123456789012:WirelessDevice/1fffd32c8-8130-4194-96df-622f072a315f",
```

```
"Id": "1ffd32c8-8130-4194-96df-622f072a315f"
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateWirelessDevice](#) 中的。

## create-wireless-gateway-task-definition

以下代码示例演示了如何使用 `create-wireless-gateway-task-definition`。

### AWS CLI

创建无线网关任务定义

以下内容使用此任务定义 `create-wireless-gateway-task-definition` 自动为所有具有指定当前版本的网关创建任务。

```
aws iotwireless create-wireless-gateway-task-definition \
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{
  "AutoCreateTasks": true,
  "Name": "TestAutoUpdate",
  "Update": {
    "UpdateDataSource" : "s3://cupsalphagafirmwarebin/station",
    "UpdateDataRole" : "arn:aws:iam::001234567890:role/SDK_Test_Role",
    "LoRaWAN" : {
      "CurrentVersion" : {
        "PackageVersion" : "1.0.0",
        "Station" : "2.0.5",
        "Model" : "linux"
      },
      "UpdateVersion" : {
        "PackageVersion" : "1.0.1",
        "Station" : "2.0.5",
        "Model" : "minihub"
      }
    }
  }
}
```

```
    }  
  }  
}
```

输出：

```
{  
  "Id": "b7d3baad-25c7-35e7-a4e1-1683a0d61da9"  
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》中的将设备和网关连接到适用于 LoRa WAN 的AWS IoT Core。](#)

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateWirelessGatewayTaskDefinition](#)中的。

## create-wireless-gateway-task

以下代码示例演示了如何使用 create-wireless-gateway-task。

### AWS CLI

为无线网关创建任务

以下create-wireless-gateway-task示例为无线网关创建任务。

```
aws iotwireless create-wireless-gateway-task \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d" \  
  --wireless-gateway-task-definition-id "aa000102-0304-b0cd-ef56-a1b23cde456a"
```

输出：

```
{  
  "WirelessGatewayTaskDefinitionId": "aa204003-0604-30fb-ac82-a4f95aaf450a",  
  "Status": "Success"  
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》中的将设备和网关连接到适用于 LoRa WAN 的AWS IoT Core。](#)

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateWirelessGatewayTask](#)中的。

## create-wireless-gateway

以下代码示例演示了如何使用 create-wireless-gateway。

### AWS CLI

#### 创建无线网关

以下create-wireless-gateway示例创建了无线 LoRa WAN 设备网关。

```
aws iotwireless create-wireless-gateway \  
  --lorawan GatewayEui="a1b2c3d4567890ab",RfRegion="US915" \  
  --name "myFirstLoRaWANGateway" \  
  --description "Using my first LoRaWAN gateway"
```

#### 输出：

```
{  
  "Arn": "arn:aws:iotwireless:us-east-1:123456789012:WirelessGateway/12345678-  
a1b2-3c45-67d8-e90fa1b2c34d",  
  "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d"  
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateWirelessGateway](#) 中的。

## delete-destination

以下代码示例演示了如何使用 delete-destination。

### AWS CLI

#### 删除 IoT 无线目的地

以下delete-destination示例使用您创建的名称IoTWirelessDestination删除无线目标资源。

```
aws iotwireless delete-destination \  
  --name "IoTWirelessDestination"
```

此命令不生成任何输出。

有关更多信息，请参阅《物联网开发人员指南》中的[向 AWS IoT Core 为 AWS 物联网添加 LoRa WAN 添加目标](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteDestination](#)中的。

## delete-device-profile

以下代码示例演示了如何使用 delete-device-profile。

### AWS CLI

删除设备配置文件

以下 delete-device-profile 示例删除了您创建的具有指定 ID 的设备配置文件。

```
aws iotwireless delete-device-profile \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的[向适用于 LoRa WAN 的 AWS IoT Core 添加配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteDeviceProfile](#)中的。

## delete-service-profile

以下代码示例演示了如何使用 delete-service-profile。

### AWS CLI

删除服务配置文件

以下 delete-service-profile 示例删除了您创建的具有指定 ID 的服务配置文件。

```
aws iotwireless delete-service-profile \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
```

此命令不生成任何输出。



有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [向适用于 LoRa WAN 的 AWS IoT Core 添加配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteServiceProfile](#) 中的。

## delete-wireless-device

以下代码示例演示了如何使用 delete-wireless-device。

### AWS CLI

#### 删除无线设备

以下 delete-wireless-device 示例删除具有指定 ID 的无线设备。

```
aws iotwireless delete-wireless-device \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteWirelessDevice](#) 中的。

## delete-wireless-gateway-task-definition

以下代码示例演示了如何使用 delete-wireless-gateway-task-definition。

### AWS CLI

#### 删除无线网关任务定义

以下 delete-wireless-gateway-task-definition 示例删除您使用以下 ID 创建的无线网关任务定义。

```
aws iotwireless delete-wireless-gateway-task-definition \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteWirelessGatewayTaskDefinition](#)中的。

## delete-wireless-gateway-task

以下代码示例演示了如何使用 delete-wireless-gateway-task。

### AWS CLI

#### 删除无线网关任务

以下delete-wireless-gateway-task示例删除具有指定 ID 的无线网关任务。

```
aws iotwireless delete-wireless-gateway-task \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的[将设备和网关连接到适用于 LoRa WAN 的AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteWirelessGatewayTask](#)中的。

## delete-wireless-gateway

以下代码示例演示了如何使用 delete-wireless-gateway。

### AWS CLI

#### 删除无线网关

以下delete-wireless-gateway示例删除具有指定 ID 的无线网关。

```
aws iotwireless delete-wireless-gateway \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的[将设备和网关连接到适用于 LoRa WAN 的AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteWirelessGateway](#)中的。

## disassociate-aws-account-from-partner-account

以下代码示例演示了如何使用 `disassociate-aws-account-from-partner-account`。

### AWS CLI

取消合作伙伴账户与账户的 AWS 关联

以下 `disassociate-aws-account-from-partner-account` 示例取消合作伙伴账户与您当前关联 AWS 账户的关联。

```
aws iotwireless disassociate-aws-account-from-partner-account \  
  --partner-account-id "12345678901234" \  
  --partner-type "Sidewalk"
```

此命令不生成任何输出。

有关更多信息，请参阅《[物联网开发人员指南](#)》中的[将您的网关和无线设备添加到 AWS IoT Core for LoRa WAN](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateAwsAccountFromPartnerAccount](#) 中的。

## disassociate-wireless-device-from-thing

以下代码示例演示了如何使用 `disassociate-wireless-device-from-thing`。

### AWS CLI

断开设备与无线设备的关联

以下 `disassociate-wireless-device-from-thing` 示例将无线设备与其当前关联的事物断开关联。

```
aws iotwireless disassociate-wireless-device-from-thing \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
```

此命令不生成任何输出。

有关更多信息，请参阅《[物联网开发人员指南](#)》中的[将您的网关和无线设备添加到 AWS IoT Core for LoRa WAN](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateWirelessDeviceFromThing](#) 中的。

## disassociate-wireless-gateway-from-certificate

以下代码示例演示了如何使用 `disassociate-wireless-gateway-from-certificate`。

### AWS CLI

取消证书与无线网关的关联

以下步骤将无线网关与其当前关联的证书 `disassociate-wireless-gateway-from-certificate` 断开关联。

```
aws iotwireless disassociate-wireless-gateway-from-certificate \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
```

此命令不生成任何输出。

有关更多信息，请参阅《[物联网开发人员指南](#)》中的[将您的网关和无线设备添加到 AWS IoT Core for LoRa WAN](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateWirelessGatewayFromCertificate](#) 中的。

## disassociate-wireless-gateway-from-thing

以下代码示例演示了如何使用 `disassociate-wireless-gateway-from-thing`。

### AWS CLI

断开设备与无线网关的关联

以下 `disassociate-wireless-gateway-from-thing` 示例将无线网关与其当前关联的事物断开关联。

```
aws iotwireless disassociate-wireless-gateway-from-thing \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
```

此命令不生成任何输出。

有关更多信息，请参阅《[物联网开发人员指南](#)》中的[将您的网关和无线设备添加到 AWS IoT Core for LoRa WAN](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateWirelessGatewayFromThing](#)中的。

## get-destination

以下代码示例演示了如何使用 get-destination。

### AWS CLI

获取有关物联网无线目的地的信息

以下get-destination示例使用您创建的名称IoTWirelessDestination获取有关目标资源的信息。

```
aws iotwireless get-destination \  
  --name "IoTWirelessDestination"
```

输出：

```
{  
  "Arn": "arn:aws:iotwireless:us-east-1:123456789012:Destination/  
IoTWirelessDestination",  
  "Name": "IoTWirelessDestination",  
  "Expression": "IoTWirelessRule",  
  "ExpressionType": "RuleName",  
  "RoleArn": "arn:aws:iam::123456789012:role/IoTWirelessDestinationRole"  
}
```

有关更多信息，请参阅《物联网开发人员指南》中的向 AWS IoT Core 为 AWS 物联网添加 [LoRa WAN 目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDestination](#)中的。

## get-device-profile

以下代码示例演示了如何使用 get-device-profile。

### AWS CLI

获取有关设备配置文件的信息

以下`get-device-profile`示例使用您创建的指定 ID 获取有关设备配置文件的信息。

```
aws iotwireless get-device-profile \  
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
```

输出：

```
{  
  "Arn": "arn:aws:iotwireless:us-east-1:123456789012:DeviceProfile/12345678-  
a1b2-3c45-67d8-e90fa1b2c34d",  
  "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d",  
  "LoRaWAN": {  
    "MacVersion": "1.0.3",  
    "MaxDutyCycle": 10,  
    "Supports32BitFCnt": false,  
    "RegParamsRevision": "RP002-1.0.1",  
    "SupportsJoin": true,  
    "RfRegion": "US915",  
    "MaxEirp": 13,  
    "SupportsClassB": false,  
    "SupportsClassC": false  
  }  
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [向适用于 LoRa WAN 的 AWS IoT Core 添加配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDeviceProfile](#) 中的。

## get-partner-account

以下代码示例演示了如何使用 `get-partner-account`。

### AWS CLI

获取合作伙伴账户信息

以下`get-partner-account`示例获取有关具有以下 ID 的 Sidewalk 账户的信息。

```
aws iotwireless get-partner-account \  
  --partner-account-id "12345678901234" \  
  --partner-type "Sidewalk"
```

输出：

```
{
  "Sidewalk": {
    "AmazonId": "12345678901234",
    "Fingerprint":
"a123b45c6d78e9f012a34cd5e6a7890b12c3d45e6f78a1b234c56d7e890a1234"
  },
  "AccountLinked": false
}
```

有关更多信息，请参阅《物联网开发人员指南》中的 Amazon Sidewalk | AWS IoT Core [集成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetPartnerAccount](#) 中的。

## get-service-endpoint

以下代码示例演示了如何使用 get-service-endpoint。

AWS CLI

获取服务终端节点

以下 get-service-endpoint 示例获取 CUPS 协议的账户特定端点。

```
aws iotwireless get-service-endpoint
```

输出：

```
{
  "ServiceType": "CUPS",
  "ServiceEndpoint": "https://A1RMKZ37ACAGOT.cups.lorawan.us-east-1.amazonaws.com:443",
  "ServerTrust": "-----BEGIN CERTIFICATE-----\n
MIIESTCCAzGgAwIBAgITBn+UV4WH6Kx33rJTMlu8mYtWDTANBgkqhkiG9w0BAQsF\n
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hcm9uMRkwFwYDVQQDEwBBbWF6\n
b24gUm9vdCBDQSAxMB4XDTE1MTAyMjAwMDAwMFoXDTE1MTAxOTAwMDAwMFowRjEL\n
MAKGA1UEBhMCVVMxMjAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAw\n
IDFCMQ8wDQYDVQQDEwZBbWF6b24wgwEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK\n
AoIBAQCThZn3c68asg3Wuw6MLAd5tES6BIOsMzoKcG5b1PVo+sDORrMd4f2AbnZ\n
cMzPa43j4wNxhplty6aUKk4T1qe9B0wKFjwK6zmxXLVYo7bHVixsPlJ6q0MpFge5\n
-----"
```

```

b1DP+18x+B26A0piiQ0uPkfyDyeR4xQghfj66Yo19V+emU3nazfvpFA+R0z6WoVm\n
B5x+F2pV8xeKNR7u6azDdU5YVX1Tawp1mxRC1+WsAYmz6qP+z8ArDITC2FMVy2fw\n
0IjK0tEXc/VfmtTFch5+AfGYMGmqqvJ6LcXiAhqG5TI+Dr0RtM88k+8XUBCeQ8IG\n
KuANaL7TiItKZYxK1MMuTJtV9Ib1AgMBAAGjggE7MIIBNzASBgNVHRMBAf8ECDAG\n
AQH/AgEAMA4GA1UdDwEB/wQEAwIBhjAdBgNVHQ4EFgQUWaRmBlKge5WSPK0UByew\n
dFv5PdAwHwYDVR0jBBgwFoAUhBjMhTTsvAyU1C4IWZzHshB0CggwewYIKwYBBQUH\n
AQEEbzBtMC8GCCsGAQUFBzABhiNodHRwOi8vb2Nzc5yb290Y2ExLmFtYXpvbnRy\n
dXN0LmNvbTA6BgggrBgEFBQcwAoYuaHR0cDovL2NydC5yb290Y2ExLmFtYXpvbnRy\n
dXN0LmNvbS9yb290Y2ExLmN1c2A/BgNVHR8EODA2MDSgMqAwhi5odHRwOi8vY3Js\n
LnJvb3RjYTEuYW1hem9udHJ1c3QuY29tL3Jvb3RjYTEuY3JsMBMGA1UdIAQMAow\n
CAYGZ4EMAQIBMA0GCSqGSIb3DQEBCwUAA4IBAQCfkr41u3nPo4FCH0TjY3NT0V11\n
59Gt/a6ZiqyJEi+752+a1U5y6iAwYfmXss2lJwJFqMp2PphKg5625kXg8kP2CN5t\n
6G7bMQcT8C8xDZntYTd7WPD8UZiRKAJPBXa30/AbwuZe0GaFEQ8ugcYQgSn+IGBI\n
8/LwhBNTZTUVEWuCUUBVV18YtbAiPq3yXqMB480z+ctBWuZSkbvkNodPLamkB2g1\n
upRyzQ7qDn1X8nn8N8V7YJ6y68AtkHcNSRAnpTitxBKjtKPISLMVCx7i4hncxHZS\n
yLyKQXhw2W2Xs0qLeC1etA+jTGDK4UfLeC0SF7FSi8o5LL21L8IzApar2pR/\n
-----END CERTIFICATE-----\n"
}

```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetServiceEndpoint](#) 中的。

## get-service-profile

以下代码示例演示了如何使用 `get-service-profile`。

### AWS CLI

获取有关服务配置文件的信息

以下 `get-service-profile` 示例获取有关您创建的具有指定 ID 的服务配置文件的信息。

```

aws iotwireless get-service-profile \
  --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d"

```

输出：

```

{
  "Arn": "arn:aws:iotwireless:us-east-1:651419225604:ServiceProfile/538185bb-
d7e7-4b95-96a0-c51aa4a5b9a0",

```



```
"Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d",
"LoRaWAN": {
  "HrAllowed": false,
  "NwkGeoLoc": false,
  "DrMax": 15,
  "UlBucketSize": 4096,
  "PrAllowed": false,
  "ReportDevStatusBattery": false,
  "DrMin": 0,
  "DlRate": 60,
  "AddGwMetadata": false,
  "ReportDevStatusMargin": false,
  "MinGwDiversity": 1,
  "RaAllowed": false,
  "DlBucketSize": 4096,
  "DevStatusReqFreq": 24,
  "TargetPer": 5,
  "UlRate": 60
}
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [向适用于 LoRa WAN 的 AWS IoT Core 添加配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetServiceProfile](#) 中的。

## get-wireless-device-statistics

以下代码示例演示了如何使用 `get-wireless-device-statistics`。

### AWS CLI

获取有关无线设备的操作信息

以下 `get-wireless-device-statistics` 示例获取有关无线设备的操作信息。

```
aws iotwireless get-wireless-device-statistics \
  --wireless-device-id "1ffd32c8-8130-4194-96df-622f072a315f"
```

输出：

```
{
  "WirelessDeviceId": "1ffd32c8-8130-4194-96df-622f072a315f"
```

```
}
```

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的[将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetWirelessDeviceStatistics](#) 中的。

## get-wireless-device

以下代码示例演示了如何使用 get-wireless-device。

### AWS CLI

获取有关无线设备的信息

以下 get-wireless-device 示例列出了您 AWS 账户中可用的微件。

```
aws iotwireless get-wireless-device \  
  --identifier "1fffd32c8-8130-4194-96df-622f072a315f" \  
  --identifier-type WirelessDeviceID
```

输出：

```
{  
  "Name": "myLoRaWANDevice",  
  "ThingArn": "arn:aws:iot:us-east-1:123456789012:thing/44b87eb4-9bce-423d-  
b5fc-973f5ecc358b",  
  "DestinationName": "IoTWirelessDestination",  
  "Id": "1fffd32c8-8130-4194-96df-622f072a315f",  
  "ThingName": "44b87eb4-9bce-423d-b5fc-973f5ecc358b",  
  "Type": "LoRaWAN",  
  "LoRaWAN": {  
    "DeviceProfileId": "ab0c23d3-b001-45ef-6a01-2bc3de4f5333",  
    "ServiceProfileId": "fe98dc76-cd12-001e-2d34-5550432da100",  
    "OtaaV1_1": {  
      "AppKey": "3f4ca100e2fc675ea123f4eb12c4a012",  
      "JoinEui": "b4c231a359bc2e3d",  
      "NwkKey": "01c3f004a2d6efffe32c4eda14bcd2b4"  
    },  
    "DevEui": "ac12efc654d23fc2"  
  },  
  "Arn": "arn:aws:iotwireless:us-  
east-1:123456789012:WirelessDevice/1fffd32c8-8130-4194-96df-622f072a315f",
```

```
"Description": "My LoRaWAN wireless device"
}
```

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的[将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetWirelessDevice](#) 中的。

## get-wireless-gateway-certificate

以下代码示例演示了如何使用 `get-wireless-gateway-certificate`。

### AWS CLI

获取与无线网关关联的证书的 ID

以下 `get-wireless-gateway-certificate` 示例获取与具有指定 ID 的无线网关关联的证书 ID。

```
aws iotwireless get-wireless-gateway-certificate \
  --id "6c44ab31-8b4d-407a-bed3-19b6c7cda551"
```

输出：

```
{
  "IotCertificateId":
  "8ea4aeae3db34c78cce75d9abd830356869ead6972997e0603e5fd032c804b6f"
}
```

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的[将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetWirelessGatewayCertificate](#) 中的。

## get-wireless-gateway-firmware-information

以下代码示例演示了如何使用 `get-wireless-gateway-firmware-information`。

### AWS CLI

获取有关无线网关的固件信息

以下`get-wireless-gateway-firmware-information`示例获取有关无线网关的固件版本和其他信息。

```
aws iotwireless get-wireless-gateway-firmware-information \
  --id "3039b406-5cc9-4307-925b-9948c63da25b"
```

输出：

```
{
  "LoRaWAN" :{
    "CurrentVersion" :{
      "PackageVersion" : "1.0.0",
      "Station" : "2.0.5",
      "Model" : "linux"
    }
  }
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetWirelessGatewayFirmwareInformation](#) 中的。

## `get-wireless-gateway-statistics`

以下代码示例演示了如何使用 `get-wireless-gateway-statistics`。

AWS CLI

获取有关无线网关的操作信息

以下`get-wireless-gateway-statistics`示例获取有关无线网关的操作信息。

```
aws iotwireless get-wireless-gateway-statistics \
  --wireless-gateway-id "3039b406-5cc9-4307-925b-9948c63da25b"
```

输出：

```
{
  "WirelessGatewayId": "3039b406-5cc9-4307-925b-9948c63da25b"
```

```
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetWirelessGatewayStatistics](#) 中的。

## get-wireless-gateway-task-definition

以下代码示例演示了如何使用 `get-wireless-gateway-task-definition`。

### AWS CLI

获取有关无线网关任务定义的信息

以下 `get-wireless-gateway-task-definition` 示例获取有关具有指定 ID 的无线任务定义的信息。

```
aws iotwireless get-wireless-gateway-task-definition \
  --id "b7d3baad-25c7-35e7-a4e1-1683a0d61da9"
```

输出：

```
{
  "AutoCreateTasks": true,
  "Name": "TestAutoUpdate",
  "Update": {
    "UpdateDataSource" : "s3://cupsalphagafirmwarebin/station",
    "UpdateDataRole" : "arn:aws:iam::001234567890:role/SDK_Test_Role",
    "LoRaWAN" : {
      "CurrentVersion" : {
        "PackageVersion" : "1.0.0",
        "Station" : "2.0.5",
        "Model" : "linux"
      },
      "UpdateVersion" : {
        "PackageVersion" : "1.0.1",
        "Station" : "2.0.5",
        "Model" : "minihub"
      }
    }
  }
}
```

```
}
```

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的[将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetWirelessGatewayTaskDefinition](#) 中的。

## get-wireless-gateway-task

以下代码示例演示了如何使用 `get-wireless-gateway-task`。

### AWS CLI

获取有关无线网关任务的信息

以下 `get-wireless-gateway-task` 示例使用指定 ID 获取有关无线网关任务的信息。

```
aws iotwireless get-wireless-gateway-task \
  --id "11693a46-6866-47c3-a031-c9a616e7644b"
```

输出：

```
{
  "WirelessGatewayId": "6c44ab31-8b4d-407a-bed3-19b6c7cda551",
  "WirelessGatewayTaskDefinitionId": "b7d3baad-25c7-35e7-a4e1-1683a0d61da9",
  "Status": "Success"
}
```

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的[将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetWirelessGatewayTask](#) 中的。

## get-wireless-gateway

以下代码示例演示了如何使用 `get-wireless-gateway`。

### AWS CLI

获取有关无线网关的信息

以下 `get-wireless-gateway` 示例获取有关无线网关的信息 `myFirstLoRaWANGateway`。

```
aws iotwireless get-wireless-gateway \  
  --identifier "12345678-a1b2-3c45-67d8-e90fa1b2c34d" \  
  --identifier-type WirelessGatewayId
```

输出：

```
{  
  "Description": "My first LoRaWAN gateway",  
  "ThingArn": "arn:aws:iot:us-east-1:123456789012:thing/a1b2c3d4-5678-90ab-  
cdef-12ab345c67de",  
  "LoRaWAN": {  
    "RfRegion": "US915",  
    "GatewayEui": "a1b2c3d4567890ab"  
  },  
  "ThingName": "a1b2c3d4-5678-90ab-cdef-12ab345c67de",  
  "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d",  
  "Arn": "arn:aws:iotwireless:us-  
east-1:123456789012:WirelessGateway/6c44ab31-8b4d-407a-bed3-19b6c7cda551",  
  "Name": "myFirstLoRaWANGateway"  
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetWirelessGateway](#) 中的。

## list-destinations

以下代码示例演示了如何使用 list-destinations。

### AWS CLI

列出无线目的地

以下 list-destinations 示例列出了在您的 AWS 账户中注册的可用目的地。

```
aws iotwireless list-destinations
```

输出：

```
{
```

```
"DestinationList": [  
  {  
    "Arn": "arn:aws:iotwireless:us-east-1:123456789012:Destination/  
IoTWirelessDestination",  
    "Name": "IoTWirelessDestination",  
    "Expression": "IoTWirelessRule",  
    "Description": "Destination for messages processed using  
IoTWirelessRule",  
    "RoleArn": "arn:aws:iam::123456789012:role/IoTWirelessDestinationRole"  
  },  
  {  
    "Arn": "arn:aws:iotwireless:us-east-1:123456789012:Destination/  
IoTWirelessDestination2",  
    "Name": "IoTWirelessDestination2",  
    "Expression": "IoTWirelessRule2",  
    "RoleArn": "arn:aws:iam::123456789012:role/IoTWirelessDestinationRole"  
  }  
]  
}
```

有关更多信息，请参阅《物联网开发人员指南》中的向 AWS IoT Core 为 AWS 物联网添加 LoRa WAN 添加目标。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDestinations](#) 中的。

## list-device-profiles

以下代码示例演示了如何使用 list-device-profiles。

### AWS CLI

列出设备配置文件

以下 list-device-profiles 示例列出了注册到您的 AWS 账户的可用设备配置文件。

```
aws iotwireless list-device-profiles
```

输出：

```
{  
  "DeviceProfileList": [  
    {
```



```

        "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d",
        "Arn": "arn:aws:iotwireless:us-
east-1:123456789012:DeviceProfile/12345678-a1b2-3c45-67d8-e90fa1b2c34d"
    },
    {
        "Id": "a1b2c3d4-5678-90ab-cdef-12ab345c67de",
        "Arn": "arn:aws:iotwireless:us-east-1:123456789012:DeviceProfile/
a1b2c3d4-5678-90ab-cdef-12ab345c67de"
    }
]
}

```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [向适用于 LoRa WAN 的 AWS IoT Core 添加配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDeviceProfiles](#) 中的。

## list-partner-accounts

以下代码示例演示了如何使用 `list-partner-accounts`。

### AWS CLI

列出合作伙伴账户

以下 `list-partner-accounts` 示例列出了与您的账户关联的可用合作伙伴 AWS 账户。

```
aws iotwireless list-partner-accounts
```

输出：

```

{
  "Sidewalk": [
    {
      "AmazonId": "78965678771228",
      "Fingerprint":
"bd96d8ef66dbfd2160eb60e156849e82ad7018b8b73c1ba0b4fc65c32498ee35"
    },
    {
      "AmazonId": "89656787651228",
      "Fingerprint":
"bc5e99e151c07be14be7e6603e4489c53f858b271213a36ebe3370777ba06e9b"
    }
  ]
}

```

```
    }  
  ]  
}
```

有关更多信息，请参阅《物联网开发人员指南》中的 Amazon Sidewalk I AWS o AWS T [Core 集成](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListPartnerAccounts](#) 中的。

## list-service-profiles

以下代码示例演示了如何使用 list-service-profiles。

### AWS CLI

列出服务配置文件

以下 list-service-profiles 示例列出了注册到您的 AWS 账户的可用服务配置文件。

```
aws iotwireless list-service-profiles
```

输出：

```
{  
  "ServiceProfileList": [  
    {  
      "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d",  
      "Arn": "arn:aws:iotwireless:us-east-1:123456789012:ServiceProfile/538185bb-d7e7-4b95-96a0-c51aa4a5b9a0"  
    },  
    {  
      "Id": "a1b2c3d4-5678-90ab-cdef-12ab345c67de",  
      "Arn": "arn:aws:iotwireless:us-east-1:123456789012:ServiceProfile/ea8bc823-5d13-472e-8d26-9550737d8100"  
    }  
  ]  
}
```

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的向适用于 LoRa WAN 的 AWS IoT Core 添加配置文件。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListServiceProfiles](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出分配给资源的标签

以下 `list-tags-for-resource` 示例列出了分配给无线目标资源的标签。

```
aws iotwireless list-tags-for-resource \
  --resource-arn "arn:aws:iotwireless:us-east-1:123456789012:Destination/
  IoTWirelessDestination"
```

输出：

```
{
  "Tags": [
    {
      "Value": "MyValue",
      "Key": "MyTag"
    }
  ]
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的“[描述适用于 LoRa WAN 的 AWS IoT Core 资源](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## list-wireless-devices

以下代码示例演示了如何使用 `list-wireless-devices`。

### AWS CLI

列出可用的无线设备

以下 `list-wireless-devices` 示例列出了注册到您的 AWS 账户的可用无线设备。

```
aws iotwireless list-wireless-devices
```

输出：

```
{
  "WirelessDeviceList": [
    {
      "Name": "myLoRaWANDevice",
      "DestinationName": "IoTWirelessDestination",
      "Id": "1fffd32c8-8130-4194-96df-622f072a315f",
      "Type": "LoRaWAN",
      "LoRaWAN": {
        "DevEui": "ac12efc654d23fc2"
      },
      "Arn": "arn:aws:iotwireless:us-east-1:123456789012:WirelessDevice/1fffd32c8-8130-4194-96df-622f072a315f"
    }
  ]
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListWirelessDevices](#) 中的。

## list-wireless-gateway-task-definitions

以下代码示例演示了如何使用 `list-wireless-gateway-task-definitions`。

### AWS CLI

列出无线网关任务定义

以下 `list-wireless-gateway-task-definitions` 示例列出了注册到您的 AWS 账户的可用无线网关任务定义。

```
aws iotwireless list-wireless-gateway-task-definitions
```

输出：

```
{
  "TaskDefinitions": [
    {
      "Id": "b7d3baad-25c7-35e7-a4e1-1683a0d61da9",
      "LoRaWAN" :
        {
```

```
        "CurrentVersion" : {
            "PackageVersion" : "1.0.0",
            "Station" : "2.0.5",
            "Model" : "linux"
        },
        "UpdateVersion" : {
            "PackageVersion" : "1.0.1",
            "Station" : "2.0.5",
            "Model" : "minihub"
        }
    }
}
]
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListWirelessGatewayTaskDefinitions](#) 中的。

## list-wireless-gateways

以下代码示例演示了如何使用 list-wireless-gateways。

### AWS CLI

列出无线网关

以下 list-wireless-gateways 示例列出了您 AWS 账户中可用的无线网关。

```
aws iotwireless list-wireless-gateways
```

输出：

```
{
  "WirelessGatewayList": [
    {
      "Description": "My first LoRaWAN gateway",
      "LoRaWAN": {
        "RfRegion": "US915",
        "GatewayEui": "dac632ebc01d23e4"
      },
      "Id": "3039b406-5cc9-4307-925b-9948c63da25b",
    }
  ]
}
```

```

    "Arn": "arn:aws:iotwireless:us-
east-1:123456789012:WirelessGateway/3039b406-5cc9-4307-925b-9948c63da25b",
    "Name": "myFirstLoRaWANGateway"
  },
  {
    "Description": "My second LoRaWAN gateway",
    "LoRaWAN": {
      "RfRegion": "US915",
      "GatewayEui": "cda123ffffe92ecd2"
    },
    "Id": "3285bdc7-5a12-4991-84ed-dadca65e342e",
    "Arn": "arn:aws:iotwireless:us-
east-1:123456789012:WirelessGateway/3285bdc7-5a12-4991-84ed-dadca65e342e",
    "Name": "mySecondLoRaWANGateway"
  }
]
}

```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListWirelessGateways](#) 中的。

## send-data-to-wireless-device

以下代码示例演示了如何使用 `send-data-to-wireless-device`。

### AWS CLI

#### 向无线设备发送数据

以下 `send-data-to-wireless-device` 示例将解密后的应用程序数据帧发送到无线设备。

```

aws iotwireless send-data-to-wireless-device \
  --id "11aa5eae-2f56-4b8e-a023-b28d98494e49" \
  --transmit-mode "1" \
  --payload-data "SGVsbG8gVG8gRGV2c2lt" \
  --wireless-metadata LoRaWAN={FPort=1}

```

输出：

```

{
  MessageId: "6011dd36-0043d6eb-0072-0008"
}

```

```
}
```

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的[将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SendDataToWirelessDevice](#) 中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源指定标签键和值

以下 tag-resource 示例 IoTWirelessDestination 使用密钥 MyTag 和值标记无线目的地 MyValue。

```
aws iotwireless tag-resource \  
  --resource-arn "arn:aws:iotwireless:us-east-1:651419225604:Destination/  
IoTWirelessDestination" \  
  --tags Key="MyTag",Value="MyValue"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的“[描述适用于 LoRa WAN 的 AWS IoT Core 资源](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## test-wireless-device

以下代码示例演示了如何使用 test-wireless-device。

### AWS CLI

测试无线设备

以下 test-wireless-device 示例将的上行链路数据发送 Hello 到具有指定 ID 的设备。

```
aws iotwireless test-wireless-device \  
  --id "11aa5eae-2f56-4b8e-a023-b28d98494e49"
```

输出：

```
{
  Result: "Test succeeded. one message is sent with payload: hello"
}
```

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [将设备和网关连接到适用于 LoRa WAN 的 AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TestWirelessDevice](#) 中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

AWS CLI

从资源中移除一个或多个标签

以下 untag-resource 示例将标签 MyTag 及其值从无线目标中删除 IoTWirelessDestination。

```
aws iotwireless untag-resource \
  --resource-arn "arn:aws:iotwireless:us-east-1:123456789012:Destination/
IoTWirelessDestination" \
  --tag-keys "MyTag"
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS 物联网开发人员指南》](#) 中的 [“描述适用于 LoRa WAN 的 AWS IoT Core 资源”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-destination

以下代码示例演示了如何使用 update-destination。

AWS CLI

更新目的地的属性



以下update-destination示例更新了无线目标的描述属性。

```
aws iotwireless update-destination \  
  --name "IoTWirelessDestination" \  
  --description "Destination for messages processed using IoTWirelessRule"
```

此命令不生成任何输出。

有关更多信息，请参阅《物联网开发人员指南》中的向 AWS IoT Core 为 AWS 物联网添加 [LoRa WAN 目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDestination](#)中的。

## update-partner-account

以下代码示例演示了如何使用 update-partner-account。

### AWS CLI

更新合作伙伴账户的属性

以下内容update-partner-account更新AppServerPrivateKey了具有指定 ID 的账户的。

```
aws iotwireless update-partner-account \  
  --partner-account-id "78965678771228" \  
  --partner-type "Sidewalk" \  
  --sidewalk  
  AppServerPrivateKey="f798ab4899346a88599180fee9e14fa1ada7b6df989425b7c6d2146dd6c815bb"
```

此命令不生成任何输出。

有关更多信息，请参阅《物联网开发人员指南》中的 Amazon Sidewalk 与 AWS IoT Core [集成](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePartnerAccount](#)中的。

## update-wireless-device

以下代码示例演示了如何使用 update-wireless-device。

### AWS CLI

更新无线设备的属性

以下update-wireless-device示例更新了注册到您的 AWS 帐户的无线设备的属性。

```
aws iotwireless update-wireless-device \  
  --id "1ffd32c8-8130-4194-96df-622f072a315f" \  
  --destination-name IoTWirelessDestination2 \  
  --description "Using my first LoRaWAN device"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的[将设备和网关连接到适用于 LoRa WAN 的AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateWirelessDevice](#)中的。

## update-wireless-gateway

以下代码示例演示了如何使用 update-wireless-gateway。

### AWS CLI

#### 更新无线网关

以下update-wireless-gateway示例更新了您的无线网关的描述。

```
aws iotwireless update-wireless-gateway \  
  --id "3285bdc7-5a12-4991-84ed-dadca65e342e" \  
  --description "Using my LoRaWAN gateway"
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS 物联网开发人员指南](#)》中的[将设备和网关连接到适用于 LoRa WAN 的AWS IoT Core](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateWirelessGateway](#)中的。

## 使用 Amazon IVS 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon IVS 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## batch-get-channel

以下代码示例演示了如何使用 batch-get-channel。

AWS CLI

获取有关多个频道的频道配置信息

以下batch-get-channel示例列出了有关指定频道的信息。

```
aws ivs batch-get-channel \  
  --arns arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh \  
        arn:aws:ivs:us-west-2:123456789012:channel/efghEFGHijkl
```

输出：

```
{  
  "channels": [  
    {  
      "arn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",  
      "authorized": false,  
      "ingestEndpoint": "a1b2c3d4e5f6.global-contribute.live-video.net",  
      "insecureIngest": false,  
      "latencyMode": "LOW",  
      "name": "channel-1",  
      "playbackUrl": "https://a1b2c3d4e5f6.us-west-2.playback.live-video.net/  
api/video/v1/us-west-2.123456789012.channel-1.abcdEFGH.m3u8",  
      "preset": "",  
      "playbackRestrictionPolicyArn": "",  
      "recordingConfigurationArn": "arn:aws:ivs:us-  
west-2:123456789012:recording-configuration/ABCD12cdEFgh",
```

```

    "srt": {
      "endpoint": "a1b2c3d4e5f6.srt.live-video.net",
      "passphrase":
"AB1C2defGHijkLMNo3PqQRstUvwxyzaBCDEfghh4ijkLMN5opqrStuVWxyzAbCDEfghIJ"
    },
    "tags": {},
    "type": "STANDARD"
  },
  {
    "arn": "arn:aws:ivs:us-west-2:123456789012:channel/efghEFGHijkl",
    "authorized": false,
    "ingestEndpoint": "a1b2c3d4e5f6.global-contribute.live-video.net",
    "insecureIngest": true,
    "latencyMode": "LOW",
    "name": "channel-2",
    "playbackUrl": "https://a1b2c3d4e5f6.us-west-2.playback.live-video.net/
api/video/v1/us-west-2.123456789012.channel-2.abcdEFGH.m3u8",
    "preset": "",
    "playbackRestrictionPolicyArn": "arn:aws:ivs:us-
west-2:123456789012:playback-restriction-policy/ABcdef34ghIJ",
    "recordingConfigurationArn": "",
    "srt": {
      "endpoint": "a1b2c3d4e5f6.srt.live-video.net",
      "passphrase":
"BA1C2defGHijkLMNo3PqQRstUvwxyzaBCDEfghh4ijkLMN5opqrStuVWxyzAbCDEfghIJ"
    },
    "tags": {},
    "type": "STANDARD"
  }
]
}

```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetChannel](#)中的。

## batch-get-stream-key

以下代码示例演示了如何使用 batch-get-stream-key。

### AWS CLI

获取有关多个直播密钥的信息

以下 `batch-get-stream-key` 示例获取有关指定流密钥的信息。

```
aws ivs batch-get-stream-key \  
  --arns arn:aws:ivs:us-west-2:123456789012:stream-key/skSKABCDefgh \  
        arn:aws:ivs:us-west-2:123456789012:stream-key/skSKIJKLmnop
```

输出：

```
{  
  "streamKeys": [  
    {  
      "arn": "arn:aws:ivs:us-west-2:123456789012:stream-key/skSKABCDefgh",  
      "value": "sk_us-west-2_abcdABCDefgh_567890abcdef",  
      "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",  
      "tags": {}  
    },  
    {  
      "arn": "arn:aws:ivs:us-west-2:123456789012:stream-key/skSKIJKLmnop",  
      "value": "sk_us-west-2_abcdABCDefgh_567890ghijkl",  
      "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",  
      "tags": {}  
    }  
  ]  
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchGetStreamKey](#) 中的。

## **batch-start-viewer-session-revocation**

以下代码示例演示了如何使用 `batch-start-viewer-session-revocation`。

### AWS CLI

撤消多个频道 ARN 和 Viewer-ID 对的观看者会话

以下 `batch-start-viewer-session-revocation` 示例同时对多个 Channel-ARN 和 Viewer-ID 对执行会话撤销。请求可能正常完成，但如果调用者无权撤销指定会话，则会在错误字段中返回。

```
aws ivs batch-start-viewer-session-revocation \  
  --arns arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh \  
        arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh
```

```
--viewer-sessions '[{"channelArn":"arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh1","viewerId":"abcdefg1","viewerSessionVersionsLessThanOrEqualTo":1234567890},
\
  {"channelArn":"arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh2","viewerId":"abcdefg2","viewerSessionVersionsLessThanOrEqualTo":1234567890}]'
```

输出：

```
{
  "errors": [
    {
      "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh1",
      "viewerId": "abcdefg1",
      "code": "403",
      "message": "not authorized",
    },
    {
      "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh2",
      "viewerId": "abcdefg2",
      "code": "403",
      "message": "not authorized",
    }
  ]
}
```

有关更多信息，请参阅《Amazon Interactive Video Service 用户指南》中的[设置私有通道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchStartViewerSessionRevocation](#)中的。

## create-channel

以下代码示例演示了如何使用 create-channel。

### AWS CLI

示例 1：创建没有录音的频道

以下create-channel示例创建了一个新频道和关联的直播密钥以开始直播。

```
aws ivs create-channel \
  --name "test-channel" \
```

```
--no-insecure-ingest
```

输出：

```
{
  "channel": {
    "arn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
    "authorized": false,
    "name": "test-channel",
    "latencyMode": "LOW",
    "playbackRestrictionPolicyArn": "",
    "recordingConfigurationArn": "",
    "srt": {
      "endpoint": "a1b2c3d4e5f6.srt.live-video.net",
      "passphrase":
"AB1C2defGHijkLMNo3PqQRstUvwxyzABCDefghh4ijklMN5opqrStuVWxyzAbCDEfghIJ"
    },
    "ingestEndpoint": "a1b2c3d4e5f6.global-contribute.live-video.net",
    "insecureIngest": false,
    "playbackUrl": "https://a1b2c3d4e5f6.us-west-2.playback.live-video.net/api/
video/v1/us-west-2.123456789012.channel.abcdEFGH.m3u8",
    "preset": "",
    "tags": {},
    "type": "STANDARD"
  },
  "streamKey": {
    "arn": "arn:aws:ivs:us-west-2:123456789012:stream-key/g1H2I3j4k5L6",
    "value": "sk_us-west-2_abcdABCDefgh_567890abcdef",
    "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
    "tags": {}
  }
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

示例 2：使用其 ARN 指定的 RecordingConfiguration 资源创建启用了录制功能的频道

以下 create-channel 示例创建了一个新频道和关联的直播密钥以开始直播，并为该频道设置了录制。

```
aws ivs create-channel \
  --name test-channel-with-recording \
  --insecure-ingest \
```

```
--recording-configuration-arn "arn:aws:ivs:us-west-2:123456789012:recording-configuration/ABCD12cdEFgh"
```

输出：

```
{
  "channel": {
    "arn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
    "name": "test-channel-with-recording",
    "latencyMode": "LOW",
    "type": "STANDARD",
    "playbackRestrictionPolicyArn": "",
    "recordingConfigurationArn": "arn:aws:ivs:us-west-2:123456789012:recording-configuration/ABCD12cdEFgh",
    "srt": {
      "endpoint": "a1b2c3d4e5f6.srt.live-video.net",
      "passphrase":
"BA1C2defGHijkLMNo3PqQRstUvwxyzABCDefghh4ijk1MN5opqrStuVWxyzAbCDEfghIJ"
    },
    "ingestEndpoint": "a1b2c3d4e5f6.global-contribute.live-video.net",
    "insecureIngest": true,
    "playbackUrl": "https://a1b2c3d4e5f6.us-west-2.playback.live-video.net/api/video/v1/us-west-2.123456789012.channel.abcdEFGH.m3u8",
    "preset": "",
    "authorized": false,
    "tags": {},
    "type": "STANDARD"
  },
  "streamKey": {
    "arn": "arn:aws:ivs:us-west-2:123456789012:stream-key/abcdABCDefgh",
    "value": "sk_us-west-2_abcdABCDefgh_567890abcdef",
    "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
    "tags": {}
  }
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[录制到 Amazon S3](#)。

示例 3：使用其 ARN 指定的播放限制策略创建频道

以下 `create-channel` 示例创建了一个新频道和关联的直播密钥以开始直播，并为该频道设置了播放限制政策。



```
aws ivs create-channel \  
  --name test-channel-with-playback-restriction-policy \  
  --insecure-ingest \  
  --playback-restriction-policy-arn "arn:aws:ivs:us-west-2:123456789012:playback-  
restriction-policy/ABcdef34ghIJ"
```

输出：

```
{  
  "channel": {  
    "arn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",  
    "name": "test-channel-with-playback-restriction-policy",  
    "latencyMode": "LOW",  
    "type": "STANDARD",  
    "playbackRestrictionPolicyArn": "arn:aws:ivs:us-  
west-2:123456789012:playback-restriction-policy/ABcdef34ghIJ",  
    "recordingConfigurationArn": "",  
    "srt": {  
      "endpoint": "a1b2c3d4e5f6.srt.live-video.net",  
      "passphrase":  
"AB1C2edfGHijkLMNo3PqQRstUvwxyzaBCDEfghh4ijklMN5opqrStuVWxyzAbCDEfghIJ"  
    },  
    "ingestEndpoint": "a1b2c3d4e5f6.global-contribute.live-video.net",  
    "insecureIngest": true,  
    "playbackUrl": "https://a1b2c3d4e5f6.us-west-2.playback.live-video.net/api/  
video/v1/us-west-2.123456789012.channel.abcdEFGH.m3u8",  
    "preset": "",  
    "authorized": false,  
    "tags": {},  
    "type": "STANDARD"  
  },  
  "streamKey": {  
    "arn": "arn:aws:ivs:us-west-2:123456789012:stream-key/abcdABCDefgh",  
    "value": "sk_us-west-2_abcdABCDefgh_567890abcdef",  
    "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",  
    "tags": {}  
  }  
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[不需要的内容和查看器](#)。

• 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateChannel](#)中的。

## create-playback-restriction-policy

以下代码示例演示了如何使用 create-playback-restriction-policy。

### AWS CLI

#### 创建播放限制政策

以下create-playback-restriction-policy示例创建了新的播放限制策略。

```
aws ivs create-playback-restriction-policy \  
  --name "test-playback-restriction-policy" \  
  --enable-strict-origin-enforcement \  
  --tags "key1=value1, key2=value2" \  
  --allowed-countries US MX \  
  --allowed-origins https://www.website1.com https://www.website2.com
```

输出：

```
{  
  "playbackRestrictionPolicy": {  
    "arn": "arn:aws:ivs:us-west-2:123456789012:playback-restriction-policy/  
ABcdef34ghIJ",  
    "allowedCountries": [  
      "US",  
      "MX"  
    ],  
    "allowedOrigins": [  
      "https://www.website1.com",  
      "https://www.website2.com"  
    ],  
    "enableStrictOriginEnforcement": true,  
    "name": "test-playback-restriction-policy",  
    "tags": {  
      "key1": "value1",  
      "key2": "value2"  
    }  
  }  
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[不需要的内容和查看器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePlaybackRestrictionPolicy](#)中的。

## create-recording-configuration

以下代码示例演示了如何使用 create-recording-configuration。

### AWS CLI

#### 创建 RecordingConfiguration 资源

以下 create-recording-configuration 示例创建了一个 RecordingConfiguration 资源以启用录制到 Amazon S3。

```
aws ivs create-recording-configuration \  
  --name "test-recording-config" \  
  --recording-reconnect-window-seconds 60 \  
  --tags "key1=value1, key2=value2" \  
  --rendition-configuration renditionSelection="CUSTOM",renditions="HD" \  
  --thumbnail-configuration  
  recordingMode="INTERVAL",targetIntervalSeconds=1,storage="LATEST",resolution="LOWEST_RESOLUTION" \  
  --destination-configuration s3={bucketName=demo-recording-bucket}
```

输出：

```
{  
  "recordingConfiguration": {  
    "arn": "arn:aws:ivs:us-west-2:123456789012:recording-configuration/  
ABcdef34ghIJ",  
    "name": "test-recording-config",  
    "destinationConfiguration": {  
      "s3": {  
        "bucketName": "demo-recording-bucket"  
      }  
    },  
    "state": "CREATING",  
    "tags": {  
      "key1": "value1",  
      "key2": "value2"  
    },  
    "thumbnailConfiguration": {  
      "recordingMode": "INTERVAL",  
      "targetIntervalSeconds": 1,  
      "resolution": "LOWEST_RESOLUTION",  
      "storage": [  

```

```

        "LATEST"
      ]
    },
    "recordingReconnectWindowSeconds": 60,
    "renditionConfiguration": {
      "renditionSelection": "CUSTOM",
      "renditions": [
        "HD"
      ]
    }
  }
}

```

有关更多信息，请参阅《[亚马逊互动视频服务用户指南](#)》中的“[录制到 Amazon S3](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateRecordingConfiguration](#) 中的。

## create-stream-key

以下代码示例演示了如何使用 create-stream-key。

### AWS CLI

#### 创建直播密钥

以下 create-stream-key 示例为指定的 ARN ( Amazon 资源名称 ) 创建流密钥。

```
aws ivs create-stream-key \
  --channel-arn arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh
```

输出：

```
{
  "streamKey": {
    "arn": "arn:aws:ivs:us-west-2:123456789012:stream-key/abcdABCDefgh",
    "value": "sk_us-west-2_abcdABCDefgh_567890abcdef",
    "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
    "tags": {}
  }
}
```

有关更多信息，请参阅《[IVS 低延迟用户指南](#)》中的 [创建频道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateStreamKey](#)中的。

## delete-channel

以下代码示例演示了如何使用 delete-channel。

### AWS CLI

删除频道及其关联的直播密钥

以下delete-channel示例删除具有指定 ARN ( Amazon 资源名称 ) 的频道。

```
aws ivs delete-channel \  
  --arn arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh
```

此命令不生成任何输出。

有关更多信息，请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteChannel](#)中的。

## delete-playback-key-pair

以下代码示例演示了如何使用 delete-playback-key-pair。

### AWS CLI

删除指定的回放 key pair

以下delete-playback-key-pair示例返回指定 key pair 的指纹。

```
aws ivs delete-playback-key-pair \  
  --arn arn:aws:ivs:us-west-2:123456789012:playback-key/abcd1234efgh
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Interactive Video Service 用户指南》中的[设置私有通道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePlaybackKeyPair](#)中的。

## delete-playback-restriction-policy

以下代码示例演示了如何使用 delete-playback-restriction-policy。

## AWS CLI

### 删除播放限制政策

以下delete-playback-restriction-policy示例删除了具有指定策略 ARN ( Amazon 资源名称 ) 的播放限制政策。

```
aws ivs delete-playback-restriction-policy \  
    --arn "arn:aws:ivs:us-west-2:123456789012:playback-restriction-policy/  
ABCdef34ghIJ"
```

此命令不生成任何输出。

有关更多信息，请参阅《IVS 低延迟用户指南》中的[不需要的内容和查看器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePlaybackRestrictionPolicy](#)中的。

## delete-recording-configuration

以下代码示例演示了如何使用 delete-recording-configuration。

### AWS CLI

删除由其 ARN 指定的 RecordingConfiguration 资源

以下delete-recording-configuration示例删除具有指定 ARN 的 RecordingConfiguration 资源。

```
aws ivs delete-recording-configuration \  
    --arn "arn:aws:ivs:us-west-2:123456789012:recording-configuration/ABCdef34ghIJ"
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊互动视频服务用户指南](#)》中的“[录制到 Amazon S3](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRecordingConfiguration](#)中的。

## delete-stream-key

以下代码示例演示了如何使用 delete-stream-key。

## AWS CLI

### 删除直播密钥

以下delete-stream-key示例删除了指定 ARN 的直播密钥 ( Amazon 资源名称 ) , 因此该密钥不能再用于流式传输。

```
aws ivs delete-stream-key \  
  --arn arn:aws:ivs:us-west-2:123456789012:stream-key/g1H2I3j4k5L6
```

此命令不生成任何输出。

有关更多信息, 请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

- 有关 API 的详细信息, 请参阅AWS CLI 命令参考[DeleteStreamKey](#)中的。

## get-channel

以下代码示例演示了如何使用 get-channel。

## AWS CLI

### 获取频道的配置信息

以下get-channel示例获取指定频道 ARN 的频道配置 ( Amazon 资源名称 ) 。

```
aws ivs get-channel \  
  --arn arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh
```

输出 :

```
{  
  "channel": {  
    "arn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",  
    "name": "channel-1",  
    "latencyMode": "LOW",  
    "type": "STANDARD",  
    "playbackRestrictionPolicyArn": "",  
    "preset": "",  
    "recordingConfigurationArn": "arn:aws:ivs:us-west-2:123456789012:recording-  
configuration/ABCD12cdEFgh",
```

```
    "srt": {
      "endpoint": "a1b2c3d4e5f6.srt.live-video.net",
      "passphrase":
"AB1C2defGHijkLMNo3PqQRstUvwxyzAbCDEfghh4ijklMN5opqrStuVWxyzAbCDEfghIJ"
    },
    "ingestEndpoint": "a1b2c3d4e5f6.global-contribute.live-video.net",
    "insecureIngest": false,
    "playbackUrl": "https://a1b2c3d4e5f6.us-west-2.playback.live-video.net/api/
video/v1/us-west-2.123456789012.channel.abcdEFGH.m3u8",
    "tags": {}
  }
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetChannel](#) 中的。

## get-playback-key-pair

以下代码示例演示了如何使用 `get-playback-key-pair`。

### AWS CLI

获取指定的播放 key pair

以下 `get-playback-key-pair` 示例返回指定 key pair 的指纹。

```
aws ivs get-playback-key-pair \
  --arn arn:aws:ivs:us-west-2:123456789012:playback-key/abcd1234efgh
```

输出：

```
{
  "keyPair": {
    "arn": "arn:aws:ivs:us-west-2:123456789012:playback-key/abcd1234efgh",
    "name": "my-playback-key",
    "fingerprint": "0a:1b:2c:ab:cd:ef:34:56:70:b1:b2:71:01:2a:a3:72",
    "tags": {}
  }
}
```

有关更多信息，请参阅《Amazon Interactive Video Service 用户指南》中的[设置私有通道](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPlaybackKeyPair](#)中的。

## get-playback-restriction-policy

以下代码示例演示了如何使用 `get-playback-restriction-policy`。

### AWS CLI

获取播放限制策略的配置信息

以下`get-playback-restriction-policy`示例使用指定策略 ARN ( Amazon 资源名称 ) 获取播放限制策略配置。

```
aws ivs get-playback-restriction-policy \  
  --arn "arn:aws:ivs:us-west-2:123456789012:playback-restriction-policy/  
ABCdef34ghIJ"
```

输出：

```
{  
  "playbackRestrictionPolicy": {  
    "arn": "arn:aws:ivs:us-west-2:123456789012:playback-restriction-policy/  
ABCdef34ghIJ",  
    "allowedCountries": [  
      "US",  
      "MX"  
    ],  
    "allowedOrigins": [  
      "https://www.website1.com",  
      "https://www.website2.com"  
    ],  
    "enableStrictOriginEnforcement": true,  
    "name": "test-playback-restriction-policy",  
    "tags": {  
      "key1": "value1",  
      "key2": "value2"  
    }  
  }  
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[不需要的内容和查看器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPlaybackRestrictionPolicy](#)中的。

## get-recording-configuration

以下代码示例演示了如何使用 `get-recording-configuration`。

### AWS CLI

获取有关 `RecordingConfiguration` 资源的信息

以下 `get-recording-configuration` 示例获取有关指定 ARN 的 `RecordingConfiguration` 资源的信息。

```
aws ivs get-recording-configuration \  
  --arn "arn:aws:ivs:us-west-2:123456789012:recording-configuration/ABcdef34ghIJ"
```

输出：

```
{  
  "recordingConfiguration": {  
    "arn": "arn:aws:ivs:us-west-2:123456789012:recording-configuration/  
ABcdef34ghIJ",  
    "destinationConfiguration": {  
      "s3": {  
        "bucketName": "demo-recording-bucket"  
      }  
    },  
    "name": "test-recording-config",  
    "recordingReconnectWindowSeconds": 60,  
    "state": "ACTIVE",  
    "tags": {  
      "key1" : "value1",  
      "key2" : "value2"  
    },  
    "thumbnailConfiguration": {  
      "recordingMode": "INTERVAL",  
      "targetIntervalSeconds": 1,  
      "resolution": "LOWEST_RESOLUTION",  
      "storage": [  
        "LATEST"  
      ]  
    },  
    "renditionConfiguration": {  
      "renditionSelection": "CUSTOM",  
      "renditions": [  

```

```
        "HD"  
      ]  
    }  
  }  
}
```

有关更多信息，请参阅《[亚马逊互动视频服务用户指南](#)》中的“[录制到 Amazon S3](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRecordingConfiguration](#)中的。

## get-stream-key

以下代码示例演示了如何使用 get-stream-key。

### AWS CLI

获取有关直播的信息

以下get-stream-key示例获取有关指定流密钥的信息。

```
aws ivs get-stream-key \  
    --arn arn:aws:ivs:us-west-2:123456789012:stream-key/skSKABCDefgh --region=us-  
west-2
```

输出：

```
{  
  "streamKey": {  
    "arn": "arn:aws:ivs:us-west-2:123456789012:stream-key/skSKABCDefgh",  
    "value": "sk_us-west-2_abcdABCDefgh_567890abcdef",  
    "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",  
    "tags": {}  
  }  
}
```

有关更多信息，请参阅《[IVS 低延迟用户指南](#)》中的[创建频道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetStreamKey](#)中的。

## get-stream-session

以下代码示例演示了如何使用 get-stream-session。

## AWS CLI

### 获取指定直播的元数据

以下`get-stream-session`示例获取指定频道 ARN ( Amazon 资源名称 ) 和指定直播的元数据配置；如果未提供 `StreamID`，则会选择该频道的最新直播。

```
aws ivs get-stream-session \  
  --channel-arn arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh \  
  --stream-id "mystream"
```

输出：

```
{  
  "streamSession": {  
    "streamId": "mystream1",  
    "startTime": "2023-06-26T19:09:28+00:00",  
    "channel": {  
      "arn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",  
      "name": "mychannel",  
      "latencyMode": "LOW",  
      "type": "STANDARD",  
      "recordingConfigurationArn": "arn:aws:ivs:us-west-2:123456789012:recording-configuration/ABCdef34ghIJ",  
      "srt": {  
        "endpoint": "a1b2c3d4e5f6.srt.live-video.net",  
        "passphrase":  
"AB1C2defGHijklMNop3PqQRstUvwxyzABCDefghh4ijklMN5opqrStuVWxyzAbCDEfghIJ"  
      },  
      "ingestEndpoint": "a1b2c3d4e5f6.global-contribute.live-video.net",  
      "playbackUrl": "url-string",  
      "authorized": false,  
      "insecureIngest": false,  
      "preset": ""  
    },  
    "ingestConfiguration": {  
      "video": {  
        "avcProfile": "Baseline",  
        "avcLevel": "4.2",  
        "codec": "avc1.42C02A",  
        "encoder": "Lavf58.45.100",  
        "targetBitrate": 8789062,  
        "targetFramerate": 60,  
      }  
    }  
  }  
}
```

```
        "videoHeight": 1080,
        "videoWidth": 1920
    },
    "audio": {
        "codec": "mp4a.40.2",
        "targetBitrate": 46875,
        "sampleRate": 8000,
        "channels": 2
    }
},
"recordingConfiguration": {
    "arn": "arn:aws:ivs:us-west-2:123456789012:recording-configuration/
ABCdef34ghIJ",
    "name": "test-recording-config",
    "destinationConfiguration": {
        "s3": {
            "bucketName": "demo-recording-bucket"
        }
    },
    "state": "ACTIVE",
    "tags": {
        "key1": "value1",
        "key2": "value2"
    },
    "thumbnailConfiguration": {
        "recordingMode": "INTERVAL",
        "targetIntervalSeconds": 1,
        "resolution": "LOWEST_RESOLUTION",
        "storage": [
            "LATEST"
        ]
    },
    "recordingReconnectWindowSeconds": 60,
    "renditionConfiguration": {
        "renditionSelection": "CUSTOM",
        "renditions": [
            "HD"
        ]
    }
},
"truncatedEvents": [
    {
        "name": "Recording Start",
        "type": "IVS Recording State Change",
```

```

        "eventTime": "2023-06-26T19:09:35+00:00"
      },
      {
        "name": "Stream Start",
        "type": "IVS Stream State Change",
        "eventTime": "2023-06-26T19:09:34+00:00"
      },
      {
        "name": "Session Created",
        "type": "IVS Stream State Change",
        "eventTime": "2023-06-26T19:09:28+00:00"
      }
    ]
  }
}

```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetStreamSession](#)中的。

## get-stream

以下代码示例演示了如何使用 get-stream。

### AWS CLI

获取有关直播的信息

以下get-stream示例获取有关指定频道直播的信息。

```

aws ivs get-stream \
  --channel-arn arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh

```

输出：

```

{
  "stream": {
    "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
    "playbackUrl": "https://a1b2c3d4e5f6.us-west-2.playback.live-video.net/api/video/v1/us-west-2.123456789012.channel.abcdEFGH.m3u8",
    "startTime": "2020-05-05T21:55:38Z",
    "state": "LIVE",
    "health": "HEALTHY",
  }
}

```

```
    "streamId": "st-ABCDEFGHIJ01234KLMN5678",
    "viewerCount": 1
  }
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetStream](#)中的。

## import-playback-key-pair

以下代码示例演示了如何使用 `import-playback-key-pair`。

### AWS CLI

导入新 key pair 的公共部分

以下`import-playback-key-pair`示例导入指定的公钥（以 PEM 格式的字符串形式指定），并返回新密钥对的 arn 和指纹。

```
aws ivs import-playback-key-pair \
  --name "my-playback-key" \
  --public-key-material "G1lbnQx0TA3BgNVBAMMFdoeSBhcmUgeW91IGR1..."
```

输出：

```
{
  "keyPair": {
    "arn": "arn:aws:ivs:us-west-2:123456789012:playback-key/abcd1234efgh",
    "name": "my-playback-key",
    "fingerprint": "0a:1b:2c:ab:cd:ef:34:56:70:b1:b2:71:01:2a:a3:72",
    "tags": {}
  }
}
```

有关更多信息，请参阅《Amazon Interactive Video Service 用户指南》中的[设置私有通道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ImportPlaybackKeyPair](#)中的。

## list-channels

以下代码示例演示了如何使用 `list-channels`。

## AWS CLI

示例 1：获取有关所有频道的摘要信息

以下 `list-channels` 示例列出了您 AWS 账户的所有频道。

```
aws ivs list-channels
```

输出：

```
{
  "channels": [
    {
      "arn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
      "name": "channel-1",
      "latencyMode": "LOW",
      "authorized": false,
      "insecureIngest": false,
      "preset": "",
      "playbackRestrictionPolicyArn": "",
      "recordingConfigurationArn": "arn:aws:ivs:us-west-2:123456789012:recording-configuration/ABCD12cdEFgh",
      "tags": {},
      "type": "STANDARD"
    },
    {
      "arn": "arn:aws:ivs:us-west-2:123456789012:channel/efghEFGHijkl",
      "name": "channel-2",
      "latencyMode": "LOW",
      "authorized": false,
      "preset": "",
      "playbackRestrictionPolicyArn": "arn:aws:ivs:us-west-2:123456789012:playback-restriction-policy/ABcdef34ghIJ",
      "recordingConfigurationArn": "",
      "tags": {},
      "type": "STANDARD"
    }
  ]
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

示例 2：获取按指定 RecordingConfiguration ARN 筛选的所有频道的摘要信息



以下list-channels示例列出了您的 AWS 账户中与指定 RecordingConfiguration ARN 关联的所有频道。

```
aws ivs list-channels \  
  --filter-by-recording-configuration-arn "arn:aws:ivs:us-  
west-2:123456789012:recording-configuration/ABCD12cdEFgh"
```

输出：

```
{  
  "channels": [  
    {  
      "arn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",  
      "name": "channel-1",  
      "latencyMode": "LOW",  
      "authorized": false,  
      "insecureIngest": false,  
      "preset": "",  
      "playbackRestrictionPolicyArn": "",  
      "recordingConfigurationArn": "arn:aws:ivs:us-  
west-2:123456789012:recording-configuration/ABCD12cdEFgh",  
      "tags": {},  
      "type": "STANDARD"  
    }  
  ]  
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[录制到 Amazon S3](#)。

示例 3：获取按指定 PlaybackRestrictionPolicy ARN 筛选的所有频道的摘要信息

以下list-channels示例列出了您的 AWS 账户中与指定 PlaybackRestrictionPolicy ARN 关联的所有频道。

```
aws ivs list-channels \  
  --filter-by-playback-restriction-policy-arn "arn:aws:ivs:us-  
west-2:123456789012:playback-restriction-policy/ABcdef34ghIJ"
```

输出：

```
{
```

```
"channels": [  
  {  
    "arn": "arn:aws:ivs:us-west-2:123456789012:channel/efghEFGHijkl",  
    "name": "channel-2",  
    "latencyMode": "LOW",  
    "authorized": false,  
    "preset": "",  
    "playbackRestrictionPolicyArn": "arn:aws:ivs:us-west-2:123456789012:playback-restriction-policy/ABCdef34ghIJ",  
    "recordingConfigurationArn": "",  
    "tags": {},  
    "type": "STANDARD"  
  }  
]
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[不需要的内容和查看器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListChannels](#)中的。

## list-playback-key-pairs

以下代码示例演示了如何使用 `list-playback-key-pairs`。

### AWS CLI

获取有关所有播放密钥对的摘要信息

以下`list-playback-key-pairs`示例返回有关所有密钥对的信息。

```
aws ivs list-playback-key-pairs
```

输出：

```
{  
  "keyPairs": [  
    {  
      "arn": "arn:aws:ivs:us-west-2:123456789012:playback-key/abcd1234efgh",  
      "name": "test-key-0",  
      "tags": {}  
    },  
    {  
      "arn": "arn:aws:ivs:us-west-2:123456789012:playback-key/ijkl15678mnop",  
      "name": "test-key-1",  
      "tags": {}  
    }  
  ]  
}
```

```
        "name": "test-key-1",
        "tags": {}
      }
    ]
  }
```

有关更多信息，请参阅《Amazon Interactive Video Service 用户指南》中的[设置私有通道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPlaybackKeyPairs](#)中的。

## list-playback-restriction-policies

以下代码示例演示了如何使用 list-playback-restriction-policies。

### AWS CLI

获取有关所有播放限制政策的摘要信息

以下list-playback-restriction-policies示例列出了您 AWS 账号的所有播放限制政策。

```
aws ivs list-playback-restriction-policies
```

输出：

```
{
  "playbackRestrictionPolicies": [
    {
      "arn": "arn:aws:ivs:us-west-2:123456789012:playback-restriction-policy/ABcdef34ghIJ",
      "allowedCountries": [
        "US",
        "MX"
      ],
      "allowedOrigins": [
        "https://www.website1.com",
        "https://www.website2.com"
      ],
      "enableStrictOriginEnforcement": true,
      "name": "test-playback-restriction-policy",
      "tags": {
        "key1": "value1",
        "key2": "value2"
      }
    }
  ]
}
```

```
    }  
  ]  
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[不需要的内容和查看器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPlaybackRestrictionPolicies](#)中的。

## list-recording-configurations

以下代码示例演示了如何使用 list-recording-configurations。

### AWS CLI

列出在此账户中创建的所有 RecordingConfiguration 资源

以下list-recording-configurations示例获取有关您账户中所有 RecordingConfiguration 资源的信息。

```
aws ivs list-recording-configurations
```

输出：

```
{  
  "recordingConfigurations": [  
    {  
      "arn": "arn:aws:ivs:us-west-2:123456789012:recording-configuration/  
ABcdef34ghIJ",  
      "name": "test-recording-config-1",  
      "destinationConfiguration": {  
        "s3": {  
          "bucketName": "demo-recording-bucket-1"  
        }  
      },  
      "state": "ACTIVE",  
      "tags": {}  
    },  
    {  
      "arn": "arn:aws:ivs:us-west-2:123456789012:recording-configuration/  
CD12abcdGHIJ",  
      "name": "test-recording-config-2",  
      "destinationConfiguration": {
```

```
        "s3": {
            "bucketName": "demo-recording-bucket-2"
        },
        "state": "ACTIVE",
        "tags": {}
    }
]
}
```

有关更多信息，请参阅《[亚马逊互动视频服务用户指南](#)》中的“[录制到 Amazon S3](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRecordingConfigurations](#)中的。

## list-stream-keys

以下代码示例演示了如何使用 list-stream-keys。

### AWS CLI

获取直播密钥列表

以下list-stream-keys示例列出了指定 ARN ( Amazon 资源名称 ) 的所有直播密钥。

```
aws ivs list-stream-keys \
  --channel-arn arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh
```

输出：

```
{
  "streamKeys": [
    {
      "arn": "arn:aws:ivs:us-west-2:123456789012:stream-key/abcdABCDefgh",
      "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
      "tags": {}
    }
  ]
}
```

有关更多信息，请参阅《[IVS 低延迟用户指南](#)》中的[创建频道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStreamKeys](#)中的。

## list-stream-sessions

以下代码示例演示了如何使用 `list-stream-sessions`。

### AWS CLI

获取当前 AWS 区域中指定频道当前和之前直播的摘要

以下 `list-stream-sessions` 示例报告指定频道 ARN ( Amazon 资源名称 ) 的直播摘要信息。

```
aws ivs list-stream-sessions \
  --channel-arn arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh \
  --max-results 25 \
  --next-token ""
```

输出：

```
{
  "nextToken": "set-2",
  "streamSessions": [
    {
      "startTime": 1641578182,
      "endTime": 1641579982,
      "hasErrorEvent": false,
      "streamId": "mystream"
    }
    ...
  ]
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListStreamSessions](#) 中的。

## list-streams

以下代码示例演示了如何使用 `list-streams`。

### AWS CLI

获取直播列表及其状态

以下 `list-streams` 示例列出了您 AWS 账户的所有直播。

```
aws ivs list-streams
```

输出：

```
{
  "streams": [
    {
      "channelArn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
      "state": "LIVE",
      "health": "HEALTHY",
      "streamId": "st-ABCDEFghij01234KLMN5678",
      "viewerCount": 1
    }
  ]
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStreams](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

AWS CLI

列出 AWS 资源的所有标签（例如：频道、直播密钥）

以下`list-tags-for-resource`示例列出了指定资源 ARN（Amazon 资源名称）的所有标签。

```
aws ivs list-tags-for-resource \
  --resource-arn arn:aws:ivs:us-west-2:12345689012:channel/abcdABCDefgh
```

输出：

```
{
  "tags":
  {
    "key1": "value1",
    "key2": "value2"
  }
}
```

```
}
```

有关更多信息，请参阅《亚马逊互动视频服务 API 参考》中的[添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## put-metadata

以下代码示例演示了如何使用 put-metadata。

### AWS CLI

将元数据插入指定频道的活动直播中

以下put-metadata示例将给定元数据插入到指定频道的直播中。

```
aws ivs put-metadata \  
  --channel-arn arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh \  
  --metadata '{"my": "metadata"}'
```

此命令不生成任何输出。

有关更多信息，请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutMetadata](#)中的。

## start-viewer-session-revocation

以下代码示例演示了如何使用 start-viewer-session-revocation。

### AWS CLI

撤消给定多频道 ARN 和 Viewer-ID 对的观看者会话

以下start-viewer-session-revocation示例启动撤消与指定频道 ARN 和观众 ID 关联的观看者会话的过程，该会话不超过并包含指定的会话版本号。如果未提供版本，则默认为 0。

```
aws ivs batch-start-viewer-session-revocation \  
  --channel-arn arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh \  
  --viewer-id abcdefg \  
  --viewer-session-versions-less-than-or-equal-to 1234567890
```



此命令不生成任何输出。

有关更多信息，请参阅《Amazon Interactive Video Service 用户指南》中的[设置私有通道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartViewerSessionRevocation](#)中的。

## stop-stream

以下代码示例演示了如何使用 stop-stream。

### AWS CLI

停止指定直播

以下stop-stream示例在指定频道上停止直播。

```
aws ivs stop-stream \  
  --channel-arn arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh
```

此命令不生成任何输出。

有关更多信息，请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopStream](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为 AWS 资源添加或更新标签（例如：频道、直播密钥）

以下tag-resource示例为指定资源 ARN（Amazon 资源名称）添加或更新标签。

```
aws ivs tag-resource \  
  --resource-arn arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh \  
  --tags "tagkey1=tagvalue1, tagkey2=tagvalue2"
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊互动视频服务 API 参考》中的[添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 `untag-resource`。

### AWS CLI

移除 AWS 资源的标签（例如：频道、直播密钥）

以下 `untag-resource` 示例删除指定资源 ARN 的指定标签（Amazon 资源名称）。

```
aws ivs untag-resource \  
  --resource-arn arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh \  
  --tag-keys "tagkey1, tagkey2"
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊互动视频服务 API 参考》中的[添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-channel

以下代码示例演示了如何使用 `update-channel`。

### AWS CLI

示例 1：更新频道的配置信息

以下 `update-channel` 示例更新指定频道 ARN 的频道配置以更改频道名称。这不会影响此频道的持续直播；您必须停止并重新启动直播才能使更改生效。

```
aws ivs update-channel \  
  --arn arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh \  
  --name "channel-1" \  
  --insecure-ingest
```

输出：

```
{  
  "channel": {  
    "arn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
```

```

    "name": "channel-1",
    "latencyMode": "LOW",
    "type": "STANDARD",
    "playbackRestrictionPolicyArn": "",
    "recordingConfigurationArn": "",
    "srt": {
      "endpoint": "a1b2c3d4e5f6.srt.live-video.net",
      "passphrase":
"AB1C2defGHijkLMNo3PqQRstUvwxyzABCDEFghh4ijklMN5opqrStuVWxyzAbCDEfghIJ"
    },
    "ingestEndpoint": "a1b2c3d4e5f6.global-contribute.live-video.net",
    "insecureIngest": true,
    "playbackUrl": "https://a1b2c3d4e5f6.us-west-2.playback.live-video.net/api/
video/v1/us-west-2.123456789012.channel.abcdEFGH.m3u8",
    "preset": "",
    "authorized": false,
    "tags": {}
  }
}

```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[创建频道](#)。

#### 示例 2：更新频道配置以启用录制

以下 update-channel 示例更新了指定频道 ARN 的频道配置以启用录制。这不会影响此频道的持续直播；您必须停止并重新启动直播才能使更改生效。

```

aws ivs update-channel \
  --arn "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh" \
  --no-insecure-ingest \
  --recording-configuration-arn "arn:aws:ivs:us-west-2:123456789012:recording-
configuration/ABCD12cdEFgh"

```

输出：

```

{
  "channel": {
    "arn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
    "name": "test-channel-with-recording",
    "latencyMode": "LOW",
    "type": "STANDARD",
    "playbackRestrictionPolicyArn": "",
    "recordingConfigurationArn": "arn:aws:ivs:us-west-2:123456789012:recording-
configuration/ABCD12cdEFgh",

```

```

    "srt": {
      "endpoint": "a1b2c3d4e5f6.srt.live-video.net",
      "passphrase":
"BA1C2defGHijkLMNo3PqQRstUvwxyzaBCDEfghh4ijklMN5opqrStuVWxyzAbCDEfghIJ"
    },
    "ingestEndpoint": "a1b2c3d4e5f6.global-contribute.live-video.net",
    "insecureIngest": false,
    "playbackUrl": "https://a1b2c3d4e5f6.us-west-2.playback.live-video.net/api/
video/v1/us-west-2.123456789012.channel.abcdEFGH.m3u8",
    "preset": "",
    "authorized": false,
    "tags": {}
  }
}

```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[录制到 Amazon S3](#)。

### 示例 3：更新频道配置以禁用录制

以下update-channel示例更新了指定频道 ARN 的频道配置以禁用录制。这不会影响此频道的持续直播；您必须停止并重新启动直播才能使更改生效。

```

aws ivs update-channel \
  --arn "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh" \
  --recording-configuration-arn ""

```

输出：

```

{
  "channel": {
    "arn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
    "name": "test-channel-with-recording",
    "latencyMode": "LOW",
    "type": "STANDARD",
    "playbackRestrictionPolicyArn": "",
    "recordingConfigurationArn": "",
    "srt": {
      "endpoint": "a1b2c3d4e5f6.srt.live-video.net",
      "passphrase":
"AB1C2edfGHijkLMNo3PqQRstUvwxyzaBCDEfghh4ijklMN5opqrStuVWxyzAbCDEfghIJ"
    },
    "ingestEndpoint": "a1b2c3d4e5f6.global-contribute.live-video.net",
  }
}

```

```

    "insecureIngest": false,
    "playbackUrl": "https://a1b2c3d4e5f6.us-west-2.playback.live-video.net/api/
video/v1/us-west-2.123456789012.channel.abcdEFGH.m3u8",
    "preset": "",
    "authorized": false,
    "tags": {}
  }
}

```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[录制到 Amazon S3](#)。

#### 示例 4：更新频道配置以启用播放限制

以下update-channel示例更新指定频道 ARN 的频道配置以应用播放限制策略。这不会影响此频道的持续直播；您必须停止并重新启动直播才能使更改生效。

```

aws ivs update-channel \
  --arn "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh" \
  --no-insecure-ingest \
  --playback-restriction-policy-arn "arn:aws:ivs:us-west-2:123456789012:playback-
restriction-policy/ABcdef34ghIJ"

```

输出：

```

{
  "channel": {
    "arn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
    "name": "test-channel-with-playback-restriction-policy",
    "latencyMode": "LOW",
    "type": "STANDARD",
    "playbackRestrictionPolicyArn": "arn:aws:ivs:us-
west-2:123456789012:playback-restriction-policy/ABcdef34ghIJ",
    "recordingConfigurationArn": "",
    "srt": {
      "endpoint": "a1b2c3d4e5f6.srt.live-video.net",
      "passphrase":
"AB1C2defGHijklMN03PqQRstUvwxyzaCBDEfghh4ijklMN5opqrStuVWxyzAbCDEfghIJ"
    },
    "ingestEndpoint": "a1b2c3d4e5f6.global-contribute.live-video.net",
    "insecureIngest": false,
    "playbackUrl": "https://a1b2c3d4e5f6.us-west-2.playback.live-video.net/api/
video/v1/us-west-2.123456789012.channel.abcdEFGH.m3u8",
    "preset": "",

```

```

    "authorized": false,
    "tags": {}
  }
}

```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[不需要的内容和查看器](#)。

#### 示例 5：更新频道配置以禁用播放限制

以下update-channel示例更新了指定频道 ARN 的频道配置以禁用播放限制。这不会影响此频道的持续直播；您必须停止并重新启动直播才能使更改生效。

```

aws ivs update-channel \
  --arn "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh" \
  --playback-restriction-policy-arn ""

```

输出：

```

{
  "channel": {
    "arn": "arn:aws:ivs:us-west-2:123456789012:channel/abcdABCDefgh",
    "name": "test-channel-with-playback-restriction-policy",
    "latencyMode": "LOW",
    "type": "STANDARD",
    "playbackRestrictionPolicyArn": "",
    "recordingConfigurationArn": "",
    "srt": {
      "endpoint": "a1b2c3d4e5f6.srt.live-video.net",
      "passphrase":
"AB1C2defGHijkLMNo3PqQRstUvwxyzaBCDeFghh4ijklMN5opqrStuVWxyzAbCDEfghIJ"
    },
    "ingestEndpoint": "a1b2c3d4e5f6.global-contribute.live-video.net",
    "insecureIngest": false,
    "playbackUrl": "https://a1b2c3d4e5f6.us-west-2.playback.live-video.net/api/
video/v1/us-west-2.123456789012.channel.abcdEFGH.m3u8",
    "preset": "",
    "authorized": false,
    "tags": {}
  }
}

```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[不需要的内容和查看器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateChannel](#)中的。

## update-playback-restriction-policy

以下代码示例演示了如何使用 `update-playback-restriction-policy`。

### AWS CLI

#### 更新播放限制政策

以下`update-playback-restriction-policy`示例使用指定的策略 ARN 更新播放限制策略，以禁用严格的来源强制执行。这不会影响关联频道的持续直播；您必须停止并重新启动直播才能使更改生效。

```
aws ivs update-playback-restriction-policy \  
  --arn "arn:aws:ivs:us-west-2:123456789012:playback-restriction-policy/  
ABCdef34ghIJ" \  
  --no-enable-strict-origin-enforcement
```

输出：

```
{  
  "playbackRestrictionPolicy": {  
    "arn": "arn:aws:ivs:us-west-2:123456789012:playback-restriction-policy/  
ABCdef34ghIJ",  
    "allowedCountries": [  
      "US",  
      "MX"  
    ],  
    "allowedOrigins": [  
      "https://www.website1.com",  
      "https://www.website2.com"  
    ],  
    "enableStrictOriginEnforcement": false,  
    "name": "test-playback-restriction-policy",  
    "tags": {  
      "key1": "value1",  
      "key2": "value2"  
    }  
  }  
}
```

有关更多信息，请参阅《IVS 低延迟用户指南》中的[不需要的内容和查看器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePlaybackRestrictionPolicy](#)中的。

## 使用 Amazon IVS 聊天示例 AWS CLI

以下代码示例向您展示了如何使用 with Amazon IVS Chat 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### create-chat-token

以下代码示例演示了如何使用 create-chat-token。

AWS CLI

创建聊天令牌

以下create-chat-token示例创建了一个加密的聊天令牌，该令牌用于建立与房间的个人WebSocket连接。令牌的有效期为一分钟，使用该令牌建立的连接（会话）在指定的持续时间内有效。

```
aws ivschat create-chat-token \  
  --roomIdentifier "arn:aws:ivschat:us-west-2:12345689012:room/g1H2I3j4k5L6", \  
  --userId "11231234" \  
  --capabilities "SEND_MESSAGE", \  
  --sessionDurationInMinutes" 30
```

输出：



```
{
  "token": "ACEGmnoq#1rstu2...BDFH3vxyw!4hlm!#5",
  "sessionExpirationTime": "2022-03-16T04:44:09+00:00"
  "state": "CREATING",
  "tokenExpirationTime": "2022-03-16T03:45:09+00:00"
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[步骤 3：对聊天客户端进行身份验证和授权](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateChatToken](#)中的。

## create-logging-configuration

以下代码示例演示了如何使用 create-logging-configuration。

### AWS CLI

创建聊天 LoggingConfiguration 资源

以下 create-logging-configuration 示例创建了一个 LoggingConfiguration 资源，允许客户端存储和记录已发送的消息。

```
aws ivschat create-logging-configuration \
  --destination-configuration s3={bucketName=demo-logging-bucket} \
  --name "test-logging-config" \
  --tags "key1=value1, key2=value2"
```

输出：

```
{
  "arn": "arn:aws:ivschat:us-west-2:123456789012:logging-configuration/
ABcdef34ghIJ",
  "createTime": "2022-09-14T17:48:00.653000+00:00",
  "destinationConfiguration": {
    "s3": {
      "bucketName": "demo-logging-bucket"
    }
  },
  "id": "ABcdef34ghIJ",
  "name": "test-logging-config",
```

```
"state": "ACTIVE",
"tags": { "key1" : "value1", "key2" : "value2" },
"updateTime": "2022-09-14T17:48:01.104000+00:00"
}
```

有关更多信息，请参阅《[亚马逊互动视频服务用户指南](#)》中的 [Amazon IVS 聊天入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateLoggingConfiguration](#) 中的。

## create-room

以下代码示例演示了如何使用 create-room。

### AWS CLI

#### 创建房间

以下 create-room 示例创建了一个新房间。

```
aws ivschat create-room \
  --name "test-room-1" \
  --logging-configuration-identifiers "arn:aws:ivschat:us-
west-2:123456789012:logging-configuration/ABCdef34ghIJ" \
  --maximum-message-length 256 \
  --maximum-message-rate-per-second 5
```

输出：

```
{
  "arn": "arn:aws:ivschat:us-west-2:123456789012:room/g1H2I3j4k5L6",
  "id": "g1H2I3j4k5L6",
  "createTime": "2022-03-16T04:44:09+00:00",
  "loggingConfigurationIdentifiers": ["arn:aws:ivschat:us-
west-2:123456789012:logging-configuration/ABCdef34ghIJ"],
  "maximumMessageLength": 256,
  "maximumMessageRatePerSecond": 5,
  "name": "test-room-1",
  "tags": {}
  "updateTime": "2022-03-16T07:22:09+00:00"
}
```

有关更多信息，请参阅《[亚马逊互动视频服务用户指南](#)》中的 [步骤 2：创建聊天室](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRoom](#)中的。

## delete-logging-configuration

以下代码示例演示了如何使用 delete-logging-configuration。

### AWS CLI

删除聊天 LoggingConfiguration 资源

以下delete-logging-configuration示例删除指定 ARN 的 LoggingConfiguration 资源。

```
aws ivschat delete-logging-configuration \  
  --identifier "arn:aws:ivschat:us-west-2:123456789012:logging-configuration/  
  ABcdef34ghIJ"
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊互动视频服务用户指南](#)》中的 [Amazon IVS 聊天入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLoggingConfiguration](#)中的。

## delete-message

以下代码示例演示了如何使用 delete-message。

### AWS CLI

从指定房间删除消息

以下delete-message示例向指定的聊天室发送一个偶数，指示客户端删除指定的消息：也就是说，取消呈现该消息，然后将其从客户端的聊天记录中删除。

```
aws ivschat delete-message \  
  --roomIdIdentifier "arn:aws:ivschat:us-west-2:12345689012:room/g1H2I3j4k5L6" \  
  --id "ABC123def456" \  
  --reason "Message contains profanity"
```

输出：

```
{
```

```
"id": "12345689012"  
}
```

有关更多信息，请参阅《[亚马逊互动视频服务用户指南](#)》中的 [Amazon IVS 聊天入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMessage](#)中的。

## delete-room

以下代码示例演示了如何使用 delete-room。

### AWS CLI

#### 删除房间

以下delete-room示例删除了指定的房间。已连接的客户端已断开连接。成功后，它将返回带有空响应正文的 HTTP 204。

```
aws ivschat delete-room \  
  --identifier "arn:aws:ivschat:us-west-2:12345689012:room/g1H2I3j4k5L6"
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊互动视频服务用户指南](#)》中的 [Amazon IVS 聊天入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRoom](#)中的。

## disconnect-user

以下代码示例演示了如何使用 disconnect-user。

### AWS CLI

#### 断开用户与房间的连接

以下disconnect-user示例断开指定用户与指定房间的所有连接。成功后，它将返回带有空响应正文的 HTTP 200。

```
aws ivschat disconnect-user \  
  --roomIdIdentifier "arn:aws:ivschat:us-west-2:12345689012:room/g1H2I3j4k5L6" \  
  --userId "ABC123def456" \  
  --reason "Violated terms of service"
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊互动视频服务用户指南](#)》中的 [Amazon IVS 聊天入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisconnectUser](#)中的。

## get-logging-configuration

以下代码示例演示了如何使用 get-logging-configuration。

### AWS CLI

获取有关 LoggingConfiguration 资源的信息

以下get-logging-configuration示例获取有关指定 ARN 的 LoggingConfiguration 资源的信息。

```
aws ivschat get-logging-configuration \  
  --identifier "arn:aws:ivschat:us-west-2:123456789012:logging-configuration/  
  ABcdef34ghIJ"
```

输出：

```
{  
  "arn": "arn:aws:ivschat:us-west-2:123456789012:logging-configuration/  
  ABcdef34ghIJ",  
  "createTime": "2022-09-14T17:48:00.653000+00:00",  
  "destinationConfiguration": {  
    "s3": {  
      "bucketName": "demo-logging-bucket"  
    }  
  },  
  "id": "ABcdef34ghIJ",  
  "name": "test-logging-config",  
  "state": "ACTIVE",  
  "tags": { "key1" : "value1", "key2" : "value2" },  
  "updateTime": "2022-09-14T17:48:01.104000+00:00"  
}
```

有关更多信息，请参阅《[亚马逊互动视频服务用户指南](#)》中的 [Amazon IVS 聊天入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLoggingConfiguration](#)中的。

## get-room

以下代码示例演示了如何使用 `get-room`。

### AWS CLI

获取指定房间

以下 `get-room` 示例获取有关指定房间的信息。

```
aws ivschat get-room \  
  --identifier "arn:aws:ivschat:us-west-2:12345689012:room/g1H2I3j4k5L6"
```

输出：

```
{  
  "arn": "arn:aws:ivschat:us-west-2:12345689012:room/g1H2I3j4k5L6",  
  "createTime": "2022-03-16T04:44:09+00:00",  
  "id": "g1H2I3j4k5L6",  
  "loggingConfigurationIdentifiers": ["arn:aws:ivschat:us-  
west-2:123456789012:logging-configuration/ABCdef34ghIJ"],  
  "maximumMessageLength": 256,  
  "maximumMessageRatePerSecond": 5,  
  "name": "test-room-1",  
  "tags": {},  
  "updateTime": "2022-03-16T07:22:09+00:00"  
}
```

有关更多信息，请参阅 [《亚马逊互动视频服务用户指南》中的 Amazon IVS 聊天入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetRoom](#) 中的。

## list-logging-configurations

以下代码示例演示了如何使用 `list-logging-configurations`。

### AWS CLI

获取有关处理 API 请求的 AWS 区域中用户的所有日志配置的摘要信息

以下 `list-logging-configurations` 示例列出了处理 API 请求的 AWS 区域中该用户的所有 `LoggingConfiguration` 资源的信息。

```
aws ivschat list-logging-configurations \  
  --max-results 2 \  
  --next-token ""
```

输出：

```
{  
  "nextToken": "set-2",  
  "loggingConfigurations": [  
    {  
      "arn": "arn:aws:ivschat:us-west-2:123456789012:logging-configuration/  
ABCdef34ghIJ",  
      "createTime": "2022-09-14T17:48:00.653000+00:00",  
      "destinationConfiguration": {  
        "s3": {  
          "bucketName": "demo-logging-bucket"  
        }  
      },  
      "id": "ABCdef34ghIJ",  
      "name": "test-logging-config",  
      "state": "ACTIVE",  
      "tags": { "key1" : "value1", "key2" : "value2" },  
      "updateTime": "2022-09-14T17:48:01.104000+00:00"  
    }  
    ...  
  ]  
}
```

有关更多信息，请参阅 [《亚马逊互动视频服务用户指南》中的 Amazon IVS 聊天入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListLoggingConfigurations](#) 中的。

## list-rooms

以下代码示例演示了如何使用 `list-rooms`。

### AWS CLI

获取当前区域内所有房间的摘要信息

以下 `list-rooms` 示例获取有关处理请求的 AWS 区域中所有房间的摘要信息。结果按照 `updateTime` 的降序排序。

```
aws ivschat list-rooms \  
  --logging-configuration-identifier "arn:aws:ivschat:us-  
west-2:123456789012:logging-configuration/ABCdef34ghIJ" \  
  --max-results 10 \  
  --next-token ""
```

输出：

```
{  
  "nextToken": "page3",  
  "rooms": [  
    {  
      "arn:aws:ivschat:us-west-2:12345689012:room/g1H2I3j4k5L6",  
      "createTime": "2022-03-16T04:44:09+00:00",  
      "id": "g1H2I3j4k5L6",  
      "loggingConfigurationIdentifiers": ["arn:aws:ivschat:us-  
west-2:123456789012:logging-configuration/ABCdef34ghIJ"],  
      "name": "test-room-1",  
      "tags": {},  
      "updateTime": "2022-03-16T07:22:09+00:00"  
    }  
  ]  
}
```

有关更多信息，请参阅《[亚马逊互动视频服务用户指南](#)》中的 [Amazon IVS 聊天入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRooms](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 list-tags-for-resource。

### AWS CLI

列出 AWS 资源的所有标签（例如：房间）

以下list-tags-for-resource示例列出了指定资源 ARN（Amazon 资源名称）的所有标签。

```
aws ivschat list-tags-for-resource \  
  --resource-arn arn:aws:ivschat:us-west-2:12345689012:room/g1H2I3j4k5L6
```

输出：



```
{
  "tags":
  {
    "key1": "value1",
    "key2": "value2"
  }
}
```

有关更多信息，请参阅《[亚马逊互动视频服务 API 参考](#)》中的[添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## send-event

以下代码示例演示了如何使用 send-event。

### AWS CLI

#### 向房间发送活动

以下send-event示例将给定事件发送到指定的房间。

```
aws ivschat send-event \
  --roomIdentifier "arn:aws:ivschat:us-west-2:12345689012:room/g1H2I3j4k5L6" \
  --eventName "SystemMessage" \
  --attributes \
    "msgType"="user-notification", \
    "msgText"="This chat room will close in 15 minutes."
```

输出：

```
{
  "id": "12345689012"
}
```

有关更多信息，请参阅《[亚马逊互动视频服务用户指南](#)》中的[Amazon IVS 聊天入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SendEvent](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

## AWS CLI

为 AWS 资源添加或更新标签 ( 例如 : 房间 )

以下tag-resource示例为指定资源 ARN ( Amazon 资源名称 ) 添加或更新标签。成功后 , 它将返回带有空响应正文的 HTTP 200。

```
aws ivschat tag-resource \  
  --resource-arn arn:aws:ivschat:us-west-2:12345689012:room/g1H2I3j4k5L6 \  
  --tags "tagkey1=tagkeyvalue1, tagkey2=tagkeyvalue2"
```

此命令不生成任何输出。

有关更多信息 , 请参阅《亚马逊互动视频服务 API 参考》中的[添加标签](#)。

- 有关 API 的详细信息 , 请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

## AWS CLI

移除 AWS 资源的标签 ( 例如 : 房间 )

以下untag-resource示例删除指定资源 ARN 的指定标签 ( Amazon 资源名称 )。成功后 , 它将返回带有空响应正文的 HTTP 200。

```
aws ivschat untag-resource \  
  --resource-arn arn:aws:ivschat:us-west-2:12345689012:room/g1H2I3j4k5L6 \  
  --tag-keys "tagkey1, tagkey2"
```

此命令不生成任何输出。

有关更多信息 , 请参阅《亚马逊互动视频服务 API 参考》中的[添加标签](#)。

- 有关 API 的详细信息 , 请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-logging-configuration

以下代码示例演示了如何使用 update-logging-configuration。

## AWS CLI

### 更新房间的日志配置

以下update-logging-configuration示例使用给定数据更新 LoggingConfiguration 资源。

```
aws ivschat update-logging-configuration \  
  --destination-configuration s3={bucketName=demo-logging-bucket} \  
  --identifier "arn:aws:ivschat:us-west-2:123456789012:logging-configuration/  
ABcdef34ghIJ" \  
  --name "test-logging-config"
```

输出：

```
{  
  "arn": "arn:aws:ivschat:us-west-2:123456789012:logging-configuration/  
ABcdef34ghIJ",  
  "createTime": "2022-09-14T17:48:00.653000+00:00",  
  "destinationConfiguration": {  
    "s3": {  
      "bucketName": "demo-logging-bucket"  
    }  
  },  
  "id": "ABcdef34ghIJ",  
  "name": "test-logging-config",  
  "state": "ACTIVE",  
  "tags": { "key1" : "value1", "key2" : "value2" },  
  "updateTime": "2022-09-14T17:48:01.104000+00:00"  
}
```

有关更多信息，请参阅 [《亚马逊互动视频服务用户指南》](#) 中的 [Amazon IVS 聊天入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLoggingConfiguration](#)中的。

## update-room

以下代码示例演示了如何使用 update-room。

## AWS CLI

### 更新房间的配置

以下update-room示例使用给定数据更新指定房间的配置。

```
aws ivschat update-room \  
  --identifier "arn:aws:ivschat:us-west-2:12345689012:room/g1H2I3j4k5L6" \  
  --logging-configuration-identifiers "arn:aws:ivschat:us-  
west-2:123456789012:logging-configuration/ABCdef34ghIJ" \  
  --name "chat-room-a" \  
  --maximum-message-length 256 \  
  --maximum-message-rate-per-second 5
```

输出：

```
{  
  "arn": "arn:aws:ivschat:us-west-2:12345689012:room/g1H2I3j4k5L6",  
  "createTime": "2022-03-16T04:44:09+00:00",  
  "id": "g1H2I3j4k5L6",  
  "loggingConfigurationIdentifiers": ["arn:aws:ivschat:us-  
west-2:123456789012:logging-configuration/ABCdef34ghIJ"],  
  "maximumMessageLength": 256,  
  "maximumMessageRatePerSecond": 5,  
  "name": "chat-room-a",  
  "tags": {},  
  "updateTime": "2022-03-16T07:22:09+00:00"  
}
```

有关更多信息，请参阅 [《亚马逊互动视频服务用户指南》中的 Amazon IVS 聊天入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateRoom](#) 中的。

## 使用 Amazon IVS 实时直播示例 AWS CLI

以下代码示例向您展示如何使用 AWS Command Line Interface 与 Amazon IVS 实时流媒体配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### create-encoder-configuration

以下代码示例演示了如何使用 create-encoder-configuration。

#### AWS CLI

##### 创建合成编码器配置

以下create-encoder-configuration示例创建具有指定属性的合成编码器配置。

```
aws ivs-realtime create-encoder-configuration \  
  --name test-ec --video bitrate=3500000,framerate=30.0,height=1080,width=1920
```

输出：

```
{  
  "encoderConfiguration": {  
    "arn": "arn:aws:ivs:ap-northeast-1:123456789012:encoder-configuration/  
ABabCDcdEFef",  
    "name": "test-ec",  
    "tags": {},  
    "video": {  
      "bitrate": 3500000,  
      "framerate": 30,  
      "height": 1080,  
      "width": 1920  
    }  
  }  
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateEncoderConfiguration](#)中的。

### create-participant-token

以下代码示例演示了如何使用 create-participant-token。

## AWS CLI

### 创建舞台参与者令牌

以下create-participant-token示例为指定阶段创建参与者令牌。

```
aws ivs-realtime create-participant-token \  
  --stage-arn arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh \  
  --user-id bob
```

输出：

```
{  
  "participantToken": {  
    "expirationTime": "2023-03-07T09:47:43+00:00",  
    "participantId": "ABCDEFghij01234KLMN6789",  
    "token": "abcd1234defg5678"  
  }  
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateParticipantToken](#)中的。

## create-stage

以下代码示例演示了如何使用 create-stage。

## AWS CLI

### 创建舞台

以下create-stage示例为指定用户创建舞台和舞台参与者令牌。

```
aws ivs-realtime create-stage \  
  --name stage1 \  
  --participant-token-configurations userId=alice
```

输出：

```
{
```

```
"participantTokens": [
  {
    "participantId": "ABCDEFghij01234KLMN5678",
    "token": "a1b2c3d4567890ab",
    "userId": "alice"
  }
],
"stage": {
  "activeSessionId": "st-a1b2c3d4e5f6g",
  "arn": "arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh",
  "name": "stage1",
  "tags": {}
}
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateStage](#)中的。

## create-storage-configuration

以下代码示例演示了如何使用 create-storage-configuration。

### AWS CLI

#### 创建合成存储配置

以下create-storage-configuration示例创建具有指定属性的合成存储配置。

```
aws ivs-realtime create-storage-configuration \
  --name "test-sc" --s3 "bucketName=test-bucket-name"
```

输出：

```
{
  "storageConfiguration": {
    "arn": "arn:aws:ivs:ap-northeast-1:123456789012:storage-configuration/
ABabCDcdEFef",
    "name": "test-sc",
    "s3": {
      "bucketName": "test-bucket-name"
    }
  },
}
```

```
    "tags": {}  
  }  
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateStorageConfiguration](#)中的。

## delete-encoder-configuration

以下代码示例演示了如何使用 delete-encoder-configuration。

### AWS CLI

#### 删除合成编码器配置

以下内容delete-encoder-configuration删除由给定 ARN ( Amazon 资源名称 ) 指定的合成编码器配置。

```
aws ivs-realtime delete-encoder-configuration \  
    --arn "arn:aws:ivs:ap-northeast-1:123456789012:encoder-configuration/  
ABabCDcdEFef"
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteEncoderConfiguration](#)中的。

## delete-stage

以下代码示例演示了如何使用 delete-stage。

### AWS CLI

#### 删除舞台

以下delete-stage示例删除了指定的阶段。

```
aws ivs-realtime delete-stage \  

```



```
--arn arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteStage](#)中的。

## delete-storage-configuration

以下代码示例演示了如何使用 delete-storage-configuration。

### AWS CLI

#### 删除合成存储配置

以下内容delete-storage-configuration删除由给定 ARN ( Amazon 资源名称 ) 指定的合成存储配置。

```
aws ivs-realtime delete-storage-configuration \  
    --arn "arn:aws:ivs:ap-northeast-1:123456789012:storage-configuration/  
ABabCDcdEFef"
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteStorageConfiguration](#)中的。

## disconnect-participant

以下代码示例演示了如何使用 disconnect-participant。

### AWS CLI

#### 断开舞台参与者的连接

以下disconnect-participant示例断开指定参与者与指定阶段的连接。

```
aws ivs-realtime disconnect-participant \  
    --stage-name "stage-name" --participant-name "participant-name"
```

```
--stage-arn arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh \  
--participant-id ABCDEfghij01234KLMN5678
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DisconnectParticipant](#)中的。

## get-composition

以下代码示例演示了如何使用 `get-composition`。

### AWS CLI

示例 1：使用默认布局设置获取合成

以下 `get-composition` 示例获取指定的 ARN（亚马逊资源名称）的构成。

```
aws ivs-realtime get-composition \  
--arn "arn:aws:ivs:ap-northeast-1:123456789012:composition/abcdABCDefgh"
```

输出：

```
{  
  "composition": {  
    "arn": "arn:aws:ivs:ap-northeast-1:123456789012:composition/abcdABCDefgh",  
    "destinations": [  
      {  
        "configuration": {  
          "channel": {  
            "channelArn": "arn:aws:ivs:ap-northeast-1:123456789012:channel/abcABCdefDEg",  
            "encoderConfigurationArn": "arn:aws:ivs:ap-northeast-1:123456789012:encoder-configuration/ABabCDcdEFef"  
          },  
          "name": ""  
        },  
        "id": "AabBCcdDEefF",  
        "startTime": "2023-10-16T23:26:00+00:00",  
        "state": "ACTIVE"  
      },  
    ],  
  },  
}
```

```

    {
      "configuration": {
        "name": "",
        "s3": {
          "encoderConfigurationArns": [
            "arn:aws:ivs:arn:aws:ivs:ap-
northeast-1:123456789012:encoder-configuration/ABabCDcdEFef"
          ],
          "recordingConfiguration": {
            "format": "HLS"
          },
          "storageConfigurationArn": "arn:arn:aws:ivs:ap-
northeast-1:123456789012:storage-configuration/FefABabCDcdE"
        }
      },
      "detail": {
        "s3": {
          "recordingPrefix": "aBcDeFgHhGfE/AbCdEfGhHgFe/GHfabcgefABC/
composite"
        }
      },
      "id": "GHfabcgefABC",
      "startTime": "2023-10-16T23:26:00+00:00",
      "state": "STARTING"
    }
  ],
  "layout": {
    "grid": {
      "featuredParticipantAttribute": ""
      "gridGap": 2,
      "omitStoppedVideo": false,
      "videoAspectRatio": "VIDEO",
      "videoFillMode": ""
    }
  },
  "stageArn": "arn:aws:ivs:ap-northeast-1:123456789012:stage/defgABCDabcd",
  "startTime": "2023-10-16T23:24:00+00:00",
  "state": "ACTIVE",
  "tags": {}
}
}

```

有关更多信息，请参阅 Amazon 互动视频服务用户指南中的[复合录制（实时流媒体）](#)。

示例 2：使用画中画布局获取合成

以下`get-composition`示例获取指定的 ARN ( 亚马逊资源名称 ) 的构成, 该名称使用 PiP 布局。

```
aws ivs-realtime get-composition \  
  --arn "arn:aws:ivs:ap-northeast-1:123456789012:composition/wxyzWXYZpqrs"
```

输出:

```
{  
  "composition": {  
    "arn": "arn:aws:ivs:ap-northeast-1:123456789012:composition/wxyzWXYZpqrs",  
    "destinations": [  
      {  
        "configuration": {  
          "channel": {  
            "channelArn": "arn:aws:ivs:ap-northeast-1:123456789012:channel/abcABCdefDEg",  
            "encoderConfigurationArn": "arn:aws:ivs:ap-northeast-1:123456789012:encoder-configuration/ABabCDcdEFef"  
          },  
          "name": ""  
        },  
        "id": "AabBCcdDEeff",  
        "startTime": "2023-10-16T23:26:00+00:00",  
        "state": "ACTIVE"  
      },  
      {  
        "configuration": {  
          "name": "",  
          "s3": {  
            "encoderConfigurationArns": [  
              "arn:aws:ivs:arn:aws:ivs:ap-northeast-1:123456789012:encoder-configuration/ABabCDcdEFef"  
            ],  
            "recordingConfiguration": {  
              "format": "HLS"  
            },  
            "storageConfigurationArn": "arn:arn:aws:ivs:ap-northeast-1:123456789012:storage-configuration/FefABabCDcdE"  
          }  
        },  
        "detail": {  
          "s3": {
```

```
        "recordingPrefix": "aBcDeFgHhGfE/AbCdEfGhHgFe/GHFabcgefABC/
composite"
        }
    },
    "id": "GHFabcgefABC",
    "startTime": "2023-10-16T23:26:00+00:00",
    "state": "STARTING"
}
],
"layout": {
    "pip": {
        "featuredParticipantAttribute": "abcdefg",
        "gridGap": 0,
        "omitStoppedVideo": false,
        "pipBehavior": "STATIC",
        "pipOffset": 0,
        "pipParticipantAttribute": "",
        "pipPosition": "BOTTOM_RIGHT",
        "videoFillMode": "COVER"
    }
},
"stageArn": "arn:aws:ivs:ap-northeast-1:123456789012:stage/defgABCdabcd",
"startTime": "2023-10-16T23:24:00+00:00",
"state": "ACTIVE",
"tags": {}
}
}
```

有关更多信息，请参阅 Amazon 互动视频服务用户指南中的[复合录制（实时流媒体）](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetComposition](#)中的。

## get-encoder-configuration

以下代码示例演示了如何使用 get-encoder-configuration。

### AWS CLI

#### 获取合成编码器配置

以下 get-encoder-configuration 示例获取由给定 ARN（Amazon 资源名称）指定的合成编码器配置。

```
aws ivs-realtime get-encoder-configuration \  
  --arn "arn:aws:ivs:ap-northeast-1:123456789012:encoder-configuration/  
abcdABCDefgh"
```

输出：

```
{  
  "encoderConfiguration": {  
    "arn": "arn:aws:ivs:ap-northeast-1:123456789012:encoder-configuration/  
abcdABCDefgh",  
    "name": "test-ec",  
    "tags": {},  
    "video": {  
      "bitrate": 3500000,  
      "framerate": 30,  
      "height": 1080,  
      "width": 1920  
    }  
  }  
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetEncoderConfiguration](#)中的。

## get-participant

以下代码示例演示了如何使用 get-participant。

### AWS CLI

#### 招募舞台参与者

以下get-participant示例获取指定阶段 ARN ( Amazon 资源名称 ) 中指定参与者 ID 和会话 ID 的阶段参与者。

```
aws ivs-realtime get-participant \  
  --stage-arn arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh \  
  --session-id st-a1b2c3d4e5f6g \  
  --participant-id abCDEf12GHIj
```

输出：

```
{
  "participant": {
    "browserName", "Google Chrome",
    "browserVersion", "116",
    "firstJoinTime": "2023-04-26T20:30:34+00:00",
    "ispName", "Comcast",
    "osName", "Microsoft Windows 10 Pro",
    "osVersion", "10.0.19044",
    "participantId": "abCDEf12GHIj",
    "published": true,
    "sdkVersion", "",
    "state": "DISCONNECTED",
    "userId": ""
  }
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetParticipant](#)中的。

## get-stage-session

以下代码示例演示了如何使用 get-stage-session。

### AWS CLI

#### 要参加舞台演出

以下get-stage-session示例获取指定阶段 ARN 的指定会话 ID 的阶段会话 ( Amazon 资源名称 )。

```
aws ivs-realtime get-stage-session \
  --stage-arn arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh \
  --session-id st-a1b2c3d4e5f6g
```

输出：

```
{
```

```
"stageSession": {
  "endTime": "2023-04-26T20:36:29+00:00",
  "sessionId": "st-a1b2c3d4e5f6g",
  "startTime": "2023-04-26T20:30:29.602000+00:00"
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetStageSession](#)中的。

## get-stage

以下代码示例演示了如何使用 get-stage。

### AWS CLI

获取舞台的配置信息

以下get-stage示例获取指定阶段 ARN 的阶段配置 ( Amazon 资源名称 )。

```
aws ivs-realtime get-stage \
  --arn arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh
```

输出：

```
{
  "stage": {
    "activeSessionId": "st-a1b2c3d4e5f6g",
    "arn": "arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh",
    "name": "test",
    "tags": {}
  }
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetStage](#)中的。



## get-storage-configuration

以下代码示例演示了如何使用 `get-storage-configuration`。

### AWS CLI

#### 获取合成存储配置

以下 `get-storage-configuration` 示例获取由给定 ARN ( Amazon 资源名称 ) 指定的组合存储配置。

```
aws ivs-realtime get-storage-configuration \  
  --name arn "arn:aws:ivs:ap-northeast-1:123456789012:storage-configuration/  
abcdABCDefgh"
```

输出：

```
{  
  "storageConfiguration": {  
    "arn": "arn:aws:ivs:ap-northeast-1:123456789012:storage-configuration/  
abcdABCDefgh",  
    "name": "test-sc",  
    "s3": {  
      "bucketName": "test-bucket-name"  
    },  
    "tags": {}  
  }  
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetStorageConfiguration](#)中的。

## list-compositions

以下代码示例演示了如何使用 `list-compositions`。

### AWS CLI

#### 要获取作品清单

以下 `list-compositions` 列出了您 AWS 账户在处理 API 请求的 AWS 区域内的所有组合信息。

```
aws ivs-realtime list-compositions
```

输出：

```
{
  "compositions": [
    {
      "arn": "arn:aws:ivs:ap-northeast-1:123456789012:composition/
abcdABCDefgh",
      "destinations": [
        {
          "id": "AabBCcdDEeff",
          "startTime": "2023-10-16T23:25:23+00:00",
          "state": "ACTIVE"
        }
      ],
      "stageArn": "arn:aws:ivs:ap-northeast-1:123456789012:stage/
defgABCDabcd",
      "startTime": "2023-10-16T23:25:21+00:00",
      "state": "ACTIVE",
      "tags": {}
    },
    {
      "arn": "arn:aws:ivs:ap-northeast-1:123456789012:composition/
ABcdabCDefgh",
      "destinations": [
        {
          "endTime": "2023-10-16T23:25:00.786512+00:00",
          "id": "aABbcCDdeEFf",
          "startTime": "2023-10-16T23:24:01+00:00",
          "state": "STOPPED"
        },
        {
          "endTime": "2023-10-16T23:25:00.786512+00:00",
          "id": "deEFfaABbcCD",
          "startTime": "2023-10-16T23:24:01+00:00",
          "state": "STOPPED"
        }
      ],
      "endTime": "2023-10-16T23:25:00+00:00",
      "stageArn": "arn:aws:ivs:ap-northeast-1:123456789012:stage/
efghabcdABCD",
      "startTime": "2023-10-16T23:24:00+00:00",
    }
  ]
}
```

```
        "state": "STOPPED",
        "tags": {}
      }
    ]
  }
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCompositions](#)中的。

## list-encoder-configurations

以下代码示例演示了如何使用 list-encoder-configurations。

### AWS CLI

#### 列出合成编码器配置

以下list-encoder-configurations列出了在处理 API 请求的 AWS 区域中您的 AWS 账户的所有合成编码器配置。

```
aws ivs-realtime list-encoder-configurations
```

输出：

```
{
  "encoderConfigurations": [
    {
      "arn": "arn:aws:ivs:ap-northeast-1:123456789012:encoder-configuration/abcdABCDefgh",
      "name": "test-ec-1",
      "tags": {}
    },
    {
      "arn": "arn:aws:ivs:ap-northeast-1:123456789012:encoder-configuration/ABCefgEFGabc",
      "name": "test-ec-2",
      "tags": {}
    }
  ]
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListEncoderConfigurations](#)中的。

## list-participant-events

以下代码示例演示了如何使用 list-participant-events。

### AWS CLI

获取舞台参与者活动列表

以下list-participant-events示例列出了指定参与者 ID 和指定阶段 ARN 的会话 ID ( Amazon 资源名称 ) 的所有参与者事件。

```
aws ivs-realtime list-participant-events \
  --stage-arn arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh \
  --session-id st-a1b2c3d4e5f6g \
  --participant-id abCDEf12GHIj
```

输出：

```
{
  "events": [
    {
      "eventTime": "2023-04-26T20:36:28+00:00",
      "name": "LEFT",
      "participantId": "abCDEf12GHIj"
    },
    {
      "eventTime": "2023-04-26T20:36:28+00:00",
      "name": "PUBLISH_STOPPED",
      "participantId": "abCDEf12GHIj"
    },
    {
      "eventTime": "2023-04-26T20:30:34+00:00",
      "name": "JOINED",
      "participantId": "abCDEf12GHIj"
    },
    {
      "eventTime": "2023-04-26T20:30:34+00:00",
```

```
        "name": "PUBLISH_STARTED",
        "participantId": "abCDEf12GHIj"
      }
    ]
  }
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListParticipantEvents](#)中的。

## list-participants

以下代码示例演示了如何使用 list-participants。

### AWS CLI

#### 获取舞台参与者名单

以下list-participants示例列出了指定阶段 ARN ( Amazon 资源名称 ) 的指定会话 ID 的所有参与者。

```
aws ivs-realtime list-participants \
  --stage-arn arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh \
  --session-id st-a1b2c3d4e5f6g
```

输出：

```
{
  "participants": [
    {
      "firstJoinTime": "2023-04-26T20:30:34+00:00",
      "participantId": "abCDEf12GHIj"
      "published": true,
      "state": "DISCONNECTED",
      "userId": ""
    }
  ]
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListParticipants](#)中的。

## list-stage-sessions

以下代码示例演示了如何使用 `list-stage-sessions`。

### AWS CLI

获取舞台会议清单

以下`list-stage-sessions`示例列出了指定阶段 ARN ( Amazon 资源名称 ) 的所有会话。

```
aws ivs-realtime list-stage-sessions \
  --stage-arn arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh
```

输出：

```
{
  "stageSessions": [
    {
      "endTime": "2023-04-26T20:36:29+00:00",
      "sessionId": "st-a1b2c3d4e5f6g",
      "startTime": "2023-04-26T20:30:29.602000+00:00"
    }
  ]
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStageSessions](#)中的。

## list-stages

以下代码示例演示了如何使用 `list-stages`。

### AWS CLI

获取有关所有阶段的摘要信息

以下`list-stages`示例列出了您的 AWS 账户在处理 API 请求的 AWS 区域中的所有阶段。

```
aws ivs-realtime list-stages
```

输出：

```
{
  "stages": [
    {
      "activeSessionId": "st-a1b2c3d4e5f6g",
      "arn": "arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh",
      "name": "stage1",
      "tags": {}
    },
    {
      "activeSessionId": "st-a123bcd456efg",
      "arn": "arn:aws:ivs:us-west-2:123456789012:stage/abcd1234ABCD",
      "name": "stage2",
      "tags": {}
    },
    {
      "activeSessionId": "st-abcDEF1234ghi",
      "arn": "arn:aws:ivs:us-west-2:123456789012:stage/ABCD1234efgh",
      "name": "stage3",
      "tags": {}
    }
  ]
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStages](#)中的。

## list-storage-configurations

以下代码示例演示了如何使用 list-storage-configurations。

AWS CLI

### 列出合成存储配置

以下list-storage-configurations列出了在处理 API 请求的 AWS 区域中您的 AWS 账户的所有合成存储配置。

```
aws ivs-realtime list-storage-configurations
```

输出：

```
{
  "storageConfigurations": [
    {
      "arn": "arn:aws:ivs:ap-northeast-1:123456789012:storage-configuration/
abcdABCDefgh",
      "name": "test-sc-1",
      "s3": {
        "bucketName": "test-bucket-1-name"
      },
      "tags": {}
    },
    {
      "arn": "arn:aws:ivs:ap-northeast-1:123456789012:storage-configuration/
ABCefgEFGabc",
      "name": "test-sc-2",
      "s3": {
        "bucketName": "test-bucket-2-name"
      },
      "tags": {}
    }
  ]
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStorageConfigurations](#)中的。

## start-composition

以下代码示例演示了如何使用 start-composition。

### AWS CLI

示例 1：使用默认布局设置开始合成

以下start-composition示例启动指定舞台的合成，以流式传输到指定位置。



```
aws ivs-realtime start-composition \
  --stage-arn arn:aws:ivs:ap-northeast-1:123456789012:stage/defgABCDabcd \
  --destinations '[{"channel": {"channelArn": "arn:aws:ivs:ap-
northeast-1:123456789012:channel/abcABCdefDEg", \
  "encoderConfigurationArn": "arn:aws:ivs:ap-northeast-1:123456789012:encoder-
configuration/ABabCDcdEFef"}}, \
  {"s3":{"encoderConfigurationArns":["arn:aws:ivs:ap-
northeast-1:123456789012:encoder-configuration/ABabCDcdEFef"], \
  "storageConfigurationArn":"arn:aws:ivs:ap-northeast-1:123456789012:storage-
configuration/FefABabCDcdE"}}]'
```

输出：

```
{
  "composition": {
    "arn": "arn:aws:ivs:ap-northeast-1:123456789012:composition/abcdABCDefgh",
    "destinations": [
      {
        "configuration": {
          "channel": {
            "channelArn": "arn:aws:ivs:ap-
northeast-1:123456789012:channel/abcABCdefDEg",
            "encoderConfigurationArn": "arn:aws:ivs:ap-
northeast-1:123456789012:encoder-configuration/ABabCDcdEFef"
          },
          "name": ""
        },
        "id": "AabBCcdDEefF",
        "state": "STARTING"
      },
      {
        "configuration": {
          "name": "",
          "s3": {
            "encoderConfigurationArns": [
              "arn:aws:ivs:arn:aws:ivs:ap-
northeast-1:123456789012:encoder-configuration/ABabCDcdEFef"
            ],
            "recordingConfiguration": {
              "format": "HLS"
            },
            "storageConfigurationArn": "arn:arn:aws:ivs:ap-
northeast-1:123456789012:storage-configuration/FefABabCDcdE"
          }
        }
      }
    ]
  }
}
```

```

        }
      },
      "detail": {
        "s3": {
          "recordingPrefix": "aBcDeFgHhGfE/AbCdEfGhHgFe/GHFabcgefABC/
composite"
        }
      },
      "id": "GHFabcgefABC",
      "state": "STARTING"
    }
  ],
  "layout": {
    "grid": {
      "featuredParticipantAttribute": ""
      "gridGap": 2,
      "omitStoppedVideo": false,
      "videoAspectRatio": "VIDEO",
      "videoFillMode": ""
    }
  },
  "stageArn": "arn:aws:ivs:ap-northeast-1:123456789012:stage/defgABCDabcd",
  "startTime": "2023-10-16T23:24:00+00:00",
  "state": "STARTING",
  "tags": {}
}
}

```

有关更多信息，请参阅 Amazon 互动视频服务用户指南中的[复合录制（实时流媒体）](#)。

## 示例 2：使用 PiP 布局开始合成

以下 start-composition 示例开始使用画中画布局将指定舞台流式传输到指定位置的合成。

```

aws ivs-realtime start-composition \
  --stage-arn arn:aws:ivs:ap-northeast-1:123456789012:stage/defgABCDabcd \
  --destinations '[{"channel": {"channelArn": "arn:aws:ivs:ap-
northeast-1:123456789012:channel/abcABCdefDEg"}, \
  "encoderConfigurationArn": "arn:aws:ivs:ap-northeast-1:123456789012:encoder-
configuration/ABabCDcdEFef"}]', \
  {"s3":{"encoderConfigurationArns":["arn:aws:ivs:ap-
northeast-1:123456789012:encoder-configuration/ABabCDcdEFef"], \
  "storageConfigurationArn":"arn:aws:ivs:ap-northeast-1:123456789012:storage-
configuration/FefABabCDcdE"}}]' \

```

```
--layout pip='{featuredParticipantAttribute="abcdefg"}'
```

输出：

```
{
  "composition": {
    "arn": "arn:aws:ivs:ap-northeast-1:123456789012:composition/wxyzWXYZpqrs",
    "destinations": [
      {
        "configuration": {
          "channel": {
            "channelArn": "arn:aws:ivs:ap-northeast-1:123456789012:channel/abcABCdefDEg",
            "encoderConfigurationArn": "arn:aws:ivs:ap-northeast-1:123456789012:encoder-configuration/ABabCDcdEFef"
          },
          "name": ""
        },
        "id": "AabBCcdDEefF",
        "state": "STARTING"
      },
      {
        "configuration": {
          "name": "",
          "s3": {
            "encoderConfigurationArns": [
              "arn:aws:ivs:arn:aws:ivs:ap-northeast-1:123456789012:encoder-configuration/ABabCDcdEFef"
            ],
            "recordingConfiguration": {
              "format": "HLS"
            },
            "storageConfigurationArn": "arn:arn:aws:ivs:ap-northeast-1:123456789012:storage-configuration/FefABabCDcdE"
          }
        },
        "detail": {
          "s3": {
            "recordingPrefix": "aBcDeFgHhGfE/AbCdEfGhHgFe/GHFabcgefABC/"
          }
        },
        "id": "GHFabcgefABC",
        "composite": ""
      }
    ]
  }
}
```

```
        "state": "STARTING"
      }
    ],
    "layout": {
      "pip": {
        "featuredParticipantAttribute": "abcdefg",
        "gridGap": 0,
        "omitStoppedVideo": false,
        "pipBehavior": "STATIC",
        "pipOffset": 0,
        "pipParticipantAttribute": "",
        "pipPosition": "BOTTOM_RIGHT",
        "videoFillMode": "COVER"
      }
    },
    "stageArn": "arn:aws:ivs:ap-northeast-1:123456789012:stage/defgABCDabcd",
    "startTime": "2023-10-16T23:24:00+00:00",
    "state": "STARTING",
    "tags": {}
  }
}
```

有关更多信息，请参阅 Amazon 互动视频服务用户指南中的[复合录制（实时流媒体）](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[StartComposition](#)中的。

## stop-composition

以下代码示例演示了如何使用 stop-composition。

### AWS CLI

#### 停止合成

以下内容 stop-composition 停止由给定 ARN ( Amazon 资源名称 ) 指定的合成。

```
aws ivs-realtime stop-composition \
  --arn "arn:aws:ivs:ap-northeast-1:123456789012:composition/abcdABCDefgh"
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopComposition](#)中的。

## update-stage

以下代码示例演示了如何使用 update-stage。

### AWS CLI

#### 更新舞台的配置

以下update-stage示例更新指定阶段 ARN 的阶段以更新阶段名称。

```
aws ivs-realtime update-stage \  
  --arn arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh \  
  --name stage1a
```

输出：

```
{  
  "stage": {  
    "arn": "arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh",  
    "name": "stage1a"  
  }  
}
```

有关更多信息，请参阅《亚马逊互动视频服务用户指南》中的[在 Amazon IVS 直播中启用多台主机](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateStage](#)中的。

## 使用 Amazon Kendra 的示例 AWS CLI

以下代码示例向您展示了如何在 Amazon Kendra 中 AWS Command Line Interface 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### create-data-source

以下代码示例演示了如何使用 create-data-source。

## AWS CLI

### 创建 Amazon Kendra 数据源连接器

以下内容 create-data-source 创建和配置 Amazon Kendra 数据源连接器。您可以使用 describe-data-source 查看数据源连接器的状态，如果状态显示数据源连接器“失败”，则可以读取任何错误消息，以完成创建。

```
aws kendra create-data-source \  
  --name "example data source 1" \  
  --description "Example data source 1 for example index 1 contains the first set  
of example documents" \  
  --tags '{"Key": "test resources", "Value": "kendra"}, {"Key": "test resources",  
"Value": "aws"}' \  
  --role-arn "arn:aws:iam::my-account-id:role/  
KendraRoleForS3TemplateConfigDataSource" \  
  --index-id exampleindex1 \  
  --language-code "es" \  
  --schedule "0 0 18 ? * TUE,MON,WED,THU,FRI,SAT *" \  
  --configuration '{"TemplateConfiguration": {"Template": file://  
s3schemaconfig.json}}' \  
  --type "TEMPLATE" \  
  --custom-document-enrichment-configuration '{"PostExtractionHookConfiguration":  
{"LambdaArn": "arn:aws:iam::my-account-id:function/my-function-ocr-docs",  
"S3Bucket": "s3://my-s3-bucket/scanned-image-text-example-docs"}, "RoleArn":  
"arn:aws:iam:my-account-id:role/KendraRoleForCDE"}' \  
  --vpc-configuration '{"SecurityGroupIds": ["sg-1234567890abcdef0"], "SubnetIds":  
["subnet-1c234", "subnet-2b134"]}'
```

输出：

```
{
```

```
"Id": "exempldatasource1"
}
```

有关更多信息，请参阅《亚马逊 Kendra [开发者指南](#)》中的 [Amazon Kendra 索引和数据源连接器入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDataSource](#)中的。

## create-index

以下代码示例演示了如何使用 create-index。

### AWS CLI

#### 创建 Amazon Kendra 索引

以下内容create-index创建和配置 Amazon Kendra 索引。您可以使用describe-index查看索引的状态，如果状态显示索引为“失败”，则可以读取任何错误消息，以便完全创建。

```
aws kendra create-index \
  --name "example index 1" \
  --description "Example index 1 contains the first set of example documents" \
  --tags '{"Key": "test resources", "Value": "kendra"}, {"Key": "test resources", "Value": "aws"}' \
  --role-arn "arn:aws:iam::my-account-id:role/KendraRoleForExampleIndex" \
  --edition "DEVELOPER_EDITION" \
  --server-side-encryption-configuration '{"KmsKeyId": "my-kms-key-id"}' \
  --user-context-policy "USER_TOKEN" \
  --user-token-configurations '{"JsonTokenTypeConfiguration": {"GroupAttributeField": "groupNameField", "UserNameAttributeField": "userNameField"}}'
```

输出：

```
{
  "Id": index1
}
```

有关更多信息，请参阅《亚马逊 Kendra [开发者指南](#)》中的 [Amazon Kendra 索引和数据源连接器入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateIndex](#)中的。

## describe-data-source

以下代码示例演示了如何使用 `describe-data-source`。

### AWS CLI

获取有关 Amazon Kendra 数据源连接器的信息

以下内容 `describe-data-source` 获取有关 Amazon Kendra 数据源连接器的信息。您可以查看数据源连接器的配置，如果状态显示数据源连接器“失败”，则可以阅读任何错误消息，无法完全创建。

```
aws kendra describe-data-source \  
  --id exampledatasource1 \  
  --index-id exampleindex1
```

输出：

```
{  
  "Configuration": {  
    "TemplateConfiguration": {  
      "Template": {  
        "connectionConfiguration": {  
          "repositoryEndpointMetadata": {  
            "BucketName": "my-bucket"  
          }  
        },  
        "repositoryConfigurations": {  
          "document": {  
            "fieldMappings": [  
              {  
                "indexFieldName": "_document_title",  
                "indexFieldType": "STRING",  
                "dataSourceFieldName": "title"  
              },  
              {  
                "indexFieldName": "_last_updated_at",  
                "indexFieldType": "DATE",  
                "dataSourceFieldName": "modified_date"  
              }  
            ]  
          }  
        }  
      }  
    }  
  },  
}
```



```
        "additionalProperties": {
          "inclusionPatterns": [
            "*.txt",
            "*.doc",
            "*.docx"
          ],
          "exclusionPatterns": [
            "*.json"
          ],
          "inclusionPrefixes": [
            "PublicExampleDocsFolder"
          ],
          "exclusionPrefixes": [
            "PrivateDocsFolder/private"
          ],
          "aclConfigurationFilePath": "ExampleDocsFolder/AclConfig.json",
          "metadataFilesPrefix": "metadata"
        },
        "syncMode": "FULL_CRAWL",
        "type": "S3",
        "version": "1.0.0"
      }
    },
    "CreatedAt": "2024-02-25T13:30:10+00:00",
    "CustomDocumentEnrichmentConfiguration": {
      "PostExtractionHookConfiguration": {
        "LambdaArn": "arn:aws:iam::my-account-id:function/my-function-ocr-docs",
        "S3Bucket": "s3://my-s3-bucket/scanned-image-text-example-docs/function"
      },
      "RoleArn": "arn:aws:iam:my-account-id:role/KendraRoleForCDE"
    },
    "Description": "Example data source 1 for example index 1 contains the first set
of example documents",
    "Id": "exampledatasource1",
    "IndexId": "exampleindex1",
    "LanguageCode": "en",
    "Name": "example data source 1",
    "RoleArn": "arn:aws:iam::my-account-id:role/
KendraRoleForS3TemplateConfigDataSource",
    "Schedule": "0 0 18 ? * TUE,MON,WED,THU,FRI,SAT *",
    "Status": "ACTIVE",
    "Type": "TEMPLATE",
    "UpdatedAt": "1709163615,"
```

```
"VpcConfiguration": {
  "SecurityGroupIds": ["sg-1234567890abcdef0"],
  "SubnetIds": ["subnet-1c234", "subnet-2b134"]
}
```

有关更多信息，请参阅《[亚马逊 Kendra 开发者指南](#)》中的 [Amazon Kendra 索引和数据源连接器入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDataSource](#) 中的。

## describe-index

以下代码示例演示了如何使用 describe-index。

### AWS CLI

获取有关亚马逊 Kendra 索引的信息

以下内容 describe-index 获取有关亚马逊 Kendra 索引的信息。您可以查看索引的配置，如果状态显示索引为“失败”，则可以阅读任何错误消息以完成创建。

```
aws kendra describe-index \
  --id exampleindex1
```

输出：

```
{
  "CapacityUnits": {
    "QueryCapacityUnits": 0,
    "StorageCapacityUnits": 0
  },
  "CreatedAt": "2024-02-25T12:30:10+00:00",
  "Description": "Example index 1 contains the first set of example documents",
  "DocumentMetadataConfigurations": [
    {
      "Name": "_document_title",
      "Relevance": {
        "Importance": 8
      },
      "Search": {
        "Displayable": true,
        "Facetable": false,
```

```
        "Searchable": true,
        "Sortable": false
    },
    "Type": "STRING_VALUE"
},
{
    "Name": "_document_body",
    "Relevance": {
        "Importance": 5
    },
    "Search": {
        "Displayable": true,
        "Facetable": false,
        "Searchable": true,
        "Sortable": false
    },
    "Type": "STRING_VALUE"
},
{
    "Name": "_last_updated_at",
    "Relevance": {
        "Importance": 6,
        "Duration": "2628000s",
        "Freshness": true
    },
    "Search": {
        "Displayable": true,
        "Facetable": false,
        "Searchable": true,
        "Sortable": true
    },
    "Type": "DATE_VALUE"
},
{
    "Name": "department_custom_field",
    "Relevance": {
        "Importance": 7,
        "ValueImportanceMap": {
            "Human Resources" : 4,
            "Marketing and Sales" : 2,
            "Research and innvoation" : 3,
            "Admin" : 1
        }
    }
},
```

```
    "Search": {
      "Displayable": true,
      "Facetable": true,
      "Searchable": true,
      "Sortable": true
    },
    "Type": "STRING_VALUE"
  }
],
"Edition": "DEVELOPER_EDITION",
"Id": "index1",
"IndexStatistics": {
  "FaqStatistics": {
    "IndexedQuestionAnswersCount": 10
  },
  "TextDocumentStatistics": {
    "IndexedTextBytes": 1073741824,
    "IndexedTextDocumentsCount": 1200
  }
},
"Name": "example index 1",
"RoleArn": "arn:aws:iam::my-account-id:role/KendraRoleForExampleIndex",
"ServerSideEncryptionConfiguration": {
  "KmsKeyId": "my-kms-key-id"
},
"Status": "ACTIVE",
"UpdatedAt": 1709163615,
"UserContextPolicy": "USER_TOKEN",
"UserTokenConfigurations": [
  {
    "JsonTokenTypeConfiguration": {
      "GroupAttributeField": "groupNameField",
      "UserNameAttributeField": "userNameField"
    }
  }
]
}
```

有关更多信息，请参阅《[亚马逊 Kendra 开发者指南](#)》中的 [Amazon Kendra 索引和数据源连接器入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeIndex](#) 中的。

## update-data-source

以下代码示例演示了如何使用 update-data-source。

### AWS CLI

#### 更新 Amazon Kendra 数据源连接器

以下内容 update-data-source 更新了 Amazon Kendra 数据源连接器的配置。如果操作成功，服务要么不发回任何输出，要么返回 HTTP 状态码 200，要么返回 AWS CLI 返回代码 0。您可以使用 describe-data-source 查看数据源连接器的配置和状态。

```
aws kendra update-data-source \
  --id exampledatasource1 \
  --index-id exampleindex1 \
  --name "new name for example data source 1" \
  --description "new description for example data source 1" \
  --role-arn arn:aws:iam::my-account-id:role/KendraNewRoleForExampleDataSource \
  --configuration '{"TemplateConfiguration": {"Template": file://
s3schemanewconfig.json}}' \
  --custom-document-enrichment-configuration '{"PostExtractionHookConfiguration":
{"LambdaArn": "arn:aws:iam::my-account-id:function/my-function-ocr-docs",
"S3Bucket": "s3://my-s3-bucket/scanned-image-text-example-docs"}, "RoleArn":
"arn:aws:iam:my-account-id:role/KendraNewRoleForCDE"}' \
  --language-code "es" \
  --schedule "0 0 18 ? * MON,WED,FRI *" \
  --vpc-configuration '{"SecurityGroupIds": ["sg-1234567890abcdef0"], "SubnetIds":
["subnet-1c234", "subnet-2b134"]}'
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊 Kendra [开发者指南](#)》中的 [Amazon Kendra 索引和数据源连接器入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateDataSource](#) 中的。

## update-index

以下代码示例演示了如何使用 update-index。

### AWS CLI

#### 更新亚马逊 Kendra 索引

以下内容update-index更新了 Amazon Kendra 索引的配置。如果操作成功，服务要么不返回任何输出，要么返回 HTTP 状态码 200，要么返回 AWS CLI 返回代码 0。您可以使用describe-index查看索引的配置和状态。

```
aws kendra update-index \  
  --id enterpriseindex1 \  
  --name "new name for Enterprise Edition index 1" \  
  --description "new description for Enterprise Edition index 1" \  
  --role-arn arn:aws:iam::my-account-id:role/KendraNewRoleForEnterpriseIndex \  
  --capacity-units '{"QueryCapacityUnits": 2, "StorageCapacityUnits": 1}' \  
  --document-metadata-configuration-updates '{"Name": "_document_title",  
"Relevance": {"Importance": 6}}, {"Name": "_last_updated_at", "Relevance":  
{"Importance": 8}}' \  
  --user-context-policy "USER_TOKEN" \  
  --user-token-configurations '{"JsonTokenTypeConfiguration":  
{"GroupAttributeField": "groupNameField", "UserNameAttributeField":  
"userNameField"}}'
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊 Kendra [开发者指南](#)》中的 [Amazon Kendra 索引和数据源连接器入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateIndex](#)中的。

## 使用的 Kinesis 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Kinesis 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### add-tags-to-stream

以下代码示例演示了如何使用 `add-tags-to-stream`。

#### AWS CLI

向数据流添加标签

以下 `add-tags-to-stream` 示例将带有键 `samplekey` 和值的标签分配 `example` 给指定的流。

```
aws kinesis add-tags-to-stream \  
  --stream-name samplestream \  
  --tags samplekey=example
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Kinesis Data Streams 开发者指南中的为直播[添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AddTagsToStream](#) 中的。

### create-stream

以下代码示例演示了如何使用 `create-stream`。

#### AWS CLI

创建数据流

以下 `create-stream` 示例创建一个名为 `samplestream` 的数据流，其中包含 3 个分片。

```
aws kinesis create-stream \  
  --stream-name samplestream \  
  --shard-count 3
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Kinesis Data Streams 开发人员指南》中的[创建流](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateStream](#) 中的。

## decrease-stream-retention-period

以下代码示例演示了如何使用 decrease-stream-retention-period。

### AWS CLI

#### 缩短数据流保留期

以下 decrease-stream-retention-period 示例将名为 samplestream 的流的保留期 ( 将数据记录添加到流中后可以访问的时间长度 ) 缩短到 48 小时。

```
aws kinesis decrease-stream-retention-period \  
  --stream-name samplestream \  
  --retention-period-hours 48
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Kinesis Data Streams 开发者指南中的更改数据[保留期](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DecreaseStreamRetentionPeriod](#) 中的。

## delete-stream

以下代码示例演示了如何使用 delete-stream。

### AWS CLI

#### 删除数据流

以下 delete-stream 示例将删除指定的数据流。

```
aws kinesis delete-stream \  
  --stream-name samplestream
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Kinesis Data Streams 开发人员指南》中的[删除流](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteStream](#) 中的。

## deregister-stream-consumer

以下代码示例演示了如何使用 deregister-stream-consumer。



## AWS CLI

### 注销数据流使用者的注册

以下deregister-stream-consumer示例从指定数据流中取消注册指定使用者。

```
aws kinesis deregister-stream-consumer \  
  --stream-arn arn:aws:kinesis:us-west-2:123456789012:stream/samplestream \  
  --consumer-name KinesisConsumerApplication
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 Kinesis Data Streams [开发者指南中的使用 Kinesis Data Streams API 开发具有增强扇出功能的消费者](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterStreamConsumer](#)中的。

## describe-limits

以下代码示例演示了如何使用 describe-limits。

### AWS CLI

#### 描述分片限制

以下describe-limits示例显示了当前 AWS 账户的分片限制和使用情况。

```
aws kinesis describe-limits
```

输出：

```
{  
  "ShardLimit": 500,  
  "OpenShardCount": 29  
}
```

有关更多信息，请参阅 Amazon Kinesis Data Streams 开发者指南中的[重新分片流](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLimits](#)中的。

## describe-stream-consumer

以下代码示例演示了如何使用 describe-stream-consumer。

## AWS CLI

### 描述数据流使用者

以下describe-stream-consumer示例返回在指定数据流中注册的指定使用者的描述。

```
aws kinesis describe-stream-consumer \  
  --stream-arn arn:aws:kinesis:us-west-2:012345678912:stream/samplestream \  
  --consumer-name KinesisConsumerApplication
```

输出：

```
{  
  "ConsumerDescription": {  
    "ConsumerName": "KinesisConsumerApplication",  
    "ConsumerARN": "arn:aws:kinesis:us-west-2:123456789012:stream/samplestream/  
consumer/KinesisConsumerApplication:1572383852",  
    "ConsumerStatus": "ACTIVE",  
    "ConsumerCreationTimestamp": 1572383852.0,  
    "StreamARN": "arn:aws:kinesis:us-west-2:123456789012:stream/samplestream"  
  }  
}
```

有关更多信息，请参阅亚马逊 Kinesis [Data Streams 开发者指南中的从亚马逊 Kinesis Data Streams 读取数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStreamConsumer](#)中的。

## describe-stream-summary

以下代码示例演示了如何使用 describe-stream-summary。

## AWS CLI

### 描述数据流摘要

以下describe-stream-summary示例提供了指定数据流的摘要描述（不包括分片列表）。

```
aws kinesis describe-stream-summary \  
  --stream-name samplestream
```

输出：

```
{
  "StreamDescriptionSummary": {
    "StreamName": "samplestream",
    "StreamARN": "arn:aws:kinesis:us-west-2:123456789012:stream/samplestream",
    "StreamStatus": "ACTIVE",
    "RetentionPeriodHours": 48,
    "StreamCreationTimestamp": 1572297168.0,
    "EnhancedMonitoring": [
      {
        "ShardLevelMetrics": []
      }
    ],
    "EncryptionType": "NONE",
    "OpenShardCount": 3,
    "ConsumerCount": 0
  }
}
```

有关更多信息，请参阅《Amazon Kinesis Data Streams 开发人员指南》中的[创建和管理流](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStreamSummary](#)中的。

## describe-stream

以下代码示例演示了如何使用 describe-stream。

### AWS CLI

#### 描述数据流

以下 describe-stream 示例将返回指定数据流的详细信息。

```
aws kinesis describe-stream \
  --stream-name samplestream
```

输出：

```
{
  "StreamDescription": {
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
```

```
        "StartingHashKey": "0",
        "EndingHashKey": "113427455640312821154458202477256070484"
    },
    "SequenceNumberRange": {
        "StartingSequenceNumber":
"49600871682957036442365024926191073437251060580128653314"
    }
},
{
    "ShardId": "shardId-000000000001",
    "HashKeyRange": {
        "StartingHashKey": "113427455640312821154458202477256070485",
        "EndingHashKey": "226854911280625642308916404954512140969"
    },
    "SequenceNumberRange": {
        "StartingSequenceNumber":
"49600871682979337187563555549332609155523708941634633746"
    }
},
{
    "ShardId": "shardId-000000000002",
    "HashKeyRange": {
        "StartingHashKey": "226854911280625642308916404954512140970",
        "EndingHashKey": "340282366920938463463374607431768211455"
    },
    "SequenceNumberRange": {
        "StartingSequenceNumber":
"49600871683001637932762086172474144873796357303140614178"
    }
}
],
"StreamARN": "arn:aws:kinesis:us-west-2:123456789012:stream/samplestream",
"StreamName": "samplestream",
"StreamStatus": "ACTIVE",
"RetentionPeriodHours": 24,
"EnhancedMonitoring": [
    {
        "ShardLevelMetrics": []
    }
],
"EncryptionType": "NONE",
"KeyId": null,
"StreamCreationTimestamp": 1572297168.0
}
```

```
}
```

有关更多信息，请参阅《Amazon Kinesis Data Streams 开发人员指南》中的[创建和管理流](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStream](#)中的。

## disable-enhanced-monitoring

以下代码示例演示了如何使用 disable-enhanced-monitoring。

### AWS CLI

#### 禁用分片级指标的增强监控

以下disable-enhanced-monitoring示例禁用分片级指标的增强型 Kinesis 数据流监控。

```
aws kinesis disable-enhanced-monitoring \  
  --stream-name samplestream --shard-level-metrics ALL
```

输出：

```
{  
  "StreamName": "samplestream",  
  "CurrentShardLevelMetrics": [  
    "IncomingBytes",  
    "OutgoingRecords",  
    "IteratorAgeMilliseconds",  
    "IncomingRecords",  
    "ReadProvisionedThroughputExceeded",  
    "WriteProvisionedThroughputExceeded",  
    "OutgoingBytes"  
  ],  
  "DesiredShardLevelMetrics": []  
}
```

有关更多信息，请参阅《[亚马逊 Kinesis 数据流开发者指南](#)》中的“[监控 Amazon Kinesis 数据流中的流](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableEnhancedMonitoring](#)中的。

## enable-enhanced-monitoring

以下代码示例演示了如何使用 `enable-enhanced-monitoring`。

### AWS CLI

启用对分片级指标的增强监控

以下 `enable-enhanced-monitoring` 示例为分片级指标启用增强的 Kinesis 数据流监控。

```
aws kinesis enable-enhanced-monitoring \  
  --stream-name samplestream \  
  --shard-level-metrics ALL
```

输出：

```
{  
  "StreamName": "samplestream",  
  "CurrentShardLevelMetrics": [],  
  "DesiredShardLevelMetrics": [  
    "IncomingBytes",  
    "OutgoingRecords",  
    "IteratorAgeMilliseconds",  
    "IncomingRecords",  
    "ReadProvisionedThroughputExceeded",  
    "WriteProvisionedThroughputExceeded",  
    "OutgoingBytes"  
  ]  
}
```

有关更多信息，请参阅 [《亚马逊 Kinesis 数据流开发者指南》](#) 中的“[监控 Amazon Kinesis 数据流中的流](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [EnableEnhancedMonitoring](#) 中的。

## get-records

以下代码示例演示了如何使用 `get-records`。

### AWS CLI

获取分片中的记录

以下 `get-records` 示例将使用指定的分片迭代器从 Kinesis 数据流的分片中获取数据记录。

```
aws kinesis get-records \  
  --shard-iterator AAAAAAAAAAF7/0mWD7IuHj1yGv/  
TKuNgx2ukD5xipCY4cy4gU96orWwZwcSXh3K9tAmGYe0ZyLZrvzze0FVf9iN99hUPw/w/  
b0YWYeefNvnf1DYt5XpDJghLKr3DzgzknTmMymDP3R+3wRKeuEw6/kdxY2yKJH0veaiekaVc4N2VwK/  
GvaGP2Hh9Fg7N++q0Adg6fIDQPt4p8RpavDbk+A4sL9SWGE1
```

输出：

```
{  
  "Records": [],  
  "MillisBehindLatest": 80742000  
}
```

有关更多信息，请参阅亚马逊 Kinesis Data Streams 开发者指南中的使用 Kinesis Data Streams [API 和 Java 开发工具包开发消费者](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRecords](#)中的。

## get-shard-iterator

以下代码示例演示了如何使用 `get-shard-iterator`。

### AWS CLI

#### 获取分片迭代器

以下`get-shard-iterator`示例使用`AT_SEQUENCE_NUMBER`分片迭代器类型并生成分片迭代器，从指定序列号所表示的位置精确开始读取数据记录。

```
aws kinesis get-shard-iterator \  
  --stream-name samplestream \  
  --shard-id shardId-000000000001 \  
  --shard-iterator-type LATEST
```

输出：

```
{  
  "ShardIterator": "AAAAAAAAAAFEvJjIYI+3jw/4aqqH9FifJ+n48XWTh/  
IFIsbILP6o5eDueD39NXNBfpZ10WL5K6ADXk8w+5H+Qhd9cFA9k268CPXCz/kebq1TGYI7Vy
```

```
+1UkA9BuN3xvATxMBGxRY3zYK05gqgvaIRn9408SqeEqwhigwZxNWxID3Ej7YYYcxQi8Q/fIrCjGAY/  
n2r5Z9G864YpWDFn9upNNQAR/ii0Wks"  
}
```

有关更多信息，请参阅亚马逊 Kinesis Data Streams 开发者指南中的使用 Kinesis Data Streams [API 和 Java 开发工具包开发消费者](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetShardIterator](#)中的。

## increase-stream-retention-period

以下代码示例演示了如何使用 `increase-stream-retention-period`。

### AWS CLI

延长数据流保留期

以下`increase-stream-retention-period`示例将指定流的保留期（数据记录添加到流后可以访问的时间长度）延长到 168 小时。

```
aws kinesis increase-stream-retention-period \  
  --stream-name samplestream \  
  --retention-period-hours 168
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Kinesis Data Streams 开发者指南中的更改数据[保留期](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[IncreaseStreamRetentionPeriod](#)中的。

## list-shards

以下代码示例演示了如何使用 `list-shards`。

### AWS CLI

列出数据流中的分片

以下`list-shards`示例列出了指定流中的所有分片，该分片以 ID 紧随指定的`exclusive-start-shard-id`分片开头。shardId-000000000000

```
aws kinesis list-shards \  
  --stream-name samplestream \  
  --exclusive-start-shard-id shardId-000000000000
```



```
--stream-name samplestream \  
--exclusive-start-shard-id shardId-000000000000
```

输出：

```
{  
  "Shards": [  
    {  
      "ShardId": "shardId-000000000001",  
      "HashKeyRange": {  
        "StartingHashKey": "113427455640312821154458202477256070485",  
        "EndingHashKey": "226854911280625642308916404954512140969"  
      },  
      "SequenceNumberRange": {  
        "StartingSequenceNumber":  
"49600871682979337187563555549332609155523708941634633746"  
      }  
    },  
    {  
      "ShardId": "shardId-000000000002",  
      "HashKeyRange": {  
        "StartingHashKey": "226854911280625642308916404954512140970",  
        "EndingHashKey": "340282366920938463463374607431768211455"  
      },  
      "SequenceNumberRange": {  
        "StartingSequenceNumber":  
"49600871683001637932762086172474144873796357303140614178"  
      }  
    }  
  ]  
}
```

有关更多信息，请参阅《亚马逊 Kinesis Data Streams 开发者指南》中的[列出分片](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListShards](#)中的。

## list-streams

以下代码示例演示了如何使用 list-streams。

### AWS CLI

#### 列出数据流

以下 `list-streams` 示例列出了当前账户和区域中的所有活动数据流。

```
aws kinesis list-streams
```

输出：

```
{
  "StreamNames": [
    "samplestream",
    "samplestream1"
  ]
}
```

有关更多信息，请参阅《Amazon Kinesis Data Streams 开发人员指南》中的[列出流](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStreams](#)中的。

## list-tags-for-stream

以下代码示例演示了如何使用 `list-tags-for-stream`。

AWS CLI

列出数据流的标签

以下`list-tags-for-stream`示例列出了附加到指定数据流的标签。

```
aws kinesis list-tags-for-stream \
  --stream-name samplestream
```

输出：

```
{
  "Tags": [
    {
      "Key": "samplekey",
      "Value": "example"
    }
  ],
  "HasMoreTags": false
}
```

```
}
```

有关更多信息，请参阅 Amazon Kinesis Data Streams 开发者指南中的为直播[添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForStream](#)中的。

## merge-shards

以下代码示例演示了如何使用 merge-shards。

### AWS CLI

#### 合并碎片

以下merge-shards示例在指定的数据流中合并 ID 为 shardid-00000000000000 和 shardid-00000000000001 的两个相邻分片，并将它们合并为一个分片。

```
aws kinesis merge-shards \  
  --stream-name samplestream \  
  --shard-to-merge shardId-000000000000 \  
  --adjacent-shard-to-merge shardId-000000000001
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Kinesis Data Streams 开发者指南中的[合并两个分片](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[MergeShards](#)中的。

## put-record

以下代码示例演示了如何使用 put-record。

### AWS CLI

#### 将记录写入数据流

以下 put-record 示例使用指定的分区键将单个数据记录写入指定的数据流。

```
aws kinesis put-record \  
  --stream-name samplestream \  
  --data sampledatarecord \  
  --partition-key samplepartitionkey
```

```
--partition-key samplepartitionkey
```

输出：

```
{
  "ShardId": "shardId-000000000009",
  "SequenceNumber": "49600902273357540915989931256901506243878407835297513618",
  "EncryptionType": "KMS"
}
```

有关更多信息，请参阅 [《亚马逊 Kinesis Data Streams 开发者指南》](#) 中的使用 [亚马逊 Kinesis Data Streams API](#) 和 [Java 开发工具包开发生产者](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutRecord](#) 中的。

## put-records

以下代码示例演示了如何使用 put-records。

### AWS CLI

将多条记录写入数据流

以下 put-records 示例使用指定的分区键写入数据记录，在一次调用中使用不同的分区键写入另一条数据记录。

```
aws kinesis put-records \  
  --stream-name samplestream \  
  --records Data=blob1,PartitionKey=partitionkey1  
  Data=blob2,PartitionKey=partitionkey2
```

输出：

```
{
  "FailedRecordCount": 0,
  "Records": [
    {
      "SequenceNumber":
"49600883331171471519674795588238531498465399900093808706",
      "ShardId": "shardId-000000000004"
    },
    {
```

```

        "SequenceNumber":
          "49600902273357540915989931256902715169698037101720764562",
        "ShardId": "shardId-000000000009"
      }
    ],
    "EncryptionType": "KMS"
  }

```

有关更多信息，请参阅 [《亚马逊 Kinesis Data Streams 开发者指南》](#) 中的使用亚马逊 Kinesis Data Streams API 和 Java 开发工具包开发生产者。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutRecords](#) 中的。

## register-stream-consumer

以下代码示例演示了如何使用 register-stream-consumer。

### AWS CLI

#### 注册数据流使用者

以下 register-stream-consumer 示例使用指定的数据流注册了一个名为 KinesisConsumerApplication 的使用者。

```

aws kinesis register-stream-consumer \
  --stream-arn arn:aws:kinesis:us-west-2:012345678912:stream/samplestream \
  --consumer-name KinesisConsumerApplication

```

输出：

```

{
  "Consumer": {
    "ConsumerName": "KinesisConsumerApplication",
    "ConsumerARN": "arn:aws:kinesis:us-west-2:123456789012:stream/samplestream/
consumer/KinesisConsumerApplication:1572383852",
    "ConsumerStatus": "CREATING",
    "ConsumerCreationTimestamp": 1572383852.0
  }
}

```

有关更多信息，请参阅亚马逊 Kinesis Data Streams [开发者指南中的使用 Kinesis Data Streams API 开发具有增强扇出功能的消费者](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterStreamConsumer](#)中的。

## remove-tags-from-stream

以下代码示例演示了如何使用 `remove-tags-from-stream`。

### AWS CLI

从数据流中移除标签

以下`remove-tags-from-stream`示例从指定的数据流中移除具有指定密钥的标签。

```
aws kinesis remove-tags-from-stream \  
  --stream-name samplestream \  
  --tag-keys samplekey
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Kinesis Data Streams 开发者指南中的为直播[添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveTagsFromStream](#)中的。

## split-shard

以下代码示例演示了如何使用 `split-shard`。

### AWS CLI

拆分碎片

以下`split-shard`示例使用新的起始哈希键 10 将指定的分片拆分为两个新分片。

```
aws kinesis split-shard \  
  --stream-name samplestream \  
  --shard-to-split shardId-000000000000 \  
  --new-starting-hash-key 10
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Kinesis Data Streams 开发者指南中的[拆分分片](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SplitShard](#)中的。

## start-stream-encryption

以下代码示例演示了如何使用 start-stream-encryption。

### AWS CLI

#### 启用数据流加密

以下start-stream-encryption示例使用指定的 KMS 密 AWS 钥为指定流启用服务器端加密。

```
aws kinesys start-stream-encryption \  
  --encryption-type KMS \  
  --key-id arn:aws:kms:us-west-2:012345678912:key/a3c4a7cd-728b-45dd-  
b334-4d3eb496e452 \  
  --stream-name samplestream
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 Kinesis [Data Streams 开发者指南中的亚马逊 Kinesis Data Streams 中的数据保护](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartStreamEncryption](#)中的。

## stop-stream-encryption

以下代码示例演示了如何使用 stop-stream-encryption。

### AWS CLI

#### 禁用数据流加密

以下stop-stream-encryption示例使用指定的 AWS KMS 密钥禁用指定流的服务器端加密。

```
aws kinesys start-stream-encryption \  
  --encryption-type KMS \  
  --key-id arn:aws:kms:us-west-2:012345678912:key/a3c4a7cd-728b-45dd-  
b334-4d3eb496e452 \  
  --stream-name samplestream
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 Kinesis [Data Streams 开发者指南中的亚马逊 Kinesis Data Streams 中的数据保护](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopStreamEncryption](#)中的。

## update-shard-count

以下代码示例演示了如何使用 update-shard-count。

### AWS CLI

#### 更新数据流中的分片数

以下update-shard-count示例将指定数据流的分片数更新为 6。此示例使用统一缩放，创建大小相等的分片。

```
aws kinesis update-shard-count \  
  --stream-name samplestream \  
  --scaling-type UNIFORM_SCALING \  
  --target-shard-count 6
```

输出：

```
{  
  "StreamName": "samplestream",  
  "CurrentShardCount": 3,  
  "TargetShardCount": 6  
}
```

有关更多信息，请参阅 Amazon Kinesis Data Streams 开发者指南中的[重新分片流](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateShardCount](#)中的。

## AWS KMS 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS KMS。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。



场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## cancel-key-deletion

以下代码示例演示了如何使用 cancel-key-deletion。

AWS CLI

取消按计划删除客户托管的 KMS 密钥

以下cancel-key-deletion示例取消了对客户托管的 KMS 密钥的计划删除。

```
aws kms cancel-key-deletion \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

输出：

```
{  
  "KeyId": "arn:aws:kms:us-  
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"  
}
```

cancel-key-deletion命令成功后，定时删除即被取消。但是，KMS 密钥的密钥状态为Disabled，因此您不能在加密操作中使用 KMS 密钥。要恢复其功能，请使用enable-key命令。

有关更多信息，请参阅《[密钥管理服务开发者指南](#)》中的[计划和取消AWS 密钥删除](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelKeyDeletion](#)中的。

## connect-custom-key-store

以下代码示例演示了如何使用 connect-custom-key-store。

## AWS CLI

### 连接自定义密钥库

以下 `connect-custom-key-store` 示例重新连接指定的自定义密钥库。您可以使用这样的命令首次连接自定义密钥存储库或重新连接已断开连接的密钥库。

您可以使用此命令连接 AWS CloudHSM 密钥存储库或外部密钥存储库。

```
aws kms connect-custom-key-store \  
  --custom-key-store-id cks-1234567890abcdef0
```

此命令不返回任何输出。要验证命令是否有效，请使用 `describe-custom-key-stores` 命令。

有关连接 AWS CloudHSM 密钥库的信息，[请参阅《密钥管理服务开发者指南》中的连接和断开 AWS CloudHSM 密钥存储库的连接](#)。AWS

有关连接外部密钥存储的信息，[请参阅《密钥管理服务开发人员指南》中的连接和断开外部AWS 密钥存储库的连接](#)。

- 有关 API 的详细信息，[请参阅AWS CLI 命令参考ConnectCustomKeyStore](#)中的。

## create-alias

以下代码示例演示了如何使用 `create-alias`。

### AWS CLI

#### 为 KMS 密钥创建别名

以下 `create-alias` 命令为由密钥 ID `1234abcd-12ab-34cd-56ef-1234567890ab` 标识的 KMS 密钥创建名为 `example-alias` 的别名。

别名名称必须以 `alias/` 开头。不要使用以开头的别名 `alias/aws`；这些别名是保留给使用的 AWS。

```
aws kms create-alias \  
  --alias-name alias/example-alias \  
  --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

此命令不返回任何输出。要查看新别名，请使用 `list-aliases` 命令。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[使用别名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAlias](#)中的。

## create-custom-key-store

以下代码示例演示了如何使用 create-custom-key-store。

### AWS CLI

#### 示例 1：创建 AWS CloudHSM 密钥存储库

以下create-custom-key-store示例使用所需参数创建由 AWS CloudHSM 集群支持的 C AWS CloudHSM 密钥存储库。您也可以添加custom-key-store-type``parameter with the default value: ``AWS\_CLOUDHSM.

要在 AWS CLI 中为trust-anchor-certificate命令指定文件输入，需要file://使用前缀。

```
aws kms create-custom-key-store \  
  --custom-key-store-name ExampleCloudHSMKeyStore \  
  --cloud-hsm-cluster-id cluster-1a23b4cdefg \  
  --key-store-password kmsPswd \  
  --trust-anchor-certificate file://customerCA.crt
```

输出：

```
{  
  "CustomKeyId": cks-1234567890abcdef0  
}
```

有关更多信息，请参阅《[密钥管理服务开发者 AWS 指南](#)》中的[AWS 创建 CloudHSM 密钥存储](#)。

#### 示例 2：创建具有公共端点连接的外部密钥存储库

以下create-custom-key-store示例创建了通过互联网与 AWS KMS 通信的外部密钥存储库 (XKS)。

在此示例中，XksProxyUriPath使用的可选前缀为example-prefix。

注意：如果您使用的是 AWS CLI 版本 1.0，请在指定带有 HTTP 或 HTTPS 值的参数（例如 XksProxyUriEndpoint 参数）之前运行以下命令。

```
aws configure set cli_follow_urlparam false
```

否则，AWS CLI 版本 1.0 会将参数值替换为在该 URI 地址中找到的内容。

```
aws kms create-custom-key-store \  
  --custom-key-store-name ExamplePublicEndpointXKS \  
  --custom-key-store-type EXTERNAL_KEY_STORE \  
  --xks-proxy-connectivity PUBLIC_ENDPOINT \  
  --xks-proxy-uri-endpoint "https://myproxy.xks.example.com" \  
  --xks-proxy-uri-path "/example-prefix/kms/xks/v1" \  
  --xks-proxy-authentication-credential "AccessKeyId=ABCDE12345670EXAMPLE,  
RawSecretAccessKey=DXjSUawne12fr6SKC7G25CNxTyWKE5PF9XX6H/u9pSo="
```

输出：

```
{  
  "CustomKeyId": cks-2234567890abcdef0  
}
```

有关更多信息，请参阅 [《密钥管理服务开发人员指南》中的创建外部AWS 密钥存储](#)。

示例 3：创建具有 VPC 终端节点服务连接的外部密钥存储库

以下 create-custom-key-store 示例创建了使用 Amazon VPC 终端节点服务与 KMS 通信的外部密钥存储 (X AWS KS)。

注意：如果您使用的是 AWS CLI 版本 1.0，请在指定带有 HTTP 或 HTTPS 值的参数（例如 XksProxyUriEndpoint 参数）之前运行以下命令。

```
aws configure set cli_follow_urlparam false
```

否则，AWS CLI 版本 1.0 会将参数值替换为在该 URI 地址中找到的内容。

```
aws kms create-custom-key-store \  
  --custom-key-store-name ExampleVPCEndpointXKS \  
  --custom-key-store-type EXTERNAL_KEY_STORE \  
  --xks-proxy-connectivity VPC_ENDPOINT_SERVICE \  
  --xks-proxy-uri-endpoint "https://myproxy-private.xks.example.com" \  
  --xks-proxy-uri-path "/kms/xks/v1" \  
  --xks-proxy-vpc-endpoint-service-name "com.amazonaws.vpce.us-east-1.vpce-svc-  
example1" \  
  --xks-proxy-authentication-credential "AccessKeyId=ABCDE12345670EXAMPLE,  
RawSecretAccessKey=DXjSUawne12fr6SKC7G25CNxTyWKE5PF9XX6H/u9pSo="
```

```
--xks-proxy-authentication-credential "AccessKeyId=ABCDE12345670EXAMPLE,  
RawSecretAccessKey=DXjSUawne12fr6SKC7G25CNxTyWKE5PF9XX6H/u9pSo="
```

输出：

```
{  
  "CustomKeyId": cks-3234567890abcdef0  
}
```

有关更多信息，请参阅《[密钥管理服务开发人员指南](#)》中的[创建外部AWS 密钥存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCustomKeyStore](#)中的。

## create-grant

以下代码示例演示了如何使用 create-grant。

### AWS CLI

#### 创建授权

以下 create-grant 示例将创建授权，以允许 exampleUser 用户对 1234abcd-12ab-34cd-56ef-1234567890ab KMS 密钥示例使用 decrypt 命令。停用主体是 adminRole 角色。该授权使用 EncryptionContextSubset 授权约束，以便仅在 decrypt 请求中的加密上下文包含 "Department": "IT" 键值对时才允许此权限。

```
aws kms create-grant \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --grantee-principal arn:aws:iam::123456789012:user/exampleUser \  
  --operations Decrypt \  
  --constraints EncryptionContextSubset={Department=IT} \  
  --retiring-principal arn:aws:iam::123456789012:role/adminRole
```

输出：

```
{  
  "GrantId": "1a2b3c4d2f5e69f440bae30eaec9570bb1fb7358824f9ddfa1aa5a0dab1a59b2",  
  "GrantToken": "<grant token here>"  
}
```

要查看有关授权的详细信息，请使用 list-grants 命令。

有关更多信息，请参阅《密AWS 钥管理服务开发人员指南》中的 [AWS KMS 中的授权](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateGrant](#)中的。

## create-key

以下代码示例演示了如何使用 create-key。

### AWS CLI

示例 1：在 KMS 中创建客户托管的 KM AWS S 密钥

以下 create-key 示例创建对称加密 KMS 密钥。

要创建基本 KMS 密钥（对称加密密钥），您无需指定任何参数。这些参数的默认值会创建对称加密密钥。

由于此命令未指定密钥策略，因此，KMS 密钥将获得以编程方式创建的 KMS 密钥的[默认密钥策略](#)。要查看密钥策略，请使用 get-key-policy 命令。要更改密钥策略，请使用 put-key-policy 命令。

```
aws kms create-key
```

create-key 命令会返回密钥元数据，包括新 KMS 密钥的密钥 ID 和 ARN。在其他 KMS 操作中，您可以使用这些值来识别 AWS KMS 密钥。输出不包括标签。要查看 KMS 密钥的标签，请使用 list-resource-tags command。

输出：

```
{
  "KeyMetadata": {
    "AWSAccountId": "111122223333",
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "CreationDate": "2017-07-05T14:04:55-07:00",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "Description": "",
    "Enabled": true,
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "KeyState": "Enabled",
```

```
    "KeyUsage": "ENCRYPT_DECRYPT",
    "MultiRegion": false,
    "Origin": "AWS_KMS"
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ]
  }
}
```

**注意：** `create-key` 命令不允许您指定别名。要为新 KMS 密钥创建别名，请使用 `create-alias` 命令。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[创建密钥](#)。

**示例 2：** 创建用于加密和解密的非对称 RSA KMS 密钥

以下 `create-key` 示例创建了一个 KMS 密钥，其中包含用于加密和解密的非对称 RSA 密钥对。

```
aws kms create-key \
  --key-spec RSA_4096 \
  --key-usage ENCRYPT_DECRYPT
```

**输出：**

```
{
  "KeyMetadata": {
    "Arn": "arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "CreationDate": "2021-04-05T14:04:55-07:00",
    "CustomerMasterKeySpec": "RSA_4096",
    "Description": "",
    "Enabled": true,
    "EncryptionAlgorithms": [
      "RSAES_OAEP_SHA_1",
      "RSAES_OAEP_SHA_256"
    ],
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "RSA_4096",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "MultiRegion": false,
```

```
    "Origin": "AWS_KMS"  
  }  
}
```

有关更多信息，请参阅 [《密钥管理服务开发人员指南》](#) 中的 [AWS KMS 中的非对称AWS密钥](#)。

### 示例 3：创建用于签名和验证的非对称椭圆曲线 KMS 密钥

创建用于签名和验证的非对称 KMS 密钥，其中包含非对称椭圆曲线（ECC）密钥对。尽管 SIGN\_VERIFY 是 ECC KMS 密钥的唯一有效值，但 --key-usage 参数也是必需的。

```
aws kms create-key \  
  --key-spec ECC_NIST_P521 \  
  --key-usage SIGN_VERIFY
```

输出：

```
{  
  "KeyMetadata": {  
    "Arn": "arn:aws:kms:us-  
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
    "AWSAccountId": "111122223333",  
    "CreationDate": "2019-12-02T07:48:55-07:00",  
    "CustomerMasterKeySpec": "ECC_NIST_P521",  
    "Description": "",  
    "Enabled": true,  
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",  
    "KeyManager": "CUSTOMER",  
    "KeySpec": "ECC_NIST_P521",  
    "KeyState": "Enabled",  
    "KeyUsage": "SIGN_VERIFY",  
    "MultiRegion": false,  
    "Origin": "AWS_KMS",  
    "SigningAlgorithms": [  
      "ECDSA_SHA_512"  
    ]  
  }  
}
```

有关更多信息，请参阅 [《密钥管理服务开发人员指南》](#) 中的 [AWS KMS 中的非对称AWS密钥](#)。

### 示例 4：创建 HMAC KMS 密钥



以下 `create-key` 示例将创建 384 位 HMAC KMS 密钥。尽管 `--key-usage` 参数的 `GENERATE_VERIFY_MAC` 值是 HMAC KMS 密钥的唯一有效值，但该值也是必需的。

```
aws kms create-key \  
  --key-spec HMAC_384 \  
  --key-usage GENERATE_VERIFY_MAC
```

输出：

```
{  
  "KeyMetadata": {  
    "Arn": "arn:aws:kms:us-  
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
    "AWSAccountId": "111122223333",  
    "CreationDate": "2022-04-05T14:04:55-07:00",  
    "CustomerMasterKeySpec": "HMAC_384",  
    "Description": "",  
    "Enabled": true,  
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",  
    "KeyManager": "CUSTOMER",  
    "KeySpec": "HMAC_384",  
    "KeyState": "Enabled",  
    "KeyUsage": "GENERATE_VERIFY_MAC",  
    "MacAlgorithms": [  
      "HMAC_SHA_384"  
    ],  
    "MultiRegion": false,  
    "Origin": "AWS_KMS"  
  }  
}
```

有关更多信息，请参阅 [《密钥管理服务开发人员指南》中的 AWS KMS 中的 HMAC AWS 密钥](#)。

#### 示例 4：创建多区域主 KMS 密钥

以下 `create-key` 示例创建多区域主对称加密密钥。由于所有参数的默认值都会创建对称加密密钥，因此，此 KMS 密钥只需要 `--multi-region` 参数。在 AWS CLI 中，要指示布尔参数为真，只需指定参数名称即可。

```
aws kms create-key \  
  --multi-region
```

输出：

```
{
  "KeyMetadata": {
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/mrk-1234abcd12ab34cd56ef12345678990ab",
    "AWSAccountId": "111122223333",
    "CreationDate": "2021-09-02T016:15:21-09:00",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "Description": "",
    "Enabled": true,
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ],
    "KeyId": "mrk-1234abcd12ab34cd56ef12345678990ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "MultiRegion": true,
    "MultiRegionConfiguration": {
      "MultiRegionKeyType": "PRIMARY",
      "PrimaryKey": {
        "Arn": "arn:aws:kms:us-west-2:111122223333:key/mrk-1234abcd12ab34cd56ef12345678990ab",
        "Region": "us-west-2"
      },
      "ReplicaKeys": []
    },
    "Origin": "AWS_KMS"
  }
}
```

有关更多信息，请参阅 [《密钥管理服务开发人员指南》](#) 中的 [AWS KMS 中的非对称 AWS 密钥](#)。

示例 5：为导入的密钥材料创建 KMS 密钥

以下 create-key 示例创建一个不带密钥材料的 KMS 密钥。操作完成后，您可以将自己的密钥材料导入 KMS 密钥。要创建此 KMS 密钥，请将 --origin 参数设置为 EXTERNAL。

```
aws kms create-key \
  --origin EXTERNAL
```

输出：

```
{
  "KeyMetadata": {
    "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "CreationDate": "2019-12-02T07:48:55-07:00",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "Description": "",
    "Enabled": false,
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ],
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "KeyState": "PendingImport",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "MultiRegion": false,
    "Origin": "EXTERNAL"
  }
}
```

有关更多信息，请参阅 [《密钥管理服务开发人员指南》](#) 中的在 AWS KMS 密AWS 钥中导入密钥材料。

示例 6：在 AWS CloudHSM 密钥存储库中创建 KMS 密钥

以下 create-key 示例在指定的 AWS CloudHSM 密钥存储库中创建一个 KMS 密钥。该操作在 KMS 中 AWS 创建 KMS 密钥及其元数据，并在与自定义密钥存储库关联的 AWS CloudHSM 集群中创建密钥材料。--custom-key-store-id 和 --origin 参数是必需的。

```
aws kms create-key \
  --origin AWS_CLOUDHSM \
  --custom-key-store-id cks-1234567890abcdef0
```

输出：

```
{
  "KeyMetadata": {
```

```

    "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "CloudHsmClusterId": "cluster-1a23b4cdefg",
    "CreationDate": "2019-12-02T07:48:55-07:00",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "CustomKeyStoreId": "cks-1234567890abcdef0",
    "Description": "",
    "Enabled": true,
    "EncryptionAlgorithms": [
        "SYMMETRIC_DEFAULT"
    ],
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "MultiRegion": false,
    "Origin": "AWS_CLOUDHSM"
}
}

```

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的 [AWS CloudHSM 密钥存储](#)。

示例 7：在外部密钥存储中创建 KMS 密钥

以下 `create-key` 示例在指定的外部密钥存储中创建一个 KMS 密钥。在此命令中需要使用 `--custom-key-store-id`、`--origin` 和 `--xks-key-id` 参数。

`--xks-key-id` 参数指定外部密钥管理器中现有对称加密密钥的 ID。此密钥用作 KMS 密钥的外部密钥材料。`--origin` 参数的值必须为 `EXTERNAL_KEY_STORE`。`custom-key-store-id` 参数必须识别连接到其外部密钥存储代理的外部密钥存储。

```

aws kms create-key \
  --origin EXTERNAL_KEY_STORE \
  --custom-key-store-id cks-9876543210fedcba9 \
  --xks-key-id bb8562717f809024

```

输出：

```
{
```

```
"KeyMetadata": {
  "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "AWSAccountId": "111122223333",
  "CreationDate": "2022-12-02T07:48:55-07:00",
  "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
  "CustomKeyStoreId": "cks-9876543210fedcba9",
  "Description": "",
  "Enabled": true,
  "EncryptionAlgorithms": [
    "SYMMETRIC_DEFAULT"
  ],
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "KeyManager": "CUSTOMER",
  "KeySpec": "SYMMETRIC_DEFAULT",
  "KeyState": "Enabled",
  "KeyUsage": "ENCRYPT_DECRYPT",
  "MultiRegion": false,
  "Origin": "EXTERNAL_KEY_STORE",
  "XksKeyConfiguration": {
    "Id": "bb8562717f809024"
  }
}
```

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[外部密钥存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateKey](#) 中的。

## decrypt

以下代码示例演示了如何使用 decrypt。

### AWS CLI

示例 1：使用对称 KMS 密钥解密加密消息 ( Linux 和 macOS )

以下 decrypt 命令示例演示了使用 CLI AWS 解密数据的推荐方法。此版本演示了如何使用对称 KMS 密钥解密数据。

在文件中提供加密文字。在 --ciphertext-blob 参数的值中，使用 fileb:// 前缀，它将指示 CLI 从二进制文件中读取数据。如果文件不在当前目录中，请键入文件的完整路径。有关从文件读取 AWS CLI 参数值的更多信息，请参阅《AWS 命令行界面用户指南》中的“从文件加载 AWS

CLI 参数 < <https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-parameters-file.html>>”和 AWS 命令行工具博客中的本地文件参数最佳实践 < <https://aws.amazon.com/blogs/developer/best-practices-for-local-file-parameters/>>。指定 KMS 密钥来解密密文。使用对称 KMS 密钥解密时不需要参数。--key-id AWS KMS 可以从密文中的元数据中获取用于加密数据的 KMS 密钥的密钥 ID。但是，指定您正在使用的 KMS 密钥始终是最佳实践。此做法可确保您使用预期的 KMS 密钥，并防止您意外使用不信任的 KMS 密钥解密加密文字。以文本值请求明文输出。--query 参数指示 CLI 仅从输出中获取 Plaintext 字段的值。--output 参数以 text.base64 解码格式返回明文输出并将其保存在文件中。以下示例将 Plaintext 参数的值传送 ( | ) 给 Base64 实用程序，该程序负责对其进行解码。然后，它将解码后的输出重定向 ( > ) 到 ExamplePlaintext 文件。

在运行此命令之前，请将示例密钥 ID 替换为 AWS 账户中的有效密钥 ID。

```
aws kms decrypt \  
  --ciphertext-blob fileb://ExampleEncryptedFile \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --output text \  
  --query Plaintext | base64 \  
  --decode > ExamplePlaintextFile
```

此命令不生成任何输出。decrypt 命令的输出经过 base64 解码并保存在文件中。

有关更多信息，请参阅《AWS Key Management Service API 参考》中的[解密](#)。

示例 2：使用对称 KMS 密钥解密加密消息 ( Windows 命令提示符 )

以下示例与前一个示例相同，不同之处在于，它使用 certutil 实用程序对明文数据进行 Base64 解码。此过程需要两个命令，如以下示例所示。

在运行此命令之前，请将示例密钥 ID 替换为 AWS 账户中的有效密钥 ID。

```
aws kms decrypt ^\  
  --ciphertext-blob fileb://ExampleEncryptedFile ^\  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab ^\  
  --output text ^\  
  --query Plaintext > ExamplePlaintextFile.base64
```

运行 certutil 命令。

```
certutil -decode ExamplePlaintextFile.base64 ExamplePlaintextFile
```

输出：

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

有关更多信息，请参阅《AWS Key Management Service API 参考》中的[解密](#)。

示例 3：使用非对称 KMS 密钥解密加密消息 (Linux 和 macOS)

以下 `decrypt` 命令示例演示如何解密在 RSA 非对称 KMS 密钥下加密的数据。

使用非对称 KMS 密钥时，需要 `encryption-algorithm` 参数，以指定用于加密明文的算法。

在运行此命令之前，请将示例密钥 ID 替换为 AWS 账户中的有效密钥 ID。

```
aws kms decrypt \
  --ciphertext-blob fileb://ExampleEncryptedFile \
  --key-id 0987dcb0-09fe-87dc-65ba-ab0987654321 \
  --encryption-algorithm RSAES_OAEP_SHA_256 \
  --output text \
  --query Plaintext | base64 \
  --decode > ExamplePlaintextFile
```

此命令不生成任何输出。`decrypt` 命令的输出经过 base64 解码并保存在文件中。

有关更多信息，请参阅《[密钥管理服务开发人员指南](#)》中的 [AWS KMS 中的非对称 AWS 密钥](#)。

- 有关 API 详细信息，请参阅《[AWS CLI 命令参考](#)》中的[解密](#)。

## delete-alias

以下代码示例演示了如何使用 `delete-alias`。

### AWS CLI

#### 删除 AWS KMS 别名

以下 `delete-alias` 示例将删除别名 `alias/example-alias`。别名名称必须以 `alias/` 开头。

```
aws kms delete-alias \
  --alias-name alias/example-alias
```

此命令不生成任何输出。要查找别名，请使用 `list-aliases` 命令。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[删除别名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAlias](#)中的。

## **delete-custom-key-store**

以下代码示例演示了如何使用 `delete-custom-key-store`。

### AWS CLI

#### 删除自定义密钥库

以下`delete-custom-key-store`示例删除了指定的自定义密钥库。

删除 AWS CloudHSM 密钥库不会对关联的 CloudHSM 集群产生任何影响。删除外部密钥存储不会影响关联的外部密钥存储代理、外部密钥管理器或外部密钥。

注意：在删除自定义密钥存储库之前，必须计划删除自定义密钥存储库中的所有 KMS 密钥，然后等待删除这些 KMS 密钥。然后，您必须断开自定义密钥存储的连接。有关在自定义密钥存储中查找 KMS 密钥的帮助，请参阅密钥管理服务开发者[AWS 指南中的AWS 删除 CloudHSM 密钥存储 \(API\)](#)。

```
delete-custom-key-store \  
  --custom-key-store-id cks-1234567890abcdef0
```

此命令不返回任何输出。要验证自定义密钥库是否已删除，请使用`describe-custom-key-stores`命令。

有关删除 AWS CloudHSM 密钥库的信息，请参阅《密钥管理服务开发者指南》中的[删除 AWS CloudHSM 密钥存储库](#)。AWS

有关删除外部密钥库的信息，请参阅《[密钥管理服务开发人员指南](#)》中的[删除外部AWS 密钥存储库](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCustomKeyStore](#)中的。

## **delete-imported-key-material**

以下代码示例演示了如何使用 `delete-imported-key-material`。



## AWS CLI

从 KMS 密钥中删除导入的密钥材料

以下`delete-imported-key-material`示例删除已导入 KMS 密钥的密钥材料。

```
aws kms delete-imported-key-material \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

此命令不生成任何输出。要验证密钥材料是否已删除，请使用`describe-key`命令查找`PendingImport`或的密钥状态`PendingDeletion`。

有关更多信息，请参阅《密钥管理服务开发人员指南》中的“删除导入的密钥importing-keys-delete-key材料 < <https://docs.aws.amazon.com/kms/latest/developerguide/-material.html>>”AWS。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteImportedKeyMaterial](#)中的。

## describe-custom-key-stores

以下代码示例演示了如何使用 `describe-custom-key-stores`。

## AWS CLI

示例 1：获取有关 AWS CloudHSM 密钥存储库的详细信息

以下`describe-custom-key-store`示例显示了有关指定 AWS CloudHSM 密钥库的详细信息。所有类型的自定义密钥库的命令都相同，但输出因密钥库类型以及外部密钥存储的连接选项而异。

默认情况下，此命令显示有关账户和区域中所有自定义密钥存储的信息。要显示有关特定自定义密钥库的信息，请使用`custom-key-store-name`或`custom-key-store-id`参数。

```
aws kms describe-custom-key-stores \  
  --custom-key-store-name ExampleCloudHSMKeyStore
```

此命令的输出包括有关 AWS CloudHSM 密钥库的有用详细信息，包括其连接状态 (`ConnectionState`)。如果连接状态为`FAILED`，则输出将包含一个描述问题的`ConnectionErrorCode`字段。

输出：

```
{
```

```

    "CustomKeyStores": [
      {
        "CloudHsmClusterId": "cluster-1a23b4cdefg",
        "ConnectionState": "CONNECTED",
        "CreationDate": "2022-04-05T14:04:55-07:00",
        "CustomKeyStoreId": "cks-1234567890abcdef0",
        "CustomKeyStoreName": "ExampleExternalKeyStore",
        "TrustAnchorCertificate": "<certificate appears here>"
      }
    ]
  }
}

```

有关更多信息，请参阅《[密钥管理服务开发者 AWS 指南](#)》中的[AWS 查看 CloudHSM 密钥存储区](#)。

示例 2：获取有关具有公共端点连接的外部密钥存储库的详细信息

以下describe-custom-key-store示例显示有关指定外部密钥存储的详细信息。所有类型的自定义密钥库的命令都相同，但输出因密钥库类型以及外部密钥存储的连接选项而异。

默认情况下，此命令显示有关账户和区域中所有自定义密钥存储的信息。要显示有关特定自定义密钥库的信息，请使用custom-key-store-name或custom-key-store-id参数。

```

aws kms describe-custom-key-stores \
  --custom-key-store-id cks-9876543210fedcba9

```

此命令的输出包括有关外部密钥库的有用详细信息，包括其连接状态 (ConnectionState)。如果连接状态为FAILED，则输出将包含一个描述问题的ConnectionErrorCode字段。

输出：

```

{
  "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-9876543210fedcba9",
      "CustomKeyStoreName": "ExampleXKS",
      "ConnectionState": "CONNECTED",
      "CreationDate": "2022-12-02T07:48:55-07:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE12345670EXAMPLE",

```

```

        "Connectivity": "PUBLIC_ENDPOINT",
        "UriEndpoint": "https://myproxy.xks.example.com",
        "UriPath": "/example-prefix/kms/xks/v1"
    }
}
]
}

```

有关更多信息，请参阅 [《密钥管理服务开发人员指南》中的查看外部AWS 密钥存储](#)。

示例 3：获取有关具有 VPC 终端节点服务连接的外部密钥存储的详细信息

以下describe-custom-key-store示例显示有关指定外部密钥存储的详细信息。所有类型的自定义密钥库的命令都相同，但输出因密钥库类型以及外部密钥存储的连接选项而异。

默认情况下，此命令显示有关账户和区域中所有自定义密钥存储的信息。要显示有关特定自定义密钥库的信息，请使用custom-key-store-name或custom-key-store-id参数。

```

aws kms describe-custom-key-stores \
  --custom-key-store-id cks-2234567890abcdef0

```

此命令的输出包括有关外部密钥库的有用详细信息，包括其连接状态 (ConnectionState)。如果连接状态为FAILED，则输出将包含一个描述问题的ConnectionErrorCode字段。

输出：

```

{
  "CustomKeyStores": [
    {
      "CustomKeyId": "cks-3234567890abcdef0",
      "CustomKeyName": "ExampleVPCExternalKeyStore",
      "ConnectionState": "CONNECTED",
      "CreationDate": "2022-12-22T07:48:55-07:00",
      "CustomKeyType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE12345670EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://myproxy-private.xks.example.com",
        "UriPath": "/kms/xks/v1",
        "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example1"
      }
    }
  ]
}

```

```
    }  
  ]  
}
```

有关更多信息，请参阅 [《密钥管理服务开发人员指南》中的查看外部AWS 密钥存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCustomKeyStores](#)中的。

## describe-key

以下代码示例演示了如何使用 describe-key。

### AWS CLI

示例 1：查找有关 KMS 密钥的详细信息

以下describe-key示例在示例账户和区域中获取有关 Amazon S3 AWS 托管密钥的详细信息。您可以使用此命令来查找有关 AWS 托管密钥和客户托管密钥的详细信息。

要指定 KMS 密钥，请使用 key-id 参数。此示例使用别名名称值，但您可以在此命令中使用密钥 ID、密钥 ARN、别名名称或别名 ARN。

```
aws kms describe-key \  
  --key-id alias/aws/s3
```

输出：

```
{  
  "KeyMetadata": {  
    "AWSAccountId": "846764612917",  
    "KeyId": "b8a9477d-836c-491f-857e-07937918959b",  
    "Arn": "arn:aws:kms:us-west-2:846764612917:key/  
b8a9477d-836c-491f-857e-07937918959b",  
    "CreationDate": 2017-06-30T21:44:32.140000+00:00,  
    "Enabled": true,  
    "Description": "Default KMS key that protects my S3 objects when no other  
key is defined",  
    "KeyUsage": "ENCRYPT_DECRYPT",  
    "KeyState": "Enabled",  
    "Origin": "AWS_KMS",  
    "KeyManager": "AWS",
```

```
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ]
  }
}
```

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[查看密钥](#)。

示例 2：获取有关 RSA 非对称 KMS 密钥的详细信息

以下 describe-key 示例获取有关用于签名和验证的非对称 RSA KMS 密钥的详细信息。

```
aws kms describe-key \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

输出：

```
{
  "KeyMetadata": {
    "AWSAccountId": "111122223333",
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "CreationDate": "2019-12-02T19:47:14.861000+00:00",
    "CustomerMasterKeySpec": "RSA_2048",
    "Enabled": false,
    "Description": "",
    "KeyState": "Disabled",
    "Origin": "AWS_KMS",
    "MultiRegion": false,
    "KeyManager": "CUSTOMER",
    "KeySpec": "RSA_2048",
    "KeyUsage": "SIGN_VERIFY",
    "SigningAlgorithms": [
      "RSASSA_PKCS1_V1_5_SHA_256",
      "RSASSA_PKCS1_V1_5_SHA_384",
      "RSASSA_PKCS1_V1_5_SHA_512",
      "RSASSA_PSS_SHA_256",
      "RSASSA_PSS_SHA_384",
      "RSASSA_PSS_SHA_512"
    ]
  }
}
```

```
}
```

### 示例 3：获取有关多区域副本密钥的详细信息

以下 `describe-key` 示例获取多区域副本密钥的元数据。此多区域密钥是对称加密密钥。任何多区域密钥的 `describe-key` 命令输出都会返回有关主密钥及其所有副本的信息。

```
aws kms describe-key \  
  --key-id arn:aws:kms:ap-northeast-1:111122223333:key/  
mrk-1234abcd12ab34cd56ef1234567890ab
```

输出：

```
{  
  "KeyMetadata": {  
    "MultiRegion": true,  
    "AWSAccountId": "111122223333",  
    "Arn": "arn:aws:kms:ap-northeast-1:111122223333:key/  
mrk-1234abcd12ab34cd56ef1234567890ab",  
    "CreationDate": "2021-06-28T21:09:16.114000+00:00",  
    "Description": "",  
    "Enabled": true,  
    "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",  
    "KeyManager": "CUSTOMER",  
    "KeyState": "Enabled",  
    "KeyUsage": "ENCRYPT_DECRYPT",  
    "Origin": "AWS_KMS",  
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",  
    "EncryptionAlgorithms": [  
      "SYMMETRIC_DEFAULT"  
    ],  
    "MultiRegionConfiguration": {  
      "MultiRegionKeyType": "PRIMARY",  
      "PrimaryKey": {  
        "Arn": "arn:aws:kms:us-west-2:111122223333:key/  
mrk-1234abcd12ab34cd56ef1234567890ab",  
        "Region": "us-west-2"  
      },  
      "ReplicaKeys": [  
        {  
          "Arn": "arn:aws:kms:eu-west-1:111122223333:key/  
mrk-1234abcd12ab34cd56ef1234567890ab",  
          "Region": "eu-west-1"  
        }  
      ]  
    }  
  }  
}
```

```

    },
    {
      "Arn": "arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
      "Region": "ap-northeast-1"
    },
    {
      "Arn": "arn:aws:kms:sa-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
      "Region": "sa-east-1"
    }
  ]
}
}
}

```

#### 示例 4：获取有关 HMAC KMS 密钥的详细信息

以下 `describe-key` 示例获取有关 HMAC KMS 密钥的详细信息。

```

aws kms describe-key \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab

```

输出：

```

{
  "KeyMetadata": {
    "AWSAccountId": "123456789012",
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "Arn": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "CreationDate": "2022-04-03T22:23:10.194000+00:00",
    "Enabled": true,
    "Description": "Test key",
    "KeyUsage": "GENERATE_VERIFY_MAC",
    "KeyState": "Enabled",
    "Origin": "AWS_KMS",
    "KeyManager": "CUSTOMER",
    "CustomerMasterKeySpec": "HMAC_256",
    "MacAlgorithms": [
      "HMAC_SHA_256"
    ],
    "MultiRegion": false
  }
}

```

```
}  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeKey](#)中的。

## disable-key-rotation

以下代码示例演示了如何使用 `disable-key-rotation`。

### AWS CLI

#### 禁用 KMS 密钥的自动轮换

以下`disable-key-rotation`示例禁用客户托管的 KMS 密钥的自动轮换。要重新启用自动旋转，请使用`enable-key-rotation`命令。

```
aws kms disable-key-rotation \  
  --key-id arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

此命令不生成任何输出。要验证是否已禁用 KMS 密钥的自动轮换，请使用`get-key-rotation-status`命令。

有关更多信息，请参阅《[密AWS 钥管理服务开发人员指南](#)》中的轮换密钥。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableKeyRotation](#)中的。

## disable-key

以下代码示例演示了如何使用 `disable-key`。

### AWS CLI

#### 暂时禁用 KMS 密钥

以下示例使用 `disable-key` 命令禁用客户托管的 KMS 密钥。要重新启用 KMS 密钥，请使用 `enable-key` 命令。

```
aws kms disable-key \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```



此命令不生成任何输出。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[启用和禁用密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableKey](#)中的。

## disconnect-custom-key-store

以下代码示例演示了如何使用 `disconnect-custom-key-store`。

### AWS CLI

断开自定义密钥存储的连接

以下`disconnect-custom-key-store`示例断开自定义密钥库与其 Clou AWS dHSM 集群的连接。您可以断开密钥存储的连接，以解决问题、更新其设置或阻止在加密操作中使用密钥库中的 KMS 密钥。

此命令适用于所有自定义密钥存储库，包括 AWS CloudHSM 密钥存储库和外部密钥存储库。

在运行此命令之前，请将示例自定义密钥存储 ID 替换为有效 ID。

```
$ aws kms disconnect-custom-key-store \  
  --custom-key-store-id cks-1234567890abcdef0
```

此命令不产生任何输出。请验证命令是否有效，请使用该`describe-custom-key-stores`命令。

有关断开 Cloud AWS HSM 密钥库连接的更多信息，请参阅《[密钥管理服务开发者指南](#)》中的[连接和断开 AWS CloudHSM](#) 密钥存储库的连接。AWS

有关断开外部密钥存储库连接的更多信息，请参阅《[密钥管理服务开发人员指南](#)》中的[连接和断开外部AWS 密钥存储库](#)的连接。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisconnectCustomKeyStore](#)中的。

## enable-key-rotation

以下代码示例演示了如何使用 `enable-key-rotation`。

### AWS CLI

启用 KMS 密钥的自动轮换

以下 `enable-key-rotation` 示例启用客户托管的 KMS 密钥的自动轮换，轮换周期为 180 天。KMS 密钥将自该命令完成之日起一年（大约 365 天）以及此后每年轮换。

`--key-id` 参数识别 KMS 密钥。此示例使用密钥 ARN 值，但您可以使用密钥 ID 或 KMS 密钥的 ARN。该 `--rotation-period-in-days` 参数指定每个轮换日期之间的天数。请指定一个介于 90 到 2560 天之间的值。如果未指定任何值，则默认值为 365 天。

```
aws kms enable-key-rotation \  
  --key-id arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \  
  --rotation-period-in-days 180
```

此命令不生成任何输出。要验证 KMS 密钥是否已启用，请使用 `get-key-rotation-status` 命令。

有关更多信息，请参阅《[密AWS 钥管理服务开发人员指南](#)》中的轮换密钥。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [EnableKeyRotation](#) 中的。

## enable-key

以下代码示例演示了如何使用 `enable-key`。

### AWS CLI

#### 启用 KMS 密钥

以下 `enable-key` 示例启用客户托管密钥。您可以使用类似的命令来启用您通过 `disable-key` 命令暂时禁用的 KMS 密钥。您还可以使用它来启用已被禁用的 KMS 密钥，因为已计划删除该密钥且删除已取消。

要指定 KMS 密钥，请使用 `key-id` 参数。此示例使用密钥 ID 值，但您可以在此命令中使用密钥 ID 或密钥 ARN 值。

在运行此命令之前，将密钥 ID 示例替换为有效 ID。

```
aws kms enable-key \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

此命令不生成任何输出。要验证 KMS 密钥是否已启用，请使用 `describe-key` 命令。查看 `describe-key` 输出中 `KeyState` 和 `Enabled` 字段的值。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[启用和禁用密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[EnableKey](#)中的。

## encrypt

以下代码示例演示了如何使用 encrypt。

### AWS CLI

示例 1：在 Linux 或 macOS 上对文件内容进行加密

以下 encrypt 命令演示了使用 AWS CLI 加密数据的推荐方法。

```
aws kms encrypt \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --plaintext fileb://ExamplePlaintextFile \  
  --output text \  
  --query CiphertextBlob | base64 \  
  --decode > ExampleEncryptedFile
```

该命令可以执行以下几项操作：

使用 `--plaintext` 参数来指示要加密的数据。此参数值必须采用 Base64 编码。该 `plaintext` 参数的值必须采用 base64 编码，或者必须使用前缀 `fileb://`，它指示 CL AWS I 从文件中读取二进制数据。如果文件不在当前目录中，请键入文件的完整路径。例如：`fileb:///var/tmp/ExamplePlaintextFile` 或 `fileb://C:\Temp\ExamplePlaintextFile`。[有关从文件读取 AWS CLI 参数值的更多信息，请参阅《AWS 命令行界面用户指南》中的“从文件加载参数”和“AWS 命令行工具博客”上的“本地文件参数最佳实践”。](#)使用 `--output` 和 `--query` 参数控制命令的输出。这些参数从命令的输出中提取加密数据（称为密文）。[有关控制输出的更多信息，请参阅控制命令 AWS 命令行界面用户指南中的输出。](#)使用该 `base64` 实用程序将提取的输出解码为二进制数据。成功命令返回的密文是 base64 编码的文本。encrypt 必须先解码此文本，然后才能使用 AWS CLI 对其进行解密。将二进制密文保存到文件中。命令 (`> ExampleEncryptedFile`) 的最后部分将二进制密文保存到文件中，以便于解密。有关使用 AWS CLI 解密数据的命令示例，请参阅解密示例。

示例 2：使用 AWS CLI 在 Windows 上加密数据

此示例与前一个示例相同，不同之处在于，它使用 `certutil` 工具代替 `base64`。此过程需要两个命令，如以下示例所示。

```
aws kms encrypt \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --plaintext fileb://ExamplePlaintextFile \  
  --output text \  
  --query CiphertextBlob > C:\Temp\ExampleEncryptedFile.base64  
  
certutil -decode C:\Temp\ExampleEncryptedFile.base64 C:\Temp\ExampleEncryptedFile
```

### 示例 3：使用非对称 KMS 密钥进行加密

以下 `encrypt` 命令显示如何使用非对称 KMS 密钥加密明文。 `--encryption-algorithm` 参数是必需的。与所有 `encrypt` CLI 命令一样，该 `plaintext` 参数必须采用 base64 编码，或者必须使用 `fileb://` 前缀，它告诉 CL AWS I 从文件中读取二进制数据。

```
aws kms encrypt \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --encryption-algorithm RSAES_OAEP_SHA_256 \  
  --plaintext fileb://ExamplePlaintextFile \  
  --output text \  
  --query CiphertextBlob | base64 \  
  --decode > ExampleEncryptedFile
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [Encrypt](#)。

## generate-data-key-pair-without-plaintext

以下代码示例演示了如何使用 `generate-data-key-pair-without-plaintext`。

### AWS CLI

生成 ECC NIST P384 非对称数据密钥对

以下 `generate-data-key-pair-without-plaintext` 示例请求一个 ECC NIST P384 密钥对以外使用。AWS

该命令返回一个纯文本公钥和使用指定 KMS 密钥加密的私钥的副本。它不返回纯文本私钥。您可以安全地将加密的私钥与加密数据一起存储，并在需要时调用 AWS KMS 解密私钥。

要请求 ECC NIST P384 非对称数据密钥对，请使用值 `key-pair-spec` 为的参数。 `ECC_NIST_P384`

您指定的 KMS 密钥必须是对称加密 KMS 密钥，即KeySpec值为的 KMS 密钥。SYMMETRIC\_DEFAULT

注意：本示例输出中的值被截断，便于显示。

```
aws kms generate-data-key-pair-without-plaintext \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --key-pair-spec ECC_NIST_P384
```

输出：

```
{  
  "PrivateKeyCiphertextBlob": "AQIDAHi6LtupRpdK12aJTzkK6Fbh0tQkM1QJJH3PdtHvS/y  
+hAFFxmiD134doUDzMGmfCEtcAAAHaTCCB2UGCSqGSIB3DQEHbqCCB1...",  
  "PublicKey":  
  "MIIB0jANBgkqhkiG9w0BAQEFAA0CAY8AMIIBigKCAyEA3A3eGMyPrvSn7+Ld1JE1oUoQV5HpEuHAVbd0yND  
+NmYDH/mL10SIEuLrcdZ5hrMH4pk83r401...",  
  "KeyId": "arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
  "KeySpec": "ECC_NIST_P384"  
}
```

PublicKey和PrivateKeyCiphertextBlob以 base64 编码的格式返回。

有关更多信息，请参阅《[密钥管理服务开发人员指南](#)》中的[数据AWS 密钥对](#)。

• 有关 API 的详细信息，请参阅AWS CLI 命令参考[GenerateDataKeyPairWithoutPlaintext](#)中的。

## generate-data-key-pair

以下代码示例演示了如何使用 generate-data-key-pair。

### AWS CLI

生成 2048 位 RSA 非对称数据密钥对

以下generate-data-key-pair示例请求一个 2048 位 RSA 非对称数据密钥对，以便在之外使用。AWS该命令返回一个纯文本公钥和一个供立即使用和删除的纯文本私钥，以及使用指定 KMS 密钥加密的私钥的副本。您可以安全地将加密的私钥与加密数据一起存储。

要请求 2048 位 RSA 非对称数据密钥对，请使用值key-pair-spec为的参数。RSA\_2048

您指定的 KMS 密钥必须是对称加密 KMS 密钥，即KeySpec值为的 KMS 密钥。SYMMETRIC\_DEFAULT

注意：本示例输出中的值被截断，便于显示。

```
aws kms generate-data-key-pair \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --key-pair-spec RSA_2048
```

输出：

```
{  
  "PrivateKeyCiphertextBlob": "AQIDAHi6LtupRpdK12aJTzkK6Fbh0tQkM1QJJH3PdtHvS/y  
+hAFFxmiD134doUDzMGmfCEtcAAAHaTCCB2UGCSqGSIB3DQEHbqCCB1...",  
  "PrivateKeyPlaintext": "MIIG/  
QIBADANBgkqhkiG9w0BAQEFAASCbucwggbjAgEAAoIBgQDcDd4YzI  
+u9Kfv4t2UkTWhShBXkekS4cBVt07I0P42ZgMf+YvU5IgS4ut...",  
  "PublicKey":  
  "MIIB0jANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBigKCAyEA3A3eGMyPrvSn7+Ld1JE1oUoQV5HpEuHAVbd0yND  
+NmYDH/mL10SIEuLrzdZ5hrMH4pk83r40l...",  
  "KeyId": "arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
  "KeySpec": "RSA_2048"  
}
```

PublicKeyPrivateKeyPlaintext、和PrivateKeyCiphertextBlob以 base64 编码的格式返回。

有关更多信息，请参阅 [《密钥管理服务开发人员指南》中的数据AWS 密钥对](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GenerateDataKeyPair](#)中的。

## generate-data-key-without-plaintext

以下代码示例演示了如何使用 generate-data-key-without-plaintext。

### AWS CLI

生成不带明文密钥的 256 位对称数据密钥

以下 generate-data-key-without-plaintext 示例请求 256 位对称数据密钥的加密副本以供在 AWS 外部使用。准备好使用数据密钥时，可以调用 AWS KMS 对其进行解密。

要请求 256 位数据密钥，请使用值为 AES\_256 的 `key-spec` 参数。要请求 128 位数据密钥，请使用值为 AES\_128 的 `key-spec` 参数。对于所有其他数据密钥长度，请使用 `number-of-bytes` 参数。

您指定的 KMS 密钥必须是对称加密 KMS 密钥，即密钥规格值为 SYMMETRIC\_DEFAULT 的 KMS 密钥。

```
aws kms generate-data-key-without-plaintext \
  --key-id "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" \
  --key-spec AES_256
```

输出：

```
{
  "CiphertextBlob":
  "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIHvcNAQcGoG8wbQIBADBoBgkqhki
  "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

CiphertextBlob ( 加密数据密钥 ) 以 base64 编码的格式返回。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[数据密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GenerateDataKeyWithoutPlaintext](#) 中的。

## generate-data-key

以下代码示例演示了如何使用 `generate-data-key`。

### AWS CLI

示例 1：生成 256 位对称数据密钥

以下 `generate-data-key` 示例请求一个 256 位的对称数据密钥以供外部使用。AWS 该命令返回一个明文数据密钥以供立即使用和删除，以及以指定 KMS 密钥加密的该数据密钥的副本。您可以安全地将加密的数据密钥与加密的数据一起存储。

要请求 256 位数据密钥，请使用值为 AES\_256 的 `key-spec` 参数。要请求 128 位数据密钥，请使用值为 AES\_128 的 `key-spec` 参数。对于所有其他数据密钥长度，请使用 `number-of-bytes` 参数。

您指定的 KMS 密钥必须是对称加密 KMS 密钥，即密钥规格值为 SYMMETRIC\_DEFAULT 的 KMS 密钥。

```
aws kms generate-data-key \  
  --key-id alias/ExampleAlias \  
  --key-spec AES_256
```

输出：

```
{  
  "Plaintext": "VdzKNHGzUAzJeRBVY+uUmofUGGiDzyB3+i9fVkh3piw=",  
  "KeyId": "arn:aws:kms:us-  
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
  "CiphertextBlob":  
  "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZlIhvcNAQcGoG8wbQIBADBoBgkqhki3g0YdhV8MrkBQPeac0ReRVNDt9qlEAt+SHgIRF8P0H+7U="
```

Plaintext (明文数据密钥) 和 CiphertextBlob (加密数据密钥) 均以 base64 编码的格式返回。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的“数据密钥”<<https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#data-keys>>。

示例 2：生成 512 位对称数据密钥

以下 generate-data-key 示例请求用于加密和解密的 512 位对称数据密钥。该命令返回一个明文数据密钥以供立即使用和删除，以及以指定 KMS 密钥加密的该数据密钥的副本。您可以安全地将加密的数据密钥与加密的数据一起存储。

要请求 128 或 256 位以外的密钥长度，请使用 number-of-bytes 参数。为了请求 512 位数据密钥，以下示例使用值为 64 (字节) 的 number-of-bytes 参数。

您指定的 KMS 密钥必须是对称加密 KMS 密钥，即密钥规格值为 SYMMETRIC\_DEFAULT 的 KMS 密钥。

注意：本示例输出中的值被截断，便于显示。

```
aws kms generate-data-key \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --number-of-bytes 64
```



输出：

```
{
  "CiphertextBlob": "AQIBAHi6LtupRpdK12aJTzkkK6Fbh0tQkMlQJJH3PdtHvS/y+hAEnX/
QQNmMwDfg2korNMEc8AAACaDCCAmQGCSqGSiB3DQEHBqCCA1UwggJRAgEAMIICSgYJKoZ...",
  "Plaintext": "ty8Lr0Bk60F07M2Bwt6qbFdNB
+G00ZLtf5MSEb4a13R2UKWG0p06njAwy2n72VRm2m7z/
Pm9Wpbvttz6a4lSo9hgPvKhZ5y6RTm40ovEXiVfBveyX3DQxDzRSwbKDPk/...",
  "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Plaintext (明文数据密钥) 和 CiphertextBlob (加密数据密钥) 均以 base64 编码的格式返回。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的“数据密钥”<<https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#data-keys>>。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GenerateDataKey](#) 中的。

## generate-random

以下代码示例演示了如何使用 generate-random。

### AWS CLI

示例 1：生成 256 位随机字节字符串 (Linux 或 macOS)

以下 generate-random 示例生成 256 位 (32 字节)、以 base64 编码的随机字节字符串。该示例对字节字符串进行解码并将其保存在随机文件中。

运行此命令时，您必须使用 number-of-bytes 参数指定随机值的长度 (以字节为单位)。

在运行此命令时，无需指定 KMS 密钥。随机字节字符串与任何 KMS 密钥无关。

默认情况下，AWS KMS 会生成随机数。但是，如果您指定自定义密钥存储库 <<https://docs.aws.amazon.com/kms/latest/developerguide/custom-key-store-overview.html>>，则随机字节字符串将在与自定义密钥存储库关联的 CloudHSM 集群中生 AWS 成。

本示例使用以下参数和值：

它使用值为的必需 --number-of-bytes 参数 32 来请求一个 32 字节 (256 位) 的字符串。它使用值为的 --output 参数指示 CL AWS I 将输出作为文本返回，而不是 JSON。它使用从响应中提

取Plaintext属性的值。它使用重定向--query parameter向运算符 (>) 从响应中提取的字符串中提取出来。它使用重定向运算符 (>) 来保存解码后的字节字符串到文件。它使用重定向运算符 (>) text base64 ExampleRandom将二进制密文保存到文件中。

```
aws kms generate-random \  
  --number-of-bytes 32 \  
  --output text \  
  --query Plaintext | base64 --decode > ExampleRandom
```

此命令不生成任何输出。

有关更多信息，请参阅AWS 密钥管理服务 API 参考[GenerateRandom](#)中的。

示例 2：生成 256 位随机数 ( Windows 命令提示符 )

以下示例使用 generate-random 命令生成 256 位 ( 32 字节 )、以 base64 编码的随机字节字符串。该示例对字节字符串进行解码并将其保存在随机文件中。此示例与前面的示例相同，不同之处在于：它在将随机字节字符串保存到文件之前，使用 Windows 中的 certutil 实用工具对随机字节字符串进行 base64 解码。

首先，生成一个 base64 编码的随机字节字符串并将其保存在临时文件 ExampleRandom.base64 中。

```
aws kms generate-random \  
  --number-of-bytes 32 \  
  --output text \  
  --query Plaintext > ExampleRandom.base64
```

由于 generate-random 命令的输出保存在文件中，因此，此示例不生成任何输出。

现在，使用 certutil -decode 命令解码 ExampleRandom.base64 文件中以 base64 编码的字节字符串。然后，它将解码后的字节字符串保存在 ExampleRandom 文件中。

```
certutil -decode ExampleRandom.base64 ExampleRandom
```

输出：

```
Input Length = 18  
Output Length = 12  
CertUtil: -decode command completed successfully.
```

有关更多信息，请参阅AWS 密钥管理服务 API 参考[GenerateRandom](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GenerateRandom](#)中的。

## get-key-policy

以下代码示例演示了如何使用 get-key-policy。

### AWS CLI

将密钥策略从一个 KMS 密钥复制到另一个 KMS 密钥

以下 get-key-policy 示例从一个 KMS 密钥获取密钥策略并将其保存在文本文件中。然后，它使用文本文件作为策略输入替换其他 KMS 密钥的策略。

由于 put-key-policy 的 --policy 参数需要字符串，因此，您必须使用 --output text 选项将输出作为文本字符串（而不是 JSON）返回。

```
aws kms get-key-policy \  
  --policy-name default \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --query Policy \  
  --output text > policy.txt  
  
aws kms put-key-policy \  
  --policy-name default \  
  --key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \  
  --policy file://policy.txt
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS KMS API 参考[PutKeyPolicy](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetKeyPolicy](#)中的。

## get-key-rotation-status

以下代码示例演示了如何使用 get-key-rotation-status。

### AWS CLI

检索 KMS 密钥的轮换状态。

以下`get-key-rotation-status`示例返回有关指定 KMS 密钥轮换状态的信息，包括是否启用自动轮换、轮换周期和下一个计划轮换日期。您可以对客户托管的 KMS 密钥和 AWS 托管 KMS 密钥使用此命令。但是，所有 AWS 托管的 KMS 密钥每年都会自动轮换。

```
aws kms get-key-rotation-status \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

输出：

```
{  
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",  
  "KeyRotationEnabled": true,  
  "NextRotationDate": "2024-02-14T18:14:33.587000+00:00",  
  "RotationPeriodInDays": 365  
}
```

有关更多信息，请参阅《[密AWS 钥管理服务开发人员指南](#)》中的轮换密钥。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetKeyRotationStatus](#)中的。

## get-parameters-for-import

以下代码示例演示了如何使用 `get-parameters-for-import`。

### AWS CLI

获取将密钥材料导入 KMS 密钥所需的物品

以下`get-parameters-for-import`示例获取将密钥材料导入 KMS 密钥所需的公钥和导入令牌。使用该`import-key-material`命令时，请务必使用在同一`get-parameters-for-import`命令中返回的由公钥加密的导入令牌和密钥材料。此外，您在此命令中指定的包装算法必须是用于使用公钥加密密钥材料的算法。

要指定 KMS 密钥，请使用 `key-id` 参数。此示例使用密钥 ID，但您可以在此命令中使用密钥 ID 或密钥 ARN。

```
aws kms get-parameters-for-import \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --wrapping-algorithm RSAES_OAEP_SHA_256 \  
  --wrapping-key-spec RSA_2048
```

输出：

```
{
  "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "PublicKey": "<public key base64 encoded data>",
  "ImportToken": "<import token base64 encoded data>",
  "ParametersValidTo": 1593893322.32
}
```

有关更多信息，请参阅 [《密钥管理服务开发者指南》](#) 中的 [下载公AWS 钥和导入令牌](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetParametersForImport](#) 中的。

## get-public-key

以下代码示例演示了如何使用 get-public-key。

### AWS CLI

示例 1：下载非对称 KMS 密钥的公钥

以下 get-public-key 示例下载非对称 KMS 密钥的公钥。

除了返回公钥外，输出还包括在 AWS KMS 之外安全使用公钥所需的信息，包括密钥使用情况和支持的加密算法。

```
aws kms get-public-key \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

输出：

```
{
  "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "PublicKey": "jANBgkqhkiG9w0BAQEFAAOCAG8AMIICCGKCAgEA15epvg1/
QtJhxSi2g9SDEVg8QV/...",
  "CustomerMasterKeySpec": "RSA_4096",
  "KeyUsage": "ENCRYPT_DECRYPT",
  "EncryptionAlgorithms": [
    "RSAES_OAEP_SHA_1",
    "RSAES_OAEP_SHA_256"
  ]
}
```

```
}
```

有关在 KMS 中 AWS 使用非对称 KMS 密钥的更多信息，请参阅密钥管理服务 API 参考中的[使用对称和非对称密钥](#)。AWS

示例 2：将公钥转换为 DER 格式（Linux 和 macOS）

以下 `get-public-key` 示例下载非对称 KMS 密钥的公钥并将其保存在 DER 文件中。

当你在 AWS CLI 中使用该 `get-public-key` 命令时，它会返回一个经过 Base64 编码的 DER 编码的 X.509 公钥。此示例以文本形式获取 `PublicKey` 属性的值。它对 Base64 进行解码 `PublicKey` 并将其保存在文件中。 `public_key.der` 该 `output` 参数将输出以文本形式返回，而不是 JSON。该 `--query` 参数仅获取 `PublicKey` 属性，而不是在 AWS KMS 之外安全使用公钥所需的属性。

在运行此命令之前，请将示例密钥 ID 替换为 AWS 账户中的有效密钥 ID。

```
aws kms get-public-key \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --output text \  
  --query PublicKey | base64 --decode > public_key.der
```

此命令不生成任何输出。

有关在 KMS 中 AWS 使用非对称 KMS 密钥的更多信息，请参阅密钥管理服务 API 参考中的[使用对称和非对称密钥](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetPublicKey](#) 中的。

## import-key-material

以下代码示例演示了如何使用 `import-key-material`。

### AWS CLI

将密钥材料导入 KMS 密钥

以下 `import-key-material` 示例将密钥材料上传到创建时没有密钥材料的 KMS 密钥中。KMS 密钥的密钥状态必须为 `PendingImport`。

此命令使用您使用 `get-parameters-for-import` 命令返回的公钥加密的密钥材料。它还使用来自同一 `get-parameters-for-import` 命令的导入令牌。

该`expiration-model`参数表示密钥材料在`valid-to`参数指定的日期和时间自动过期。密钥材料到期后，AWS KMS 会删除密钥材料，KMS 密钥的密钥状态更改为`Pending import`，KMS 密钥将无法使用。要恢复 KMS 密钥，必须重新导入相同的密钥材料。要使用不同的密钥材料，必须创建新的 KMS 密钥。

在运行此命令之前，请将示例密钥 ID 替换为您 AWS 账户中的有效密钥 ID 或密钥 ARN。

```
aws kms import-key-material \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --encrypted-key-material fileb://EncryptedKeyMaterial.bin \
  --import-token fileb://ImportToken.bin \
  --expiration-model KEY_MATERIAL_EXPIRES \
  --valid-to 2021-09-21T19:00:00Z
```

此命令不生成任何输出。

有关导入密钥材料的更多信息，请参阅 [《密钥管理服务开发人员指南》中的导入AWS 密钥材料](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ImportKeyMaterial](#)中的。

## list-aliases

以下代码示例演示了如何使用 `list-aliases`。

### AWS CLI

示例 1：列出 AWS 账户和区域中的所有别名

以下示例使用`list-aliases`命令列出 AWS 账户默认区域中的所有别名。输出包括与 AWS 托管 KMS 密钥和客户托管 KMS 密钥关联的别名。

```
aws kms list-aliases
```

输出：

```
{
  "Aliases": [
    {
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/testKey",
      "AliasName": "alias/testKey",
      "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ]
}
```

```

    },
    {
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/FinanceDept",
      "AliasName": "alias/FinanceDept",
      "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"
    },
    {
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/dynamodb",
      "AliasName": "alias/aws/dynamodb",
      "TargetKeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
    },
    {
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/ebs",
      "AliasName": "alias/aws/ebs",
      "TargetKeyId": "0987ab65-43cd-21ef-09ab-87654321cdef"
    },
    ...
  ]
}

```

## 示例 2：列出特定 KMS 密钥的所有别名

以下示例使用 `list-aliases` 命令及其 `key-id` 参数列出与特定 KMS 密钥关联的所有别名。

每个别名都仅与一个 KMS 密钥关联，但一个 KMS 密钥可以有多个别名。此命令非常有用，因为 AWS KMS 控制台仅列出每个 KMS 密钥的一个别名。要查找 KMS 密钥的所有别名，您必须使用 `list-aliases` 命令。

此示例使用 KMS 密钥的密钥 ID 作为 `--key-id` 参数，但您可以在此命令中使用密钥 ID、密钥 ARN、别名名称或别名 ARN。

```
aws kms list-aliases --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

输出：

```

{
  "Aliases": [
    {
      "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/oregon-test-key",
      "AliasName": "alias/oregon-test-key"
    },
  ],
}

```



```
{
  "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/project121-test",
  "AliasName": "alias/project121-test"
}
]
```

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[使用别名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAliases](#)中的。

## list-grants

以下代码示例演示了如何使用 list-grants。

### AWS CLI

查看 AWS KMS 密钥的授权

以下list-grants示例显示了您的账户中针对亚马逊 DynamoDB 的指定 AWS 托管 KMS 密钥的所有授权。该授权允许 DynamoDB 在将 DynamoDB 表写入磁盘之前代表您使用 KMS 密钥对其进行加密。您可以使用这样的命令来查看 AWS 账户和区域中 AWS 托管 KMS 密钥和客户托管 KMS 密钥的授权。

此命令使用带有密钥 ID 的 key-id 参数来识别 KMS 密钥。您可以使用密钥 ID 或密钥 ARN 来识别 KMS 密钥。要获取 AWS 托管 KMS 密钥的密钥 ID 或密钥 ARN，请使用list-keys或list-aliases命令。

```
aws kms list-grants \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

输出显示，该授权向 Amazon DynamoDB 授予了使用 KMS 密钥进行加密操作的权限，并向其授予了查看有关 KMS 密钥的详细信息 ( DescribeKey ) 和停用授权 ( RetireGrant ) 的权限。EncryptionContextSubset 约束将这些权限限制为包含指定加密上下文对的请求。因此，授权中的权限仅对指定账户和 DynamoDB 表有效。

```
{
  "Grants": [
    {
      "Constraints": {
```

```

        "EncryptionContextSubset": {
            "aws:dynamodb:subscriberId": "123456789012",
            "aws:dynamodb:tableName": "Services"
        }
    },
    "IssuingAccount": "arn:aws:iam::123456789012:root",
    "Name": "8276b9a6-6cf0-46f1-b2f0-7993a7f8c89a",
    "Operations": [
        "Decrypt",
        "Encrypt",
        "GenerateDataKey",
        "ReEncryptFrom",
        "ReEncryptTo",
        "RetireGrant",
        "DescribeKey"
    ],
    "GrantId":
    "1667b97d27cf748cf05b487217dd4179526c949d14fb3903858e25193253fe59",
    "KeyId": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "RetiringPrincipal": "dynamodb.us-west-2.amazonaws.com",
    "GranteePrincipal": "dynamodb.us-west-2.amazonaws.com",
    "CreationDate": "2021-05-13T18:32:45.144000+00:00"
    }
]
}

```

有关更多信息，请参阅《密AWS 钥管理服务开发人员指南》中的 [AWS KMS 中的授权](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListGrants](#)中的。

## list-key-policies

以下代码示例演示了如何使用 list-key-policies。

### AWS CLI

获取 KMS 密钥的密钥策略名称

以下 list-key-policies 示例获取示例账户和区域中客户托管密钥的密钥策略名称。您可以使用此命令查找托管密钥和客户 AWS 托管密钥的密钥策略名称。

由于唯一有效的密钥策略名称是 default，因此，此命令没有用。

要指定 KMS 密钥，请使用 `key-id` 参数。此示例使用密钥 ID 值，但您可以在此命令中使用密钥 ID 或密钥 ARN。

```
aws kms list-key-policies \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

输出：

```
{  
  "PolicyNames": [  
    "default"  
  ]  
}
```

有关 AWS KMS 密钥策略的更多信息，请参阅《[密钥管理服务开发人员指南](#)》中的在 [AWS KMS 中使用 AWS 密钥策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListKeyPolicies](#) 中的。

## list-key-rotations

以下代码示例演示了如何使用 `list-key-rotations`。

AWS CLI

检索有关所有已完成的密钥材料轮换的信息

以下 `list-key-rotations` 示例列出了有关指定 KMS 密钥的所有已完成密钥材料轮换的信息。

```
aws kms list-key-rotations \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

输出：

```
{  
  "Rotations": [  
    {  
      "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",  
      "RotationDate": "2024-03-02T10:11:36.564000+00:00",  
      "RotationType": "AUTOMATIC"  
    },  
  ],  
}
```

```
    {
      "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
      "RotationDate": "2024-04-05T15:14:47.757000+00:00",
      "RotationType": "ON_DEMAND"
    }
  ],
  "Truncated": false
}
```

有关更多信息，请参阅 [《密AWS 钥管理服务开发人员指南》中的轮换密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListKeyRotations](#)中的。

## list-keys

以下代码示例演示了如何使用 list-keys。

### AWS CLI

获取账户和区域中的 KMS 密钥

以下 list-keys 示例获取账户和区域中的 KMS 密钥。此命令同时返回 AWS 托管密钥和客户托管密钥。

```
aws kms list-keys
```

输出：

```
{
  "Keys": [
    {
      "KeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    {
      "KeyArn": "arn:aws:kms:us-
west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
      "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"
    },
    {
      "KeyArn": "arn:aws:kms:us-
east-2:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d",
```

```
        "KeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
      }
    ]
  }
```

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[查看密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListKeys](#)中的。

## list-resource-tags

以下代码示例演示了如何使用 `list-resource-tags`。

### AWS CLI

获取 KMS 密钥上的标签

以下`list-resource-tags`示例获取 KMS 密钥的标签。要在 KMS 密钥上添加或替换资源标签，请使用`tag-resource`命令。输出显示此 KMS 密钥有两个资源标签，每个标签都有一个密钥和值。

要指定 KMS 密钥，请使用 `key-id` 参数。此示例使用密钥 ID 值，但您可以在此命令中使用密钥 ID 或密钥 ARN。

```
aws kms list-resource-tags \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

输出：

```
{
  "Tags": [
    {
      "TagKey": "Dept",
      "TagValue": "IT"
    },
    {
      "TagKey": "Purpose",
      "TagValue": "Test"
    }
  ],
  "Truncated": false
}
```

有关在 AWS KMS 中使用标签的更多信息，请参阅《[密钥管理服务开发人员指南](#)》中的[为密AWS 钥添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResourceTags](#)中的。

## list-retirable-grants

以下代码示例演示了如何使用 list-retirable-grants。

### AWS CLI

查看委托人可以退休的补助金

以下list-retirable-grants示例显示了ExampleAdmin用户可以停用 AWS 账户和区域中的 KMS 密钥的所有授权。您可以使用类似这样的命令来查看任何账户委托人可以停用 AWS 账户和区域中的 KMS 密钥的授权。

必填retiring-principal参数的值必须是账户、用户或角色的 Amazon 资源名称 (ARN)。

即使服务可以是即将停用的主体，也不能在此命令retiring-principal中为的值指定服务。要查找以特定服务为终止主体的授权，请使用list-grants命令。

输出显示ExampleAdmin用户有权停用账户和区域中两个不同 KMS 密钥的授权。除了退休本金外，该账户还有权撤回账户中的任何赠款。

```
aws kms list-retirable-grants \  
  --retiring-principal arn:aws:iam::111122223333:user/ExampleAdmin
```

输出：

```
{  
  "Grants": [  
    {  
      "KeyId": "arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
      "GrantId":  
"156b69c63cb154aa21f59929fff19760717be8d9d82b99df53e18b94a15a5e88e",  
      "Name": "",  
      "CreationDate": 2021-01-14T20:17:36.419000+00:00,  
      "GranteePrincipal": "arn:aws:iam::111122223333:user/ExampleUser",  
      "RetiringPrincipal": "arn:aws:iam::111122223333:user/ExampleAdmin",  
      "IssuingAccount": "arn:aws:iam::111122223333:root",
```

```

    "Operations": [
      "Encrypt"
    ],
    "Constraints": {
      "EncryptionContextSubset": {
        "Department": "IT"
      }
    }
  },
  {
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
    "GrantId":
"8c94d1f12f5e69f440bae30eaec9570bb1fb7358824f9ddf1aa5a0dab1a59b2",
    "Name": "",
    "CreationDate": "2021-02-02T19:49:49.638000+00:00",
    "GranteePrincipal": "arn:aws:iam::111122223333:role/ExampleRole",
    "RetiringPrincipal": "arn:aws:iam::111122223333:user/ExampleAdmin",
    "IssuingAccount": "arn:aws:iam::111122223333:root",
    "Operations": [
      "Decrypt"
    ],
    "Constraints": {
      "EncryptionContextSubset": {
        "Department": "IT"
      }
    }
  }
],
"Truncated": false
}

```

有关更多信息，请参阅《密AWS 钥管理服务开发人员指南》中的 [AWS KMS 中的授权](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRetirableGrants](#)中的。

## put-key-policy

以下代码示例演示了如何使用 put-key-policy。

### AWS CLI

#### 更改 KMS 密钥的密钥策略

以下 `put-key-policy` 示例更改客户托管密钥的密钥策略。

首先，创建密钥策略并将其保存在本地 JSON 文件中。在本示例中，该文件为 `key_policy.json`。您也可以将密钥策略指定为 `policy` 参数的字符串值。

此密钥策略中的第一条语句允许 AWS 账户使用 IAM 策略来控制对 KMS 密钥的访问。第二条语句向 `test-user` 用户授予针对 KMS 密钥运行 `describe-key` 和 `list-keys` 命令的权限。

`key_policy.json` 的内容：

```
{
  "Version" : "2012-10-17",
  "Id" : "key-default-1",
  "Statement" : [
    {
      "Sid" : "Enable IAM User Permissions",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:root"
      },
      "Action" : "kms:*",
      "Resource" : "*"
    },
    {
      "Sid" : "Allow Use of Key",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:user/test-user"
      },
      "Action" : [
        "kms:DescribeKey",
        "kms:ListKeys"
      ],
      "Resource" : "*"
    }
  ]
}
```

为了识别 KMS 密钥，此示例使用密钥 ID，但您也可以使用密钥 ARN。为了指定密钥策略，该命令使用 `policy` 参数。为了表示策略位于文件中，它使用所需的 `file://` 前缀。需要使用此前缀来识别所有受支持操作系统上的文件。最后，该命令使用值为 `default` 的 `policy-name` 参数。如果未指定策略名称，则默认值为 `default`。唯一有效值为 `default`。



```
aws kms put-key-policy \  
  --policy-name default \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --policy file://key_policy.json
```

此命令不生成任何输出。要验证命令是否有效，请使用 `get-key-policy` 命令。以下示例命令获取相同 KMS 密钥的密钥策略。值为 `text` 的 `output` 参数返回一种易于读取的文本格式。

```
aws kms get-key-policy \  
  --policy-name default \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --output text
```

输出：

```
{  
  "Version" : "2012-10-17",  
  "Id" : "key-default-1",  
  "Statement" : [  
    {  
      "Sid" : "Enable IAM User Permissions",  
      "Effect" : "Allow",  
      "Principal" : {  
        "AWS" : "arn:aws:iam::111122223333:root"  
      },  
      "Action" : "kms:*",  
      "Resource" : "*"   
    },  
    {  
      "Sid" : "Allow Use of Key",  
      "Effect" : "Allow",  
      "Principal" : {  
        "AWS" : "arn:aws:iam::111122223333:user/test-user"  
      },  
      "Action" : [ "kms:Describe", "kms:List" ],  
      "Resource" : "*"   
    }  
  ]  
}
```

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[更改密钥策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutKeyPolicy](#)中的。

## re-encrypt

以下代码示例演示了如何使用 re-encrypt。

### AWS CLI

示例 1：使用不同的对称 KMS 密钥重新加密加密消息 ( Linux 和 macOS )。

以下re-encrypt命令示例演示了使用 AWS CLI 重新加密数据的推荐方法。

在文件中提供加密文字。在 --ciphertext-blob 参数的值中，使用 fileb:// 前缀，它将指示 CLI 从二进制文件中读取数据。如果文件不在当前目录中，请键入文件的完整路径。有关从文件读取 AWS CLI 参数值的更多信息，请参阅《AWS 命令行界面用户指南》中的“从文件加载 AWS CLI 参数 < <https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-parameters-file.html>>”和AWS 命令行工具博客中的本地文件参数最佳实践 < <https://aws.amazon.com/blogs/developer/best-practices-for-local-file-parameters/>>。指定源 KMS 密钥，用于解密密文。使用 symmetric 解密时不需要参数 --source-key-id 加密 KMS 密钥。AWS KMS 可以从密文 blob 中的元数据中获取用于加密数据的 KMS 密钥。但是，指定您正在使用的 KMS 密钥始终是最佳实践。此做法可确保您使用预期的 KMS 密钥，并防止您意外使用不信任的 KMS 密钥解密加密文字。指定目标 KMS 密钥来重新加密数据。--destination-key-id 参数始终为必需项。此示例使用密钥 ARN，但您可以使用任何有效的密钥标识符。将明文输出请求为文本值。--query 参数指示 CLI 仅从输出中获取 Plaintext 字段的值。--output 参数以 text.base64 解码格式返回明文输出并将其保存在文件中。以下示例将 Plaintext 参数的值传送 ( | ) 给 Base64 实用工具，该程序负责对其进行解码。然后，它将解码后的输出重定向 ( > ) 到 ExamplePlaintext 文件。

在运行此命令之前，请将示例密钥 ID 替换为 AWS 账户中的有效密钥标识符。

```
aws kms re-encrypt \  
  --ciphertext-blob fileb://ExampleEncryptedFile \  
  --source-key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --destination-key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \  
  --query CiphertextBlob \  
  --output text | base64 --decode > ExampleReEncryptedFile
```

此命令不生成任何输出。re-encrypt 命令的输出经过 base64 解码并保存在文件中。

有关更多信息，请参阅《AWS 密钥管理服务 API 参考》中的 ReEncrypt < [https://docs.aws.amazon.com/kms/latest/APIReference/API\\_ReEncrypt.html](https://docs.aws.amazon.com/kms/latest/APIReference/API_ReEncrypt.html)。

示例 2：使用不同的对称 KMS 密钥重新加密加密消息 ( Windows 命令提示符 )。

以下 `re-encrypt` 命令示例与前一个示例相同，不同之处在于，它使用 `certutil` 实用工具对明文数据进行 Base64 解码。此过程需要两个命令，如以下示例所示。

在运行此命令之前，请将示例密钥 ID 替换为 AWS 账户中的有效密钥 ID。

```
aws kms re-encrypt ^
  --ciphertext-blob fileb://ExampleEncryptedFile ^
  --source-key-id 1234abcd-12ab-34cd-56ef-1234567890ab ^
  --destination-key-id 0987dcba-09fe-87dc-65ba-ab0987654321 ^
  --query CiphertextBlob ^
  --output text > ExampleReEncryptedFile.base64
```

然后使用 `certutil` 实用工具

```
certutil -decode ExamplePlaintextFile.base64 ExamplePlaintextFile
```

输出：

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

有关更多信息，请参阅《AWS 密钥管理服务 API 参考》中的 `ReEncrypt` < [https://docs.aws.amazon.com/kms/latest/APIReference/API\\_ReEncrypt.html](https://docs.aws.amazon.com/kms/latest/APIReference/API_ReEncrypt.html)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ReEncrypt](#) 中的。

## retire-grant

以下代码示例演示了如何使用 `retire-grant`。

### AWS CLI

停用对客户主密钥的授权

以下 `retire-grant` 示例从 KMS 密钥中删除授权。

以下示例命令指定 `grant-id` 和 `key-id` 参数。`key-id` 参数值必须为 KMS 密钥的密钥 ARN。

```
aws kms retire-grant \
```

```
--grant-id 1234a2345b8a4e350500d432bccf8ecd6506710e1391880c4f7f7140160c9af3 \  
--key-id arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

此命令不生成任何输出。要确认授权是否已停用，请使用 `list-grants` 命令。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[停用和撤消授权](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RetireGrant](#)中的。

## revoke-grant

以下代码示例演示了如何使用 `revoke-grant`。

### AWS CLI

#### 撤消对客户主密钥的授权

以下 `revoke-grant` 示例从 KMS 密钥中删除授权。以下示例命令指定 `grant-id` 和 `key-id` 参数。`key-id` 参数的值可以是 KMS 密钥的密钥 ID 或密钥 ARN。

```
aws kms revoke-grant \  
--grant-id 1234a2345b8a4e350500d432bccf8ecd6506710e1391880c4f7f7140160c9af3 \  
--key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

此命令不生成任何输出。要确认授权是否已撤消，请使用 `list-grants` 命令。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[停用和撤消授权](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RevokeGrant](#)中的。

## rotate-key-on-demand

以下代码示例演示了如何使用 `rotate-key-on-demand`。

### AWS CLI

#### 按需轮换 KMS 密钥

以下 `rotate-key-on-demand` 示例立即启动指定 KMS 密钥的密钥材料的轮换。

```
aws kms rotate-key-on-demand \  

```

```
--key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

输出：

```
{
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

有关更多信息，请参阅 [《密钥管理服务开发人员指南》](#) 中的 [如何执行按需 AWS 密钥轮换](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RotateKeyOnDemand](#) 中的。

## schedule-key-deletion

以下代码示例演示了如何使用 schedule-key-deletion。

### AWS CLI

计划删除客户托管的 KMS 密钥。

以下 schedule-key-deletion 示例计划在 15 天后删除指定的客户托管 KMS 密钥。

--key-id 参数识别 KMS 密钥。此示例使用密钥 ARN 值，但您可以使用 KMS 密钥的密钥 ID 或 ARN。--pending-window-in-days 参数指定 7-30 天的等待期限。默认的等待期限为 30 天。此示例指定的值为 15，表示在命令完成 15 天后永久删除 KMS 密钥。AWS

```
aws kms schedule-key-deletion \
  --key-id arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
  --pending-window-in-days 15
```

响应包括密钥 ARN、密钥状态、等待期 ( PendingWindowInDays ) 和删除日期 ( 以 Unix 时间表示 )。要以当地时间查看删除日期，请使用 AWS KMS 控制台。无法在加密操作中使用密钥状态为 PendingDeletion 的 KMS 密钥。

```
{
  "KeyId": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "DeletionDate": "2022-06-18T23:43:51.272000+00:00",
  "KeyState": "PendingDeletion",
  "PendingWindowInDays": 15
}
```

```
}
```

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[删除密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ScheduleKeyDeletion](#)中的。

## sign

以下代码示例演示了如何使用 sign。

### AWS CLI

示例 1：为消息生成数字签名

以下 sign 示例为一条短消息生成加密签名。该命令的输出包括一个 base-64 编码 Signature 字段，您可以使用该命令对其进行验证。verify

您必须指定要签名的消息以及您的非对称 KMS 密钥支持的签名算法。要获取 KMS 密钥的签名算法，请使用 describe-key 命令。

在 AWS CLI 2.0 中，message 参数的值必须采用 Base64 编码。或者，您可以将消息保存在文件中并使用 fileb:// 前缀，它告诉 AWS CLI 从文件中读取二进制数据。

在运行此命令之前，请将示例密钥 ID 替换为 AWS 账户中的有效密钥 ID。密钥 ID 必须表示密钥用法为 SIGN\_VERIFY 的非对称 KMS 密钥。

```
msg=(echo 'Hello World' | base64)

aws kms sign \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --message fileb://UnsignedMessage \
  --message-type RAW \
  --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256
```

输出：

```
{
  "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "Signature": "ABCDEFhpyVYyTxbaFE74ccSvEJLJr3zuoV1Hfymz4qv+
fxmxNLA7SE1SiF8lHw80fKZZ3bJ...",
  "SigningAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
```

```
}
```

有关在 AWS KMS 中使用非对称 KMS 密钥的更多信息，请参阅 [《密钥管理服务开发人员指南》中的 AWS KMS 中的非对称AWS密钥](#)。

示例 2：将数字签名保存在文件中（Linux 和 macOS）

以下sign示例为存储在本地文件中的短消息生成加密签名。该命令还会从响应中获取Signature属性，Base64 对其进行解码并将其保存在文件中。ExampleSignature 可以在验证签名的verify命令中使用签名文件。

该sign命令需要一条 Base64 编码的消息和您的非对称 KMS 密钥支持的签名算法。要获取您的 KMS 密钥支持的签名算法，请使用describe-key命令。

在运行此命令之前，请将示例密钥 ID 替换为 AWS 账户中的有效密钥 ID。密钥 ID 必须表示密钥用法为 SIGN\_VERIFY 的非对称 KMS 密钥。

```
echo 'hello world' | base64 > EncodedMessage

aws kms sign \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --message fileb://EncodedMessage \
  --message-type RAW \
  --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256 \
  --output text \
  --query Signature | base64 --decode > ExampleSignature
```

此命令不生成任何输出。此示例提取输出的Signature属性并将其保存在文件中。

有关在 AWS KMS 中使用非对称 KMS 密钥的更多信息，请参阅 [《密钥管理服务开发人员指南》中的 AWS KMS 中的非对称AWS密钥](#)。

- 有关 API 的详细信息，请参阅[登录AWS CLI命令参考](#)。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

向 KMS 密钥添加标签

以下tag-resource示例为客户托管的 KMS 密钥添加"Purpose":"Test"和"Dept":"IT"标记。您可以使用此类标签来标记 KMS 密钥并创建 KMS 密钥类别以进行权限和审计。

要指定 KMS 密钥，请使用 key-id 参数。此示例使用密钥 ID 值，但您可以在此命令中使用密钥 ID 或密钥 ARN。

```
aws kms tag-resource \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --tags TagKey='Purpose',TagValue='Test' TagKey='Dept',TagValue='IT'
```

此命令不生成任何输出。要查看 AWS KMS KMS 密钥上的标签，请使用list-resource-tags命令。

有关在 AWS KMS 中使用标签的更多信息，请参阅《[密钥管理服务开发人员指南](#)》中的[为密AWS 钥添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从 KMS 密钥中删除标签

以下untag-resource示例从客户托管的 KMS "Purpose" 密钥中删除带有密钥的标签。

要指定 KMS 密钥，请使用 key-id 参数。此示例使用密钥 ID 值，但您可以在此命令中使用密钥 ID 或密钥 ARN。在运行此命令之前，请将示例密钥 ID 替换为 AWS 账户中的有效密钥 ID。

```
aws kms untag-resource \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --tag-key 'Purpose'
```

此命令不生成任何输出。要查看 AWS KMS KMS 密钥上的标签，请使用list-resource-tags命令。

有关在 AWS KMS 中使用标签的更多信息，请参阅《[密钥管理服务开发人员指南](#)》中的[为密AWS 钥添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。



## update-alias

以下代码示例演示了如何使用 update-alias。

### AWS CLI

将别名与其他 KMS 密钥关联

以下 update-alias 示例将别名 alias/test-key 与其他 KMS 密钥关联起来。

--alias-name 参数指定别名。别名名称值必须以 alias/ 开头。--target-key-id 参数指定要与别名关联的 KMS 密钥。您不需要为别名指定当前 KMS 密钥。

```
aws kms update-alias \  
  --alias-name alias/test-key \  
  --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

此命令不生成任何输出。要查找别名，请使用 list-aliases 命令。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的[更新别名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateAlias](#) 中的。

## update-custom-key-store

以下代码示例演示了如何使用 update-custom-key-store。

### AWS CLI

示例 1：编辑自定义密钥库的友好名称

以下 update-custom-key-store 示例更改了自定义密钥库的名称。此示例适用于 AWS CloudHSM 密钥存储库或外部密钥存储库。

使用 custom-key-store-id 来识别密钥库。使用 new-custom-key-store-name 参数指定新的友好名称。

要更新 AWS CloudHSM 密钥库的友好名称，必须先断开密钥库的连接，例如使用命令 disconnect-custom-key-store。在外部密钥存储库连接或断开连接时，您可以更新其友好名称。要查找自定义密钥库的连接状态，请使用 describe-custom-key-store 命令。

```
aws kms update-custom-key-store \  
  --custom-key-store-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --new-custom-key-store-name my-new-key-store
```

```
--custom-key-store-id cks-1234567890abcdef0 \  
--new-custom-key-store-name ExampleKeyStore
```

此命令不返回任何数据。要验证命令是否有效，请使用 `describe-custom-key-stores` 命令。

有关更新 AWS CloudHSM 密钥库的更多信息，请参阅《[密钥管理服务开发者指南](#)》中的 [编辑 AWS CloudHSM 密钥库](#) 设置。AWS

有关更新外部密钥存储的更多信息，请参阅《[密钥管理服务开发人员指南](#)》中的 [编辑外部AWS 密钥存储库属性](#)。

示例 2：编辑 CloudHSM 密钥库的 kmsuser 密码

以下 `update-custom-key-store` 示例将 kmsuser 密码值更新为与指定密钥库关联的 CloudHSM 集群 kmsuser 中的当前密码。此命令不会更改集群的 kmsuser 密码。它只是告诉 AWS KMS 当前的密码。如果 KMS 没有当前 kmsuser 密码，则无法连接到 AWS CloudHSM 密钥存储库。

注意：在更新 AWS CloudHSM 密钥库之前，必须将其断开连接。使用 `disconnect-custom-key-store` 命令。命令完成后，您可以重新连接 Cloud AWS HSM 密钥存储库。使用 `connect-custom-key-store` 命令。

```
aws kms update-custom-key-store \  
  --custom-key-store-id cks-1234567890abcdef0 \  
  --key-store-password ExamplePassword
```

此命令不返回任何输出。要验证更改是否有效，请使用 `describe-custom-key-stores` 命令。

有关更新 AWS CloudHSM 密钥库的更多信息，请参阅《[密钥管理服务开发者指南](#)》中的 [编辑 AWS CloudHSM 密钥库](#) 设置。AWS

示例 3：编辑 AWS CloudHSM 密钥库的 CloudHSM 集群 AWS

以下示例将与 AWS CloudHSM 密钥库关联的 AWS CloudHSM 集群更改为相关集群，例如同一个集群的不同备份。

注意：在更新 AWS CloudHSM 密钥库之前，必须将其断开连接。使用 `disconnect-custom-key-store` 命令。命令完成后，您可以重新连接 Cloud AWS HSM 密钥存储库。使用 `connect-custom-key-store` 命令。

```
aws kms update-custom-key-store \  
  --key-store-id cks-1234567890abcdef0 \  
  --key-store-alias ExampleKeyStoreAlias
```

```
--custom-key-store-id cks-1234567890abcdef0 \  
--cloud-hsm-cluster-id cluster-1a23b4cdefg
```

此命令不返回任何输出。要验证更改是否有效，请使用 `describe-custom-key-stores` 命令。

有关更新 AWS CloudHSM 密钥库的更多信息，请参阅《[密钥管理服务开发者指南](#)》中的 [编辑 AWS CloudHSM 密钥库](#) 设置。AWS

#### 示例 4：编辑外部密钥存储库的代理身份验证凭据

以下示例更新了您的外部密钥存储的代理身份验证凭据。即使您只更改其中一个值 `access-key-id`，也必须同时指定 `raw-secret-access-key`。您可以使用此功能修复无效凭据或在外部密钥存储代理轮换凭据时更改凭据。

在您的外部密钥存储上建立 AWS KMS 的代理身份验证凭据。然后使用此命令向 AWS KMS 提供凭证。AWS KMS 使用此凭证对您的外部密钥存储代理的请求进行签名。

您可以在连接或断开外部密钥存储库连接或断开连接时更新代理身份验证凭据。要查找自定义密钥库的连接状态，请使用 `describe-custom-key-store` 命令。

```
aws kms update-custom-key-store \  
  --custom-key-store-id cks-1234567890abcdef0 \  
  --xks-proxy-authentication-credential "AccessKeyId=ABCDE12345670EXAMPLE,  
  RawSecretAccessKey=DXjSUawne12fr6SKC7G25CNxTyWKE5PF9XX6H/u9pSo="
```

此命令不返回任何输出。要验证更改是否有效，请使用 `describe-custom-key-stores` 命令。

有关更新外部密钥存储的更多信息，请参阅《[密钥管理服务开发人员指南](#)》中的 [编辑外部 AWS 密钥存储库属性](#)。

#### 示例 5：编辑外部密钥存储库的代理连接

以下示例将外部密钥存储代理连接选项从公共终端节点连接更改为 VPC 终端节点服务连接。除了更改 `xks-proxy-connectivity` 值外，您还必须更改该 `xks-proxy-uri-endpoint` 值以反映与 VPC 终端节点服务关联的私有 DNS 名称。您还必须添加一个 `xks-proxy-vpc-endpoint-service-name` 值。

注意：在更新外部存储的代理连接之前，必须断开其连接。使用 `disconnect-custom-key-store` 命令。命令完成后，您可以使用 `connect-custom-key-store` 命令重新连接外部密钥存储库。

```
aws kms update-custom-key-store \  
  --custom-key-store-id cks-1234567890abcdef0 \  
  --xks-proxy-connectivity VPC_ENDPOINT_SERVICE \  
  --xks-proxy-uri-endpoint "https://myproxy-private.xks.example.com" \  
  --xks-proxy-vpc-endpoint-service-name "com.amazonaws.vpce.us-east-1.vpce-svc-  
example"
```

此命令不返回任何输出。要验证更改是否有效，请使用describe-custom-key-stores命令。

有关更新外部密钥存储的更多信息，请参阅《[密钥管理服务开发人员指南](#)》中的[编辑外部AWS 密钥存储库属性](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateCustomKeyStore](#)中的。

## update-key-description

以下代码示例演示了如何使用 update-key-description。

### AWS CLI

示例 1：为客户管理的 KMS 密钥添加或更改描述

以下update-key-description示例为客户托管的 KMS 密钥添加了描述。您可以使用相同的命令来更改现有描述。

该--key-id参数用于标识命令中的 KMS 密钥。此示例使用密钥 ARN 值，但您可以使用 KMS 密钥的密钥 ID 或密钥 ARN。该--description参数指定了新的描述。此参数的值将替换 KMS 密钥的当前描述（如果有）。

```
aws kms update-key-description \  
  --key-id arn:aws:kms:us-  
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \  
  --description "IT Department test key"
```

此命令不生成任何输出。要查看 KMS 密钥的描述，请使用describe-key命令。

有关更多信息，请参阅AWS 密钥管理服务 API 参考[UpdateKeyDescription](#)中的。

示例 2：删除客户托管的 KMS 密钥的描述

以下update-key-description示例删除了客户托管的 KMS 密钥的描述。

该 `--key-id` 参数用于标识命令中的 KMS 密钥。此示例使用密钥 ID 值，但您可以使用 KMS 密钥的密钥 ID 或密钥 ARN。带有空字符串值 ("") 的 `--description` 参数会删除现有描述。

```
aws kms update-key-description \  
  --key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \  
  --description ''
```

此命令不生成任何输出。要查看 KMS 密钥的描述，请使用 `describe-key` 命令。

有关更多信息，请参阅 AWS 密钥管理服务 API 参考 [UpdateKeyDescription](#) 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateKeyDescription](#) 中的。

## verify

以下代码示例演示了如何使用 `verify`。

### AWS CLI

#### 验证数字签名

以下 `verify` 示例验证了一条 Base64 编码的简短消息的加密签名。密钥 ID、消息、消息类型和签名算法必须与用于签名消息的算法相同。您指定的签名不能采用 base64 编码。如需解码 `sign` 命令返回的签名的帮助，请参阅 `sign` 命令示例。

该命令的输出包括一个布尔 `SignatureValid` 字段，表示签名已通过验证。如果签名验证失败，`verify` 命令也会失败。

在运行此命令之前，请将示例密钥 ID 替换为 AWS 账户中的有效密钥 ID。

```
aws kms verify \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --message fileb://EncodedMessage \  
  --message-type RAW \  
  --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256 \  
  --signature fileb://ExampleSignature
```

输出：

```
{
```

```
"KeyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
"SignatureValid": true,
"SigningAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
}
```

有关在 KMS 中 AWS 使用非对称 KMS 密钥的更多信息，请参阅[密钥管理服务开发人员指南中的 AWS 使用非对称密钥](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的[“验证”](#)。

## 使用 Lake Formation 示例 AWS CLI

以下代码示例向您展示了如何使用 with Lake Formation 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **add-lf-tags-to-resource**

以下代码示例演示了如何使用 add-lf-tags-to-resource。

#### AWS CLI

将一个或多个 LF 标签附加到现有资源

以下add-lf-tags-to-resource示例将给定的 LF-Tag 附加到表资源。

```
aws lakeformation add-lf-tags-to-resource \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "CatalogId": "123456789111",
  "Resource": {
    "Table": {
      "CatalogId": "123456789111",
      "DatabaseName": "tpc",
      "Name": "dl_tpc_promotion"
    }
  },
  "LFTags": [{
    "CatalogId": "123456789111",
    "TagKey": "usergroup",
    "TagValues": [
      "analyst"
    ]
  }]
}
```

输出：

```
{
  "Failures": []
}
```

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的为数据目录资源分配 LF 标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AddLfTagsToResource](#) 中的。

## batch-grant-permissions

以下代码示例演示了如何使用 batch-grant-permissions。

### AWS CLI

向委托人批量授予资源权限

以下 batch-grant-permissions 示例批量向委托人授予对指定资源的访问权限。

```
aws lakeformation batch-grant-permissions \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "CatalogId": "123456789111",
  "Entries": [{
    "Id": "1",
    "Principal": {
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-
developer"
    },
    "Resource": {
      "Table": {
        "CatalogId": "123456789111",
        "DatabaseName": "tpc",
        "Name": "dl_tpc_promotion"
      }
    },
    "Permissions": [
      "ALL"
    ],
    "PermissionsWithGrantOption": [
      "ALL"
    ]
  },
  {
    "Id": "2",
    "Principal": {
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-
developer"
    },
    "Resource": {
      "Table": {
        "CatalogId": "123456789111",
        "DatabaseName": "tpc",
        "Name": "dl_tpc_customer"
      }
    },
    "Permissions": [
      "ALL"
    ],
    "PermissionsWithGrantOption": [
      "ALL"
    ]
  },
}
```



```
{
  "Id": "3",
  "Principal": {
    "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-
business-analyst"
  },
  "Resource": {
    "Table": {
      "CatalogId": "123456789111",
      "DatabaseName": "tpc",
      "Name": "dl_tpc_promotion"
    }
  },
  "Permissions": [
    "ALL"
  ],
  "PermissionsWithGrantOption": [
    "ALL"
  ]
},
{
  "Id": "4",
  "Principal": {
    "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-
developer"
  },
  "Resource": {
    "DataCellsFilter": {
      "TableCatalogId": "123456789111",
      "DatabaseName": "tpc",
      "TableName": "dl_tpc_item",
      "Name": "developer_item"
    }
  },
  "Permissions": [
    "SELECT"
  ],
  "PermissionsWithGrantOption": []
}
]
```

输出：

```
{
  "Failures": []
}
```

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的授予和撤消数据目录资源的权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchGrantPermissions](#) 中的。

## batch-revoke-permissions

以下代码示例演示了如何使用 batch-revoke-permissions。

### AWS CLI

批量撤消委托人对资源的权限

以下 batch-revoke-permissions 示例批量撤消委托人对指定资源的访问权限。

```
aws lakeformation batch-revoke-permissions \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "CatalogId": "123456789111",
  "Entries": [{
    "Id": "1",
    "Principal": {
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-developer"
    },
    "Resource": {
      "Table": {
        "CatalogId": "123456789111",
        "DatabaseName": "tpc",
        "Name": "dl_tpc_promotion"
      }
    },
    "Permissions": [
      "ALL"
    ],
    "PermissionsWithGrantOption": [
```

```

        "ALL"
      ]
    },
    {
      "Id": "2",
      "Principal": {
        "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-
business-analyst"
      },
      "Resource": {
        "Table": {
          "CatalogId": "123456789111",
          "DatabaseName": "tpc",
          "Name": "dl_tpc_promotion"
        }
      },
      "Permissions": [
        "ALL"
      ],
      "PermissionsWithGrantOption": [
        "ALL"
      ]
    }
  ]
}

```

输出：

```

{
  "Failures": []
}

```

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的授予和撤销数据目录资源的权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchRevokePermissions](#) 中的。

## cancel-transaction

以下代码示例演示了如何使用 cancel-transaction。

AWS CLI

取消交易

以下cancel-transaction示例取消了交易。

```
aws lakeformation cancel-transaction \  
  --transaction-id='b014d972ca8347b89825e33c5774aec4'
```

此命令不生成任何输出。

有关更多信息，请参阅 [Lake Formation 开发者指南中的在事务中读取和写入数据AWS 湖](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelTransaction](#)中的。

## commit-transaction

以下代码示例演示了如何使用 commit-transaction。

### AWS CLI

#### 提交事务

以下commit-transaction示例提交了交易。

```
aws lakeformation commit-transaction \  
  --transaction-id='b014d972ca8347b89825e33c5774aec4'
```

输出：

```
{  
  "TransactionStatus": "committed"  
}
```

有关更多信息，请参阅 [Lake Formation 开发者指南中的在事务中读取和写入数据AWS 湖](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CommitTransaction](#)中的。

## create-data-cells-filter

以下代码示例演示了如何使用 create-data-cells-filter。

### AWS CLI

#### 示例 1：创建数据单元格筛选器

以下`create-data-cells-filter`示例创建了一个数据单元格筛选器，允许用户根据行条件授予对某些列的访问权限。

```
aws lakeformation create-data-cells-filter \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "TableData": {  
    "ColumnNames": ["p_channel_details", "p_start_date_sk", "p_promo_name"],  
    "DatabaseName": "tpc",  
    "Name": "developer_promotion",  
    "RowFilter": {  
      "FilterExpression": "p_promo_name='ese'"  
    },  
    "TableCatalogId": "123456789111",  
    "TableName": "dl_tpc_promotion"  
  }  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [Lake Formation 开发者指南中的 Lake Formation 中的AWS 数据筛选和单元级安全](#)。

#### 示例 2：创建列筛选器

以下`create-data-cells-filter`示例创建了一个数据筛选器，允许用户授予对某些列的访问权限。

```
aws lakeformation create-data-cells-filter \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "TableData": {  
    "ColumnNames": ["p_channel_details", "p_start_date_sk", "p_promo_name"],  
    "DatabaseName": "tpc",  
    "Name": "developer_promotion_allrows",
```

```
    "RowFilter": {
      "AllRowsWildcard": {}
    },
    "TableCatalogId": "123456789111",
    "TableName": "dl_tpc_promotion"
  }
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [Lake Formation 开发者指南中的 Lake Formation 中的AWS 数据筛选和单元级安全](#)。

### 示例 3：使用排除列创建数据筛选器

以下create-data-cells-filter示例创建了一个数据筛选器，允许用户授予除上述列以外的所有列的访问权限。

```
aws lakeformation create-data-cells-filter \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "TableData": {
    "ColumnWildcard": {
      "ExcludedColumnNames": ["p_channel_details", "p_start_date_sk"]
    },
    "DatabaseName": "tpc",
    "Name": "developer_promotion_excludecolumn",
    "RowFilter": {
      "AllRowsWildcard": {}
    },
    "TableCatalogId": "123456789111",
    "TableName": "dl_tpc_promotion"
  }
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [Lake Formation 开发者指南中的 Lake Formation 中的AWS 数据筛选和单元级安全](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDataCellsFilter](#)中的。

## create-lf-tag

以下代码示例演示了如何使用 create-lf-tag。

### AWS CLI

#### 创建 LF-Tag

以下create-lf-tag示例使用指定的名称和值创建一个 LF-Tag。

```
aws lakeformation create-lf-tag \  
  --catalog-id '123456789111' \  
  --tag-key 'usergroup' \  
  --tag-values '["developer","analyst","campaign"]'
```

此命令不生成任何输出。

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的管理 LF 标签以实现元数据访问控制](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLfTag](#)中的。

## delete-data-cells-filter

以下代码示例演示了如何使用 delete-data-cells-filter。

### AWS CLI

#### 删除数据单元格筛选器

以下delete-data-cells-filter示例删除给定的数据单元格筛选器。

```
aws lakeformation delete-data-cells-filter \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{  
  "TableCatalogId": "123456789111",  
  "DatabaseName": "tpc",
```

```
"TableName": "dl_tpc_promotion",  
"Name": "developer_promotion"  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [Lake Formation 开发者指南中的 Lake Formation 中的AWS 数据筛选和单元级安全](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDataCellsFilter](#)中的。

## delete-lf-tag

以下代码示例演示了如何使用 delete-lf-tag。

### AWS CLI

删除 LF-tag 定义

以下delete-lf-tag示例删除了 LF-Tag 的定义。

```
aws lakeformation delete-lf-tag \  
  --catalog-id '123456789111' \  
  --tag-key 'usergroup'
```

此命令不生成任何输出。

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的管理 LF 标签以实现元数据访问控制](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLfTag](#)中的。

## delete-objects-on-cancel

以下代码示例演示了如何使用 delete-objects-on-cancel。

### AWS CLI

取消交易时删除对象

以下delete-objects-on-cancel示例在取消事务时删除列出的 s3 对象。

```
aws lakeformation delete-objects-on-cancel \  
  --catalog-id '123456789111' \  
  --tag-key 'usergroup'
```



```
--cli-input-json file://input.json
```

input.json 的内容：

```
{
  "CatalogId": "012345678901",
  "DatabaseName": "tpc",
  "TableName": "dl_tpc_household_demographics_gov",
  "TransactionId": "1234d972ca8347b89825e33c5774aec4",
  "Objects": [{
    "Uri": "s3://lf-data-lake-012345678901/target/
dl_tpc_household_demographics_gov/run-unnamed-1-part-block-0-r-00000-snappy-
ff26b17504414fe88b302cd795eabd00.parquet",
    "ETag": "1234ab1fc50a316b149b4e1f21a73800"
  }]
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [Lake Formation 开发者指南中的在事务中读取和写入数据AWS 湖](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteObjectsOnCancel](#) 中的。

## deregister-resource

以下代码示例演示了如何使用 deregister-resource。

### AWS CLI

取消注册数据湖存储

以下 deregister-resource 示例将资源注销为由 Lake Formation 管理的资源。

```
aws lakeformation deregister-resource \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "ResourceArn": "arn:aws:s3:::lf-emr-athena-result-123"
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [Lake Formation 开发者指南中的向数据湖添加 Amazon S3 位置](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeregisterResource](#) 中的。

## describe-transaction

以下代码示例演示了如何使用 describe-transaction。

### AWS CLI

检索交易详情

以下 describe-transaction 示例返回单笔交易的详细信息。

```
aws lakeformation describe-transaction \  
  --transaction-id='8cb4b1a7cc8d486fbaca9a64e7d9f5ce'
```

输出：

```
{  
  "TransactionDescription": {  
    "TransactionId": "12345972ca8347b89825e33c5774aec4",  
    "TransactionStatus": "committed",  
    "TransactionStartTime": "2022-08-10T14:29:04.046000+00:00",  
    "TransactionEndTime": "2022-08-10T14:29:09.681000+00:00"  
  }  
}
```

有关更多信息，请参阅 [Lake Formation 开发者指南中的在事务中读取和写入数据AWS 湖](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeTransaction](#) 中的。

## extend-transaction

以下代码示例演示了如何使用 extend-transaction。

### AWS CLI

延长交易

以下 `extend-transaction` 示例扩展了事务。

```
aws lakeformation extend-transaction \  
  --transaction-id='8cb4b1a7cc8d486fbaca9a64e7d9f5ce'
```

此命令不生成任何输出。

有关更多信息，请参阅 [Lake Formation 开发者指南中的在事务中读取和写入数据AWS 湖](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ExtendTransaction](#) 中的。

## get-data-lake-settings

以下代码示例演示了如何使用 `get-data-lake-settings`。

### AWS CLI

检索 AWS Lake Formation 管理的数据湖设置

以下 `get-data-lake-settings` 示例检索数据湖管理员列表和其他数据湖设置。

```
aws lakeformation get-data-lake-settings \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "CatalogId": "123456789111"  
}
```

输出：

```
{  
  "DataLakeSettings": {  
    "DataLakeAdmins": [{  
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-admin"  
    }],  
    "CreateDatabaseDefaultPermissions": [],  
    "CreateTableDefaultPermissions": [  
      {  
        "Principal": {
```

```

        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
      },
      "Permissions": [
        "ALL"
      ]
    }
  ],
  "TrustedResourceOwners": [],
  "AllowExternalDataFiltering": true,
  "ExternalDataFilteringAllowList": [{
    "DataLakePrincipalIdentifier": "123456789111"
  }],
  "AuthorizedSessionTagValueList": [
    "Amazon EMR"
  ]
}
}
}

```

有关更多信息，请参阅 Lake Formation 开发者指南中的更改数据AWS 湖的默认安全设置。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDataLakeSettings](#)中的。

## get-effective-permissions-for-path

以下代码示例演示了如何使用 get-effective-permissions-for-path。

### AWS CLI

检索位于特定路径的资源的权限

以下get-effective-permissions-for-path示例返回位于 Amazon S3 中某个路径上的指定表或数据库资源的 Lake Formation 权限。

```
aws lakeformation get-effective-permissions-for-path \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "CatalogId": "123456789111",
  "ResourceArn": "arn:aws:s3:::lf-data-lake-123456789111"
}
```

输出：

```
{
  "Permissions": [{
    "Principal": {
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-
campaign-manager"
    },
    "Resource": {
      "Database": {
        "Name": "tpc"
      }
    },
    "Permissions": [
      "DESCRIBE"
    ],
    "PermissionsWithGrantOption": []
  },
  {
    "Principal": {
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:role/EMR-
RuntimeRole"
    },
    "Resource": {
      "Database": {
        "Name": "tpc"
      }
    },
    "Permissions": [
      "ALL"
    ],
    "PermissionsWithGrantOption": []
  },
  {
    "Principal": {
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:saml-
provider/oktaSAMLProvider:user/emr-developer"
    },
    "Resource": {
      "Database": {
        "Name": "tpc"
      }
    },
    "Permissions": [
```

```
        "ALL",
        "DESCRIBE"
    ],
    "PermissionsWithGrantOption": []
},
{
    "Principal": {
        "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-
admin"
    },
    "Resource": {
        "Database": {
            "Name": "tpc"
        }
    },
    "Permissions": [
        "ALL",
        "ALTER",
        "CREATE_TABLE",
        "DESCRIBE",
        "DROP"
    ],
    "PermissionsWithGrantOption": [
        "ALL",
        "ALTER",
        "CREATE_TABLE",
        "DESCRIBE",
        "DROP"
    ]
},
{
    "Principal": {
        "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:role/LF-
GlueServiceRole"
    },
    "Resource": {
        "Database": {
            "Name": "tpc"
        }
    },
    "Permissions": [
        "CREATE_TABLE"
    ],
    "PermissionsWithGrantOption": []
}
```

```
    }
  ],
  "NextToken":
  "E5S1JDSTZ1eUp6SWpvaU9UQTN0RE0zTXpFeE5Ua3pJbjE5TENKbGVIQnBjbUYwYVc5dUlqcDdJbk5sWTI5dVpITWlP
}
```

有关更多信息，请参阅 [Lake Formation 开发者指南中的管理AWS Lake Formation 权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetEffectivePermissionsForPath](#)中的。

## get-lf-tag

以下代码示例演示了如何使用 get-lf-tag。

### AWS CLI

检索 LF-Tag 的定义

以下get-lf-tag示例检索 LF-Tag 的定义。

```
aws lakeformation get-lf-tag \
  --catalog-id '123456789111' \
  --tag-key 'usergroup'
```

输出：

```
{
  "CatalogId": "123456789111",
  "TagKey": "usergroup",
  "TagValues": [
    "analyst",
    "campaign",
    "developer"
  ]
}
```

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的管理 LF 标签以实现元数据访问控制](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLfTag](#)中的。

## get-query-state

以下代码示例演示了如何使用 `get-query-state`。

### AWS CLI

检索已提交查询的状态

以下 `get-query-state` 示例返回先前提交的查询的状态。

```
aws lakeformation get-query-state \  
  --query-id='1234273f-4a62-4cda-8d98-69615ee8be9b'
```

输出：

```
{  
  "State": "FINISHED"  
}
```

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的事务数据操作](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `GetQueryState`](#) 中的。

## get-query-statistics

以下代码示例演示了如何使用 `get-query-statistics`。

### AWS CLI

检索查询统计信息

以下 `get-query-statistics` 示例检索有关查询计划和执行的统计信息。

```
aws lakeformation get-query-statistics \  
  --query-id='1234273f-4a62-4cda-8d98-69615ee8be9b'
```

输出：

```
{  
  "ExecutionStatistics": {  
    "AverageExecutionTimeMillis": 0,  
    "DataScannedBytes": 0,  
  }  
}
```



```
    "WorkUnitsExecutedCount": 0
  },
  "PlanningStatistics": {
    "EstimatedDataToScanBytes": 43235,
    "PlanningTimeMillis": 2377,
    "QueueTimeMillis": 440,
    "WorkUnitsGeneratedCount": 1
  },
  "QuerySubmissionTime": "2022-08-11T02:14:38.641870+00:00"
}
```

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的事务数据操作](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetQueryStatistics](#) 中的。

## get-resource-lf-tags

以下代码示例演示了如何使用 `get-resource-lf-tags`。

### AWS CLI

列出 LF 标签

以下 `list-lf-tags` 示例返回请求者有权查看的 LF-Tag 列表。

```
aws lakeformation list-lf-tags \
  --cli-input-json file:///input.json
```

`input.json` 的内容：

```
{
  "CatalogId": "123456789111",
  "ResourceShareType": "ALL",
  "MaxResults": 2
}
```

输出：

```
{
  "LFTags": [{
    "CatalogId": "123456789111",
    "TagKey": "category",
```

```

    "TagValues": [
      "private",
      "public"
    ]
  },
  {
    "CatalogId": "123456789111",
    "TagKey": "group",
    "TagValues": [
      "analyst",
      "campaign",
      "developer"
    ]
  }
],
"NextToken": "kIiwiZXhwaXJhdGlvbiI6eyJzZWVbmRzIjoxNjYwMDY4dCI6ZmFsc2V9"
}

```

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的管理 LF 标签以实现元数据访问控制](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetResourceLfTags](#) 中的。

## get-table-objects

以下代码示例演示了如何使用 get-table-objects。

### AWS CLI

列出受管辖表格的对象

以下 get-table-objects 示例返回构成指定受管辖表的一组 Amazon S3 对象。

```
aws lakeformation get-table-objects \
  --cli-input-json file://input.json
```

input.json 的内容：

```

{
  "CatalogId": "012345678901",
  "DatabaseName": "tpc",
  "TableName": "dl_tpc_household_demographics_gov",
  "QueryAsOfTime": "2022-08-10T15:00:00"
}

```

```
}

```

输出：

```
{
  "Objects": [{
    "PartitionValues": [],
    "Objects": [{
      "Uri": "s3://lf-data-lake-012345678901/target/
dl_tpc_household_demographics_gov/run-unnamed-1-part-block-0-r-00000-snappy-
ff26b17504414fe88b302cd795eabd00.parquet",
      "ETag": "12345b1fc50a316b149b4e1f21a73800",
      "Size": 43235
    }]
  }]
}
```

有关更多信息，请参阅 [Lake Formation 开发者指南中的在事务中读取和写入数据AWS 湖](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTableObjects](#)中的。

## get-work-unit-results

以下代码示例演示了如何使用 `get-work-unit-results`。

### AWS CLI

检索给定查询的工作单位

以下`get-work-unit-results`示例返回查询结果的工作单元。

```
aws lakeformation get-work-units \
  --query-id='1234273f-4a62-4cda-8d98-69615ee8be9b' \
  --work-unit-id '0' \
  --work-unit-token 'B2fMSdmQXe9umX8Ux8XCo4=' outfile
```

输出：

```
outfile with Blob content.
```

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的事务数据操作](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetWorkUnitResults](#)中的。

## get-work-units

以下代码示例演示了如何使用 `get-work-units`。

### AWS CLI

#### 检索工作单元

以下 `get-work-units` 示例检索 `StartQueryPlanning` 操作生成的工作单元。

```
aws lakeformation get-work-units \
  --query-id='1234273f-4a62-4cda-8d98-69615ee8be9b'
```

输出：

```
{
  "WorkUnitRanges": [{
    "WorkUnitIdMax": 0,
    "WorkUnitIdMin": 0,
    "WorkUnitToken":
      "1234eMAk4kL04umqEL4Z5WuxL04AXwABABVhd3MtY3J5cHRvLXB1YmxpYy1rZXkAREEwYm9QbkhINmFYTWphbmMxZW
      +f88jzGrYq22gE6jkQlp0B
      +0et2eqNUMfudAAAAfjB8BgkqhkiG9w0BBwagbzBtAgEAMGgGCSqGSIb3DQEHATAeBg1ghkgBZQMEAS4wEQQMCOEWRda
      wAAAAEAAAAAAAAAAAAAAAAEAAACX3/w5h75QAPomfKH+cyEKYU1yccUmBl
      +VSojiG0tdsUk7vcjYXUUboYm3dvqRqX2s4gROM0n
      +Ij8R0/8jYmnHkpvyAFNVRPyETyIKg7k5Z9+5I1c2d3446Jw/moWGGxjH8AEG9h27ytm0hozxDOEi/
      F2ZoXz6w1GDfGUo/2WxCkY0hTyNaw6TM
      +7drTM7yrW4iNVLUM0LX0xnFjIAhLhooWJek6vjQZUAZzB1AjBH8okRtYP8R7AY2W1s/
      hqFBhG0V4142AC0LxsuZbMQrE2SszWUZ0E9Uew7/n0cyX4CMQDR79INyv4ysMByW9kKGGKyba+cCNk1ExMR
      +btBQBmMuB2fMSdmQXe9umX8Ux8XCo4="
  }],
  "QueryId": "1234273f-4a62-4cda-8d98-69615ee8be9b"
}
```

有关更多信息，请参阅 *La AWS ke Formation 开发者指南* 中的 [事务数据操作](#)。

- 有关 API 的详细信息，请参阅 *AWS CLI 命令参考* [GetWorkUnits](#) 中的。

## grant-permissions

以下代码示例演示了如何使用 `grant-permissions`。

## AWS CLI

示例 1：使用 LF-tags 向委托人授予对资源的权限

以下 `grant-permissions` 示例向委托人授予与 LF-Tag 策略匹配的数据库资源的所有权限。

```
aws lakeformation grant-permissions \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "CatalogId": "123456789111",  
  "Principal": {  
    "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-admin"  
  },  
  "Resource": {  
    "LFTagPolicy": {  
      "CatalogId": "123456789111",  
      "ResourceType": "DATABASE",  
      "Expression": [{  
        "TagKey": "usergroup",  
        "TagValues": [  
          "analyst",  
          "developer"  
        ]  
      }]  
    }  
  },  
  "Permissions": [  
    "ALL"  
  ],  
  "PermissionsWithGrantOption": [  
    "ALL"  
  ]  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的授予和撤销数据目录资源的权限](#)。

示例 2：向委托人授予列级权限

以下`grant-permissions`示例向委托人授予选择特定列的权限。

```
aws lakeformation grant-permissions \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "CatalogId": "123456789111",  
  "Principal": {  
    "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-developer"  
  },  
  "Resource": {  
    "TableWithColumns": {  
      "CatalogId": "123456789111",  
      "ColumnNames": ["p_end_date_sk"],  
      "DatabaseName": "tpc",  
      "Name": "dl_tpc_promotion"  
    }  
  },  
  "Permissions": [  
    "SELECT"  
  ],  
  "PermissionsWithGrantOption": []  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的授予和撤销数据目录资源的权限](#)。

示例 3：向委托人授予表权限

以下`grant-permissions`示例向委托人授予对给定数据库中所有表的选择权限。

```
aws lakeformation grant-permissions \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "CatalogId": "123456789111",
```

```
"Principal": {
  "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-developer"
},
"Resource": {
  "Table": {
    "CatalogId": "123456789111",
    "DatabaseName": "tpc",
    "TableWildcard": {}
  }
},
"Permissions": [
  "SELECT"
],
"PermissionsWithGrantOption": []
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的授予和撤销数据目录资源的权限](#)。

示例 4：向委托人授予 LF 标签的权限

以下 `grant-permissions` 示例向委托人授予对 LF-Tags 的关联权限。

```
aws lakeformation grant-permissions \
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{
  "CatalogId": "123456789111",
  "Principal": {
    "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-developer"
  },
  "Resource": {
    "LFTag": {
      "CatalogId": "123456789111",
      "TagKey": "category",
      "TagValues": [
        "private", "public"
      ]
    }
  }
}
```

```
  },
  "Permissions": [
    "ASSOCIATE"
  ],
  "PermissionsWithGrantOption": []
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的授予和撤销数据目录资源的权限](#)。

示例 5：向委托人授予数据位置权限

以下 `grant-permissions` 示例向委托人授予数据定位权限。

```
aws lakeformation grant-permissions \
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{
  "CatalogId": "123456789111",
  "Principal": {
    "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-developer"
  },
  "Resource": {
    "DataLocation": {
      "CatalogId": "123456789111",
      "ResourceArn": "arn:aws:s3:::lf-data-lake-123456789111"
    }
  },
  "Permissions": [
    "DATA_LOCATION_ACCESS"
  ],
  "PermissionsWithGrantOption": []
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的授予和撤销数据目录资源的权限](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GrantPermissions](#)中的。

## list-data-cells-filter

以下代码示例演示了如何使用 list-data-cells-filter。

### AWS CLI

#### 列出数据单元格筛选器

以下list-data-cells-filter示例列出了给定表的数据单元格筛选器。

```
aws lakeformation list-data-cells-filter \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{  
  "MaxResults": 2,  
  "Table": {  
    "CatalogId": "123456789111",  
    "DatabaseName": "tpc",  
    "Name": "dl_tpc_promotion"  
  }  
}
```

输出：

```
{  
  "DataCellsFilters": [{  
    "TableCatalogId": "123456789111",  
    "DatabaseName": "tpc",  
    "TableName": "dl_tpc_promotion",  
    "Name": "developer_promotion",  
    "RowFilter": {  
      "FilterExpression": "p_promo_name='ese'"  
    }  
  },  
  "ColumnNames": [  
    "p_channel_details",  
    "p_start_date_sk",  
    "p_purpose",
```

```

        "p_promo_id",
        "p_promo_name",
        "p_end_date_sk",
        "p_discount_active"
    ]
},
{
    "TableCatalogId": "123456789111",
    "DatabaseName": "tpc",
    "TableName": "dl_tpc_promotion",
    "Name": "developer_promotion_allrows",
    "RowFilter": {
        "FilterExpression": "TRUE",
        "AllRowsWildcard": {}
    },
    "ColumnNames": [
        "p_channel_details",
        "p_start_date_sk",
        "p_promo_name"
    ]
}
],
"NextToken": "2MDA2MTgwNiwibmFub3MiOjE0MDAwMDAwMH19"
}

```

有关更多信息，请参阅 [Lake Formation 开发者指南中的 Lake Formation 中的AWS 数据筛选和单元级安全](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDataCellsFilter](#) 中的。

## list-permissions

以下代码示例演示了如何使用 list-permissions。

### AWS CLI

示例 1：检索资源的主体权限列表

以下 list-permissions 示例返回数据库资源的主体权限列表。

```
aws lakeformation list-permissions \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "CatalogId": "123456789111",
  "ResourceType": "DATABASE",
  "MaxResults": 2
}
```

输出：

```
{
  "PrincipalResourcePermissions": [{
    "Principal": {
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-campaign-manager"
    },
    "Resource": {
      "Database": {
        "CatalogId": "123456789111",
        "Name": "tpc"
      }
    },
    "Permissions": [
      "DESCRIBE"
    ],
    "PermissionsWithGrantOption": []
  }],
  "NextToken":
  "E5S1JDSTZ1eUp6SWpvaU9UQTN0RE0zTXpFeE5Ua3pJbjE5TENKbGVIQnBjbUYwYVc5dUlqcDdJbk5sWTI5dVpITWlP"
}
```

有关更多信息，请参阅 [Lake Formation 开发者指南中的管理AWS Lake Formation 权限](#)。

示例 2：使用数据筛选器检索表的主体权限列表

以下list-permissions示例列出了向委托人授予的带有相关数据筛选器的表的权限。

```
aws lakeformation list-permissions \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "CatalogId": "123456789111",
  "Resource": {
    "Table": {
      "CatalogId": "123456789111",
      "DatabaseName": "tpc",
      "Name": "dl_tpc_customer"
    }
  },
  "IncludeRelated": "TRUE",
  "MaxResults": 10
}
```

输出：

```
{
  "PrincipalResourcePermissions": [{
    "Principal": {
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:role/
Admin"
    },
    "Resource": {
      "Table": {
        "CatalogId": "123456789111",
        "DatabaseName": "customer",
        "Name": "customer_invoice"
      }
    },
    "Permissions": [
      "ALL",
      "ALTER",
      "DELETE",
      "DESCRIBE",
      "DROP",
      "INSERT"
    ],
    "PermissionsWithGrantOption": [
      "ALL",
      "ALTER",
      "DELETE",
      "DESCRIBE",
      "DROP",
      "INSERT"
    ]
  }
]
```

```

    ]
  },
  {
    "Principal": {
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:role/
Admin"
    },
    "Resource": {
      "TableWithColumns": {
        "CatalogId": "123456789111",
        "DatabaseName": "customer",
        "Name": "customer_invoice",
        "ColumnWildcard": {}
      }
    },
    "Permissions": [
      "SELECT"
    ],
    "PermissionsWithGrantOption": [
      "SELECT"
    ]
  },
  {
    "Principal": {
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:role/
Admin"
    },
    "Resource": {
      "DataCellsFilter": {
        "TableCatalogId": "123456789111",
        "DatabaseName": "customer",
        "TableName": "customer_invoice",
        "Name": "dl_us_customer"
      }
    },
    "Permissions": [
      "DESCRIBE",
      "SELECT",
      "DROP"
    ],
    "PermissionsWithGrantOption": []
  }
],
"NextToken": "VyeUFjY291bnRQZXJtaXNzaW9ucyI6ZmFsc2V9"

```

```
}
```

有关更多信息，请参阅 [Lake Formation 开发者指南中的管理AWS Lake Formation 权限](#)。

示例 3：检索 LF 标签的主体权限列表

以下list-permissions示例列出了授予委托人的对 LF 标签的权限。

```
aws lakeformation list-permissions \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{  
  "CatalogId": "123456789111",  
  "Resource": {  
    "LFTag": {  
      "CatalogId": "123456789111",  
      "TagKey": "category",  
      "TagValues": [  
        "private"  
      ]  
    }  
  },  
  "MaxResults": 10  
}
```

输出：

```
{  
  "PrincipalResourcePermissions": [{  
    "Principal": {  
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-  
admin"  
    },  
    "Resource": {  
      "LFTag": {  
        "CatalogId": "123456789111",  
        "TagKey": "category",  
        "TagValues": [  
          "*"   
        ]  
      }  
    }  
  }  
}
```

```

    },
    "Permissions": [
      "DESCRIBE"
    ],
    "PermissionsWithGrantOption": [
      "DESCRIBE"
    ]
  },
  {
    "Principal": {
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-
admin"
    },
    "Resource": {
      "LFTag": {
        "CatalogId": "123456789111",
        "TagKey": "category",
        "TagValues": [
          "*"
        ]
      }
    },
    "Permissions": [
      "ASSOCIATE"
    ],
    "PermissionsWithGrantOption": [
      "ASSOCIATE"
    ]
  }
],
"NextToken": "EJwY21GMGF0XVJanA3SW50cm1pc3Npb25zIjpmYWxzZX0="
}

```

有关更多信息，请参阅 [Lake Formation 开发者指南中的管理AWS Lake Formation 权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPermissions](#)中的。

## list-resources

以下代码示例演示了如何使用 list-resources。

### AWS CLI

列出 Lake Formation 管理的资源

以下 `list-resources` 示例列出了与 Lake Formation 管理的条件相匹配的资源。

```
aws lakeformation list-resources \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "FilterConditionList": [{  
    "Field": "ROLE_ARN",  
    "ComparisonOperator": "CONTAINS",  
    "StringValueList": [  
      "123456789111"  
    ]  
  }],  
  "MaxResults": 10  
}
```

输出：

```
{  
  "ResourceInfoList": [{  
    "ResourceArn": "arn:aws:s3:::lf-data-lake-123456789111",  
    "RoleArn": "arn:aws:iam::123456789111:role/LF-GlueServiceRole",  
    "LastModified": "2022-07-21T02:12:46.669000+00:00"  
  },  
  {  
    "ResourceArn": "arn:aws:s3:::lf-emr-test-123456789111",  
    "RoleArn": "arn:aws:iam::123456789111:role/EMRLFS3Role",  
    "LastModified": "2022-07-29T16:22:03.211000+00:00"  
  }  
]  
}
```

有关更多信息，请参阅 [Lake Formation 开发者指南中的管理AWS Lake Formation 权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResources](#)中的。

## list-transactions

以下代码示例演示了如何使用 `list-transactions`。



## AWS CLI

### 列出所有交易详情

以下`list-transactions`示例返回有关交易及其状态的元数据。

```
aws lakeformation list-transactions \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "CatalogId": "123456789111",  
  "StatusFilter": "ALL",  
  "MaxResults": 3  
}
```

输出：

```
{  
  "Transactions": [{  
    "TransactionId": "1234569f08804cb790d950d4d0fe485e",  
    "TransactionStatus": "committed",  
    "TransactionStartTime": "2022-08-10T14:32:29.220000+00:00",  
    "TransactionEndTime": "2022-08-10T14:32:33.751000+00:00"  
  },  
  {  
    "TransactionId": "12345972ca8347b89825e33c5774aec4",  
    "TransactionStatus": "committed",  
    "TransactionStartTime": "2022-08-10T14:29:04.046000+00:00",  
    "TransactionEndTime": "2022-08-10T14:29:09.681000+00:00"  
  },  
  {  
    "TransactionId": "12345daf6cb047dbba8ad9b0414613b2",  
    "TransactionStatus": "committed",  
    "TransactionStartTime": "2022-08-10T13:56:51.261000+00:00",  
    "TransactionEndTime": "2022-08-10T13:56:51.547000+00:00"  
  }  
],  
  "NextToken": "77X1ebypsI7os+X21hHsZLGNC DK3nNGpwRdFpicS0HgcX1/  
QMoniUAKcpR3kj3ts3PVdMA=="  
}
```

有关更多信息，请参阅 [Lake Formation 开发者指南中的在事务中读取和写入数据AWS 湖](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTransactions](#)中的。

## put-data-lake-settings

以下代码示例演示了如何使用 put-data-lake-settings。

### AWS CLI

设置 AWS Lake Formation 管理的数据湖设置

以下put-data-lake-settings示例设置了数据湖管理员列表和其他数据湖设置。

```
aws lakeformation put-data-lake-settings \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{  
  "DataLakeSettings": {  
    "DataLakeAdmins": [{  
      "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-  
admin"  
    }  
  ],  
  "CreateDatabaseDefaultPermissions": [],  
  "CreateTableDefaultPermissions": [],  
  "TrustedResourceOwners": [],  
  "AllowExternalDataFiltering": true,  
  "ExternalDataFilteringAllowList": [{  
    "DataLakePrincipalIdentifier": "123456789111"  
  }],  
  "AuthorizedSessionTagValueList": ["Amazon EMR"]  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [Lake Formation 开发者指南中的更改数据AWS 湖的默认安全设置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutDataLakeSettings](#)中的。

## register-resource

以下代码示例演示了如何使用 `register-resource`。

### AWS CLI

示例 1：使用服务关联角色注册数据湖存储

以下 `register-resource` 示例使用服务关联角色将资源注册为由 Lake Formation 管理。

```
aws lakeformation register-resource \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "ResourceArn": "arn:aws:s3:::lf-emr-athena-result-123",  
  "UseServiceLinkedRole": true  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [Lake Formation 开发者指南中的向数据湖添加 Amazon S3 位置](#)。AWS

示例 2：使用自定义角色注册数据湖存储

以下 `register-resource` 示例使用自定义角色将资源注册为由 Lake Formation 管理的资源。

```
aws lakeformation register-resource \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "ResourceArn": "arn:aws:s3:::lf-emr-athena-result-123",  
  "UseServiceLinkedRole": false,  
  "RoleArn": "arn:aws:iam::123456789111:role/LF-GlueServiceRole"  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [Lake Formation 开发者指南中的向数据湖添加 Amazon S3 位置](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterResource](#) 中的。

## remove-lf-tags-from-resource

以下代码示例演示了如何使用 `remove-lf-tags-from-resource`。

### AWS CLI

从资源中删除 LF-Tag

以下 `remove-lf-tags-from-resource` 示例删除了与表资源的 LF-Tag 关联。

```
aws lakeformation remove-lf-tags-from-resource \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "CatalogId": "123456789111",  
  "Resource": {  
    "Table": {  
      "CatalogId": "123456789111",  
      "DatabaseName": "tpc",  
      "Name": "dl_tpc_promotion"  
    }  
  },  
  "LFTags": [{  
    "CatalogId": "123456789111",  
    "TagKey": "usergroup",  
    "TagValues": [  
      "developer"  
    ]  
  }  
}]  
}
```

输出：

```
{  
  "Failures": []  
}
```

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的为数据目录资源分配 LF 标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveLfTagsFromResource](#)中的。

## revoke-permissions

以下代码示例演示了如何使用 revoke-permissions。

### AWS CLI

撤消委托人对资源的权限

以下revoke-permissions示例撤消对给定数据库特定表的主体访问权限。

```
aws lakeformation revoke-permissions \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{  
  "CatalogId": "123456789111",  
  "Principal": {  
    "DataLakePrincipalIdentifier": "arn:aws:iam::123456789111:user/lf-developer"  
  },  
  "Resource": {  
    "Table": {  
      "CatalogId": "123456789111",  
      "DatabaseName": "tpc",  
      "Name": "dl_tpc_promotion"  
    }  
  },  
  "Permissions": [  
    "ALL"  
  ],  
  "PermissionsWithGrantOption": []  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的授予和撤消数据目录资源的权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RevokePermissions](#)中的。

## search-databases-by-lf-tags

以下代码示例演示了如何使用 `search-databases-by-lf-tags`。

### AWS CLI

通过 LFTags 搜索数据库资源

以下 `search-databases-by-lf-tags` 示例搜索与 LFTag 表达式匹配的数据库资源。

```
aws lakeformation search-databases-by-lf-tags \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "MaxResults": 1,  
  "CatalogId": "123456789111",  
  "Expression": [{  
    "TagKey": "usergroup",  
    "TagValues": [  
      "developer"  
    ]  
  }]  
}]
```

输出：

```
{  
  "DatabaseList": [{  
    "Database": {  
      "CatalogId": "123456789111",  
      "Name": "tpc"  
    },  
    "LFTags": [{  
      "CatalogId": "123456789111",  
      "TagKey": "usergroup",  
      "TagValues": [  
        "developer"  
      ]  
    }]  
  }]  
}]
```

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的查看 LF-Tag 分配给的资源](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 SearchDatabasesByLfTags](#) 中的。

## search-tables-by-lf-tags

以下代码示例演示了如何使用 search-tables-by-lf-tags。

### AWS CLI

按 LFTags 搜索表格资源

以下 search-tables-by-lf-tags 示例搜索与 LFTag 表达式匹配的表资源。

```
aws lakeformation search-tables-by-lf-tags \  
  --cli-input-json file://input.json
```

input.json 的内容：

```
{  
  "MaxResults": 2,  
  "CatalogId": "123456789111",  
  "Expression": [{  
    "TagKey": "usergroup",  
    "TagValues": [  
      "developer"  
    ]  
  }]  
}
```

输出：

```
{  
  "NextToken": "c2VhcmNoQWxsVGFnc0luVGFibGVzIjpmYWxzZX0=",  
  "TableList": [{  
    "Table": {  
      "CatalogId": "123456789111",  
      "DatabaseName": "tpc",  
      "Name": "dl_tpc_item"  
    },  
    "LFTagOnDatabase": [{  
      "CatalogId": "123456789111",  
      "TagKey": "usergroup",  
      "TagValue": "developer"  
    }]  
  }]  
}
```

```
    "TagValues": [
      "developer"
    ]
  }],
  "LFTagsOnTable": [{
    "CatalogId": "123456789111",
    "TagKey": "usergroup",
    "TagValues": [
      "developer"
    ]
  }],
  "LFTagsOnColumns": [{
    "Name": "i_item_desc",
    "LFTags": [{
      "CatalogId": "123456789111",
      "TagKey": "usergroup",
      "TagValues": [
        "developer"
      ]
    }
  ]
}],
{
  "Name": "i_container",
  "LFTags": [{
    "CatalogId": "123456789111",
    "TagKey": "usergroup",
    "TagValues": [
      "developer"
    ]
  ]
}],
{
  "Name": "i_wholesale_cost",
  "LFTags": [{
    "CatalogId": "123456789111",
    "TagKey": "usergroup",
    "TagValues": [
      "developer"
    ]
  ]
}],
{
  "Name": "i_manufact_id",
  "LFTags": [{
```



```
        "CatalogId": "123456789111",
        "TagKey": "usergroup",
        "TagValues": [
            "developer"
        ]
    }
},
{
    "Name": "i_brand_id",
    "LFTags": [{
        "CatalogId": "123456789111",
        "TagKey": "usergroup",
        "TagValues": [
            "developer"
        ]
    }]
},
{
    "Name": "i_formulation",
    "LFTags": [{
        "CatalogId": "123456789111",
        "TagKey": "usergroup",
        "TagValues": [
            "developer"
        ]
    }]
},
{
    "Name": "i_current_price",
    "LFTags": [{
        "CatalogId": "123456789111",
        "TagKey": "usergroup",
        "TagValues": [
            "developer"
        ]
    }]
},
{
    "Name": "i_size",
    "LFTags": [{
        "CatalogId": "123456789111",
        "TagKey": "usergroup",
        "TagValues": [
            "developer"
        ]
    }]
}
```

```
    ]
  ]]
},
{
  "Name": "i_rec_start_date",
  "LFTags": [{
    "CatalogId": "123456789111",
    "TagKey": "usergroup",
    "TagValues": [
      "developer"
    ]
  }]
},
{
  "Name": "i_manufact",
  "LFTags": [{
    "CatalogId": "123456789111",
    "TagKey": "usergroup",
    "TagValues": [
      "developer"
    ]
  }]
},
{
  "Name": "i_item_sk",
  "LFTags": [{
    "CatalogId": "123456789111",
    "TagKey": "usergroup",
    "TagValues": [
      "developer"
    ]
  }]
},
{
  "Name": "i_manager_id",
  "LFTags": [{
    "CatalogId": "123456789111",
    "TagKey": "usergroup",
    "TagValues": [
      "developer"
    ]
  }]
},
{
```

```
    "Name": "i_item_id",
    "LFTags": [{
      "CatalogId": "123456789111",
      "TagKey": "usergroup",
      "TagValues": [
        "developer"
      ]
    }]
  },
  {
    "Name": "i_class_id",
    "LFTags": [{
      "CatalogId": "123456789111",
      "TagKey": "usergroup",
      "TagValues": [
        "developer"
      ]
    }]
  },
  {
    "Name": "i_class",
    "LFTags": [{
      "CatalogId": "123456789111",
      "TagKey": "usergroup",
      "TagValues": [
        "developer"
      ]
    }]
  },
  {
    "Name": "i_category",
    "LFTags": [{
      "CatalogId": "123456789111",
      "TagKey": "usergroup",
      "TagValues": [
        "developer"
      ]
    }]
  },
  {
    "Name": "i_category_id",
    "LFTags": [{
      "CatalogId": "123456789111",
      "TagKey": "usergroup",
```

```
        "TagValues": [
            "developer"
        ]
    }
},
{
    "Name": "i_brand",
    "LFTags": [{
        "CatalogId": "123456789111",
        "TagKey": "usergroup",
        "TagValues": [
            "developer"
        ]
    }]
},
{
    "Name": "i_units",
    "LFTags": [{
        "CatalogId": "123456789111",
        "TagKey": "usergroup",
        "TagValues": [
            "developer"
        ]
    }]
},
{
    "Name": "i_rec_end_date",
    "LFTags": [{
        "CatalogId": "123456789111",
        "TagKey": "usergroup",
        "TagValues": [
            "developer"
        ]
    }]
},
{
    "Name": "i_color",
    "LFTags": [{
        "CatalogId": "123456789111",
        "TagKey": "usergroup",
        "TagValues": [
            "developer"
        ]
    }]
}
```

```

    },
    {
      "Name": "i_product_name",
      "LFTags": [{
        "CatalogId": "123456789111",
        "TagKey": "usergroup",
        "TagValues": [
          "developer"
        ]
      }]
    }
  ]
}]
}

```

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的查看 LF-Tag 分配给的资源](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SearchTablesByLfTags](#) 中的。

## start-query-planning

以下代码示例演示了如何使用 start-query-planning。

### AWS CLI

#### 处理查询语句

以下 start-query-planning 示例提交了处理查询语句的请求。

```
aws lakeformation start-query-planning \
  --cli-input-json file://input.json
```

input.json 的内容：

```
{
  "QueryPlanningContext": {
    "CatalogId": "012345678901",
    "DatabaseName": "tpc"
  },
  "QueryString": "select * from dl_tpc_household_demographics_gov where
hd_income_band_sk=9"
}
```

输出：

```
{
  "QueryId": "772a273f-4a62-4cda-8d98-69615ee8be9b"
}
```

有关更多信息，请参阅 [Lake Formation 开发者指南中的在事务中读取和写入数据AWS 湖](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartQueryPlanning](#)中的。

## start-transaction

以下代码示例演示了如何使用 start-transaction。

AWS CLI

开始新交易

以下start-transaction示例启动一个新事务并返回其交易 ID。

```
aws lakeformation start-transaction \
  --transaction-type = 'READ_AND_WRITE'
```

输出：

```
{
  "TransactionId": "b014d972ca8347b89825e33c5774aec4"
}
```

有关更多信息，请参阅 [Lake Formation 开发者指南中的在事务中读取和写入数据AWS 湖](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartTransaction](#)中的。

## update-lf-tag

以下代码示例演示了如何使用 update-lf-tag。

AWS CLI

更新 LF-tag 的定义

以下 `update-lf-tag` 示例更新了 LF-tag 的定义。

```
aws lakeformation update-lf-tag \  
  --catalog-id '123456789111' \  
  --tag-key 'usergroup' \  
  --tag-values-to-add ['"admin"]'
```

此命令不生成任何输出。

有关更多信息，请参阅 [La AWS ke Formation 开发者指南中的管理 LF 标签以实现元数据访问控制](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateLfTag](#) 中的。

## update-table-objects

以下代码示例演示了如何使用 `update-table-objects`。

### AWS CLI

修改受管辖表格的对象

以下 `update-table-objects` 示例将提供的 S3 对象添加到指定的受管辖表中。

```
aws lakeformation update-table-objects \  
  --cli-input-json file://input.json
```

`input.json` 的内容：

```
{  
  "CatalogId": "012345678901",  
  "DatabaseName": "tpc",  
  "TableName": "dl_tpc_household_demographics_gov",  
  "TransactionId": "12347a9f75424b9b915f6ff201d2a190",  
  "WriteOperations": [{  
    "AddObject": {  
      "Uri": "s3://lf-data-lake-012345678901/target/  
dl_tpc_household_demographics_gov/run-unnamed-1-part-block-0-r-00000-snappy-  
ff26b17504414fe88b302cd795eabd00.parquet",  
      "ETag": "1234ab1fc50a316b149b4e1f21a73800",  
      "Size": 42200  
    }  
  ]  
}
```

```
}]
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [Lake Formation 开发者指南中的在事务中读取和写入数据AWS 湖](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateTableObjects](#) 中的。

## 使用 Lambda 示例 AWS CLI

以下代码示例向您展示了如何使用 `with Lambda` 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **add-layer-version-permission**

以下代码示例演示了如何使用 `add-layer-version-permission`。

AWS CLI

向图层版本添加权限

以下 `add-layer-version-permission` 示例向指定账户授予使用图层版本 1 的权限 `my-layer`。

```
aws lambda add-layer-version-permission \
  --layer-name my-layer \
```



```
--statement-id xaccount \  
--action lambda:GetLayerVersion \  
--principal 123456789012 \  
--version-number 1
```

输出：

```
{  
  "RevisionId": "35d87451-f796-4a3f-a618-95a3671b0a0c",  
  "Statement":  
  {  
    "Sid": "xaccount",  
    "Effect": "Allow",  
    "Principal": {  
      "AWS": "arn:aws:iam::210987654321:root"  
    },  
    "Action": "lambda:GetLayerVersion",  
    "Resource": "arn:aws:lambda:us-east-2:123456789012:layer:my-layer:1"  
  }  
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 Lambda AWS 层](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AddLayerVersionPermission](#) 中的。

## add-permission

以下代码示例演示了如何使用 add-permission。

### AWS CLI

向现有 Lambda 函数添加权限

以下 add-permission 示例授予 Amazon SNS 服务调用名为 my-function 的函数的权限。

```
aws lambda add-permission \  
  --function-name my-function \  
  --action lambda:InvokeFunction \  
  --statement-id sns \  
  --principal sns.amazonaws.com
```

输出：

```
{
  "Statement":
  {
    "Sid": "sns",
    "Effect": "Allow",
    "Principal": {
      "Service": "sns.amazonaws.com"
    },
    "Action": "lambda:InvokeFunction",
    "Resource": "arn:aws:lambda:us-east-2:123456789012:function:my-function"
  }
}
```

有关更多信息，请参阅 Lambda 开发人员指南中的[AWS 对 Lambda 使用基于资源的策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AddPermission](#) 中的。

## create-alias

以下代码示例演示了如何使用 create-alias。

### AWS CLI

为 Lambda 函数创建别名

以下 create-alias 示例会创建名为 LIVE 的别名，该别名指向 my-function Lambda 函数的版本 1。

```
aws lambda create-alias \
  --function-name my-function \
  --description "alias for live version of function" \
  --function-version 1 \
  --name LIVE
```

输出：

```
{
  "FunctionVersion": "1",
  "Name": "LIVE",
  "AliasArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function:LIVE",
  "RevisionId": "873282ed-4cd3-4dc8-a069-d0c647e470c6",
  "Description": "alias for live version of function"
```

```
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的配置 Lambda AWS 函数别名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateAlias](#) 中的。

## create-event-source-mapping

以下代码示例演示了如何使用 create-event-source-mapping。

### AWS CLI

在事件源和 AWS Lambda 函数之间创建映射

以下 create-event-source-mapping 示例在 SQS 队列和 Lambda my-function a 函数之间创建了映射。

```
aws lambda create-event-source-mapping \  
  --function-name my-function \  
  --batch-size 5 \  
  --event-source-arn arn:aws:sqs:us-west-2:123456789012:mySQSqueue
```

输出：

```
{  
  "UUID": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
  "StateTransitionReason": "USER_INITIATED",  
  "LastModified": 1569284520.333,  
  "BatchSize": 5,  
  "State": "Creating",  
  "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",  
  "EventSourceArn": "arn:aws:sqs:us-west-2:123456789012:mySQSqueue"  
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 Lambda 事件源映射](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateEventSourceMapping](#) 中的。

## create-function

以下代码示例演示了如何使用 create-function。

## AWS CLI

### 创建 Lambda 函数

以下 `create-function` 示例创建一个名为 `my-function` 的 Lambda 函数。

```
aws lambda create-function \  
  --function-name my-function \  
  --runtime nodejs18.x \  
  --zip-file fileb://my-function.zip \  
  --handler my-function.handler \  
  --role arn:aws:iam::123456789012:role/service-role/MyTestFunction-role-tges6bf4
```

`my-function.zip` 的内容：

```
This file is a deployment package that contains your function code and any dependencies.
```

输出：

```
{  
  "TracingConfig": {  
    "Mode": "PassThrough"  
  },  
  "CodeSha256": "PFn4S+er27qk+UuZSTKEQfNKG/XNn7QJs90mJgq6oH8=",  
  "FunctionName": "my-function",  
  "CodeSize": 308,  
  "RevisionId": "873282ed-4cd3-4dc8-a069-d0c647e470c6",  
  "MemorySize": 128,  
  "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",  
  "Version": "$LATEST",  
  "Role": "arn:aws:iam::123456789012:role/service-role/MyTestFunction-role-zgur6bf4",  
  "Timeout": 3,  
  "LastModified": "2023-10-14T22:26:11.234+0000",  
  "Handler": "my-function.handler",  
  "Runtime": "nodejs18.x",  
  "Description": ""  
}
```

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的[配置AWS Lambda 函数选项](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFunction](#)中的。

## delete-alias

以下代码示例演示了如何使用 delete-alias。

### AWS CLI

删除 Lambda 函数别名

以下 delete-alias 示例从 my-function Lambda 函数中删除了名为 LIVE 的别名。

```
aws lambda delete-alias \  
  --function-name my-function \  
  --name LIVE
```

此命令不生成任何输出。

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的配置 Lambda AWS 函数别名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteAlias](#) 中的。

## delete-event-source-mapping

以下代码示例演示了如何使用 delete-event-source-mapping。

### AWS CLI

删除事件源和 AWS Lambda 函数之间的映射

以下 delete-event-source-mapping 示例删除了 SQS 队列和 Lambda my-function a 函数之间的映射。

```
aws lambda delete-event-source-mapping \  
  --uuid a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
```

输出：

```
{  
  "UUID": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
  "StateTransitionReason": "USER_INITIATED",  
  "LastModified": 1569285870.271,  
  "BatchSize": 5,  
  "State": "Deleting",
```

```
"FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
"EventSourceArn": "arn:aws:sqs:us-west-2:123456789012:mySQSqueue"
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 Lambda 事件源映射](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteEventSourceMapping](#) 中的。

## delete-function-concurrency

以下代码示例演示了如何使用 delete-function-concurrency。

### AWS CLI

从函数中删除预留的并发执行限制

以下 delete-function-concurrency 示例从 my-function 函数中删除了预留的并发执行限制。

```
aws lambda delete-function-concurrency \
  --function-name my-function
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 [为 Lambda 函数预留并发](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteFunctionConcurrency](#) 中的。

## delete-function-event-invoke-config

以下代码示例演示了如何使用 delete-function-event-invoke-config。

### AWS CLI

删除异步调用配置

以下 delete-function-event-invoke-config 示例删除指定函数 GREEN 别名的异步调用配置。

```
aws lambda delete-function-event-invoke-config --function-name my-function:GREEN
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteFunctionEventInvokeConfig](#) 中的。

## delete-function

以下代码示例演示了如何使用 delete-function。

### AWS CLI

示例 1：按函数名称删除 Lambda 函数

以下 delete-function 示例删除通过指定函数名称命名为 my-function 的 Lambda 函数。

```
aws lambda delete-function \  
  --function-name my-function
```

此命令不生成任何输出。

示例 2：按函数 ARN 删除 Lambda 函数

以下 delete-function 示例删除通过指定函数 ARN 命名为 my-function 的 Lambda 函数。

```
aws lambda delete-function \  
  --function-name arn:aws:lambda:us-west-2:123456789012:function:my-function
```

此命令不生成任何输出。

示例 3：按部分函数 ARN 删除 Lambda 函数

以下 delete-function 示例删除通过指定函数的部分 ARN 命名为 my-function 的 Lambda 函数。

```
aws lambda delete-function \  
  --function-name 123456789012:function:my-function
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的[配置AWS Lambda 函数选项](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFunction](#)中的。

## delete-layer-version

以下代码示例演示了如何使用 delete-layer-version。

## AWS CLI

删除 Lambda 层的某个版本

以下 `delete-layer-version` 示例删除名为的图层的版本 2 `my-layer`。

```
aws lambda delete-layer-version \  
  --layer-name my-layer \  
  --version-number 2
```

此命令不生成任何输出。

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 Lambda AWS 层](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteLayerVersion](#) 中的。

## `delete-provisioned-concurrency-config`

以下代码示例演示了如何使用 `delete-provisioned-concurrency-config`。

## AWS CLI

删除预置并发配置

以下 `delete-provisioned-concurrency-config` 示例删除了指定函数 GREEN 别名的预置并发配置。

```
aws lambda delete-provisioned-concurrency-config \  
  --function-name my-function \  
  --qualifier GREEN
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteProvisionedConcurrencyConfig](#) 中的。

## `get-account-settings`

以下代码示例演示了如何使用 `get-account-settings`。

## AWS CLI

在某个地区检索有关您的账户的 AWS 详细信息

以下 `get-account-settings` 示例展示了账户的 Lambda 限制和使用信息。



```
aws lambda get-account-settings
```

输出：

```
{
  "AccountLimit": {
    "CodeSizeUnzipped": 262144000,
    "UnreservedConcurrentExecutions": 1000,
    "ConcurrentExecutions": 1000,
    "CodeSizeZipped": 52428800,
    "TotalCodeSize": 80530636800
  },
  "AccountUsage": {
    "FunctionCount": 4,
    "TotalCodeSize": 9426
  }
}
```

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 [AWS Lambda 限制](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetAccountSettings](#) 中的。

## get-alias

以下代码示例演示了如何使用 get-alias。

### AWS CLI

检索关于函数别名的详细信息

以下 get-alias 示例展示了 my-function Lambda 函数中名为 LIVE 的别名的详细信息。

```
aws lambda get-alias \
  --function-name my-function \
  --name LIVE
```

输出：

```
{
  "FunctionVersion": "3",
  "Name": "LIVE",
```

```
"AliasArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function:LIVE",
"RevisionId": "594f41fb-b85f-4c20-95c7-6ca5f2a92c93",
"Description": "alias for live version of function"
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的配置 Lambda AWS 函数别名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetAlias](#) 中的。

## get-event-source-mapping

以下代码示例演示了如何使用 get-event-source-mapping。

### AWS CLI

检索有关事件源映射的详细信息

以下 get-event-source-mapping 示例显示了 SQS 队列和 Lambda my-function a 函数之间映射的详细信息。

```
aws lambda get-event-source-mapping \
  --uuid "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
```

输出：

```
{
  "UUID": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "StateTransitionReason": "USER_INITIATED",
  "LastModified": 1569284520.333,
  "BatchSize": 5,
  "State": "Enabled",
  "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
  "EventSourceArn": "arn:aws:sqs:us-west-2:123456789012:mySQSqueue"
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 Lambda 事件源映射](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetEventSourceMapping](#) 中的。

## get-function-concurrency

以下代码示例演示了如何使用 get-function-concurrency。

## AWS CLI

### 查看函数的预留并发设置

以下 `get-function-concurrency` 示例检索了指定函数的预留并发设置。

```
aws lambda get-function-concurrency \  
  --function-name my-function
```

输出：

```
{  
  "ReservedConcurrentExecutions": 250  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFunctionConcurrency](#)中的。

## get-function-configuration

以下代码示例演示了如何使用 `get-function-configuration`。

## AWS CLI

### 检索 Lambda 函数的版本特定设置

以下 `get-function-configuration` 示例展示了 `my-function` 函数版本 2 的设置。

```
aws lambda get-function-configuration \  
  --function-name my-function:2
```

输出：

```
{  
  "FunctionName": "my-function",  
  "LastModified": "2019-09-26T20:28:40.438+0000",  
  "RevisionId": "e52502d4-9320-4688-9cd6-152a6ab7490d",  
  "MemorySize": 256,  
  "Version": "2",  
  "Role": "arn:aws:iam::123456789012:role/service-role/my-function-role-uy3l9qyq",  
  "Timeout": 3,  
  "Runtime": "nodejs10.x",
```

```
"TracingConfig": {
  "Mode": "PassThrough"
},
"CodeSha256": "5tT2qgzYUHaqWR716pZ2dpkn/0J1FrzJmLKidWoaCgk=",
"Description": "",
"VpcConfig": {
  "SubnetIds": [],
  "VpcId": "",
  "SecurityGroupIds": []
},
"CodeSize": 304,
"FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function:2",
"Handler": "index.handler"
}
```

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的[配置AWS Lambda 函数选项](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFunctionConfiguration](#)中的。

## get-function-event-invoke-config

以下代码示例演示了如何使用 get-function-event-invoke-config。

### AWS CLI

#### 查看异步调用配置

以下get-function-event-invoke-config示例检索指定函数BLUE别名的异步调用配置。

```
aws lambda get-function-event-invoke-config \
  --function-name my-function:BLUE
```

输出：

```
{
  "LastModified": 1577824396.653,
  "FunctionArn": "arn:aws:lambda:us-east-2:123456789012:function:my-
function:BLUE",
  "MaximumRetryAttempts": 0,
  "MaximumEventAgeInSeconds": 3600,
  "DestinationConfig": {
    "OnSuccess": {},
    "OnFailure": {
```

```
        "Destination": "arn:aws:sqs:us-east-2:123456789012:failed-invocations"
      }
    }
  }
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFunctionEventInvokeConfig](#)中的。

## get-function

以下代码示例演示了如何使用 get-function。

### AWS CLI

检索有关函数的信息

以下 get-function 示例显示有关 my-function 函数的信息：

```
aws lambda get-function \
  --function-name my-function
```

输出：

```
{
  "Concurrency": {
    "ReservedConcurrentExecutions": 100
  },
  "Code": {
    "RepositoryType": "S3",
    "Location": "https://awslambda-us-west-2-tasks.s3.us-west-2.amazonaws.com/
snapshots/123456789012/my-function..."
  },
  "Configuration": {
    "TracingConfig": {
      "Mode": "PassThrough"
    },
    "Version": "$LATEST",
    "CodeSha256": "5tT2qgzYUHoqwR616pZ2dpkn/0J1FrzJmlKidWaaCgk=",
    "FunctionName": "my-function",
    "VpcConfig": {
      "SubnetIds": [],
      "VpcId": "",
      "SecurityGroupIds": []
    }
  }
}
```

```
    },
    "MemorySize": 128,
    "RevisionId": "28f0fb31-5c5c-43d3-8955-03e76c5c1075",
    "CodeSize": 304,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
    "Handler": "index.handler",
    "Role": "arn:aws:iam::123456789012:role/service-role/helloWorldPython-role-uy3l9qyq",
    "Timeout": 3,
    "LastModified": "2019-09-24T18:20:35.054+0000",
    "Runtime": "nodejs10.x",
    "Description": ""
  }
}
```

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的[配置AWS Lambda 函数选项](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFunction](#)中的。

## get-layer-version-by-arn

以下代码示例演示了如何使用 `get-layer-version-by-arn`。

### AWS CLI

检索有关 Lambda 层版本的信息

以下`get-layer-version-by-arn`示例显示有关具有指定 Amazon 资源名称 (ARN) 的层版本的信息。

```
aws lambda get-layer-version-by-arn \
  --arn "arn:aws:lambda:us-west-2:123456789012:layer:AWSLambda-Python311-SciPy1x:2"
```

输出：

```
{
  "LayerVersionArn": "arn:aws:lambda:us-west-2:123456789012:layer:AWSLambda-Python311-SciPy1x:2",
  "Description": "AWS Lambda SciPy layer for Python 3.11 (scipy-1.1.0, numpy-1.15.4) https://github.com/scipy/scipy/releases/tag/v1.1.0 https://github.com/numpy/numpy/releases/tag/v1.15.4",
```

```
"CreateDate": "2023-10-12T10:09:38.398+0000",
"LayerArn": "arn:aws:lambda:us-west-2:123456789012:layer:AWSLambda-Python311-
SciPy1x",
"Content": {
  "CodeSize": 41784542,
  "CodeSha256": "GGmv8ocUw4cly0T8HL0Vx/f5V4RmSCGNjDIslY4VskM=",
  "Location": "https://awslambda-us-west-2-layers.s3.us-west-2.amazonaws.com/
snapshots/123456789012/..."
},
"Version": 2,
"CompatibleRuntimes": [
  "python3.11"
],
"LicenseInfo": "SciPy: https://github.com/scipy/scipy/blob/main/LICENSE.txt,
NumPy: https://github.com/numpy/numpy/blob/main/LICENSE.txt"
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 Lambda AWS 层](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetLayerVersionByArn](#) 中的。

## get-layer-version-policy

以下代码示例演示了如何使用 `get-layer-version-policy`。

### AWS CLI

检索 Lambda 层版本的权限策略

以下 `get-layer-version-policy` 示例显示名为 `my-layer` 的图层的版本 1 的策略信息。

```
aws lambda get-layer-version-policy \
  --layer-name my-layer \
  --version-number 1
```

输出：

```
{
  "Policy": {
    "Version": "2012-10-17",
    "Id": "default",
    "Statement":
```

```

    [
      {
        "Sid": "xaccount",
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::123456789012:root"},
        "Action": "lambda:GetLayerVersion",
        "Resource": "arn:aws:lambda:us-west-2:123456789012:layer:my-layer:1"
      }
    ],
    "RevisionId": "c68f21d2-cbf0-4026-90f6-1375ee465cd0"
  }
}

```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 Lambda AWS 层](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetLayerVersionPolicy](#) 中的。

## get-layer-version

以下代码示例演示了如何使用 `get-layer-version`。

### AWS CLI

检索有关 Lambda 层版本的信息

以下 `get-layer-version` 示例显示名为的图层的版本 1 的信息 `my-layer`。

```

aws lambda get-layer-version \
  --layer-name my-layer \
  --version-number 1

```

输出：

```

{
  "Content": {
    "Location": "https://awslambda-us-east-2-layers.s3.us-east-2.amazonaws.com/snapshots/123456789012/my-layer-4aaa2fbb-ff77-4b0a-ad92-5b78a716a96a?versionId=27iWyA73cCAYqyH...",
    "CodeSha256": "tv9jJ0+rPbXUUXuRKi7CwHzKtLDkDRJLB3cC3Z/ouXo=",
    "CodeSize": 169
  },
  "LayerArn": "arn:aws:lambda:us-east-2:123456789012:layer:my-layer",
}

```



```
"LayerVersionArn": "arn:aws:lambda:us-east-2:123456789012:layer:my-layer:1",
"Description": "My Python layer",
"CreateDate": "2018-11-14T23:03:52.894+0000",
"Version": 1,
"LicenseInfo": "MIT",
"CompatibleRuntimes": [
  "python3.10",
  "python3.11"
]
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 Lambda AWS 层](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetLayerVersion](#) 中的。

## get-policy

以下代码示例演示了如何使用 get-policy。

### AWS CLI

检索函数、版本或别名的基于资源的 IAM 策略

以下 get-policy 示例展示了有关 my-function Lambda 函数的策略信息。

```
aws lambda get-policy \
  --function-name my-function
```

输出：

```
{
  "Policy": {
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
      {
        "Sid": "iot-events",
        "Effect": "Allow",
        "Principal": {"Service": "iotevents.amazonaws.com"},
        "Action": "lambda:InvokeFunction",
        "Resource": "arn:aws:lambda:us-west-2:123456789012:function:my-
function"
```

```
    }
  ]
},
"RevisionId": "93017fc9-59cb-41dc-901b-4845ce4bf668"
}
```

有关更多信息，请参阅 Lambda 开发人员指南中的[AWS 对 Lambda 使用基于资源的策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetPolicy](#)中的。

## get-provisioned-concurrency-config

以下代码示例演示了如何使用 `get-provisioned-concurrency-config`。

### AWS CLI

#### 查看预置并发配置

以下 `get-provisioned-concurrency-config` 示例展示了指定函数 BLUE 别名的预置并发配置的详细信息。

```
aws lambda get-provisioned-concurrency-config \
  --function-name my-function \
  --qualifier BLUE
```

输出：

```
{
  "RequestedProvisionedConcurrentExecutions": 100,
  "AvailableProvisionedConcurrentExecutions": 100,
  "AllocatedProvisionedConcurrentExecutions": 100,
  "Status": "READY",
  "LastModified": "2019-12-31T20:28:49+0000"
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetProvisionedConcurrencyConfig](#)中的。

## invoke

以下代码示例演示了如何使用 `invoke`。

## AWS CLI

### 示例 1：同步调用 Lambda 函数

以下 `invoke` 示例同步调用该 `my-function` 函数。如果您使用的是 AWS CLI 版本 2，则该 `cli-binary-format` 选项为必填项。有关更多信息，请参阅《AWS 命令行界面版本 2 的用户指南》中 [AWS CLI 支持的全局命令行选项](#)。

```
aws lambda invoke \  
  --function-name my-function \  
  --cli-binary-format raw-in-base64-out \  
  --payload '{ "name": "Bob" }' \  
  response.json
```

输出：

```
{  
  "ExecutedVersion": "$LATEST",  
  "StatusCode": 200  
}
```

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 [同步调用](#)。

### 示例 2：异步调用 Lambda 函数

以下 `invoke` 示例异步调用该 `my-function` 函数。如果您使用的是 AWS CLI 版本 2，则该 `cli-binary-format` 选项为必填项。有关更多信息，请参阅《AWS 命令行界面版本 2 的用户指南》中 [AWS CLI 支持的全局命令行选项](#)。

```
aws lambda invoke \  
  --function-name my-function \  
  --invocation-type Event \  
  --cli-binary-format raw-in-base64-out \  
  --payload '{ "name": "Bob" }' \  
  response.json
```

输出：

```
{  
  "StatusCode": 202  
}
```

```
}
```

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的[异步调用](#)。

- 有关 API 详细信息，请参阅《AWS CLI Command Reference》中的[Invoke](#)。

## list-aliases

以下代码示例演示了如何使用 `list-aliases`。

### AWS CLI

检索 Lambda 函数的别名列表

以下 `list-aliases` 示例显示了 Lambda `my-function a` 函数的别名列表。

```
aws lambda list-aliases \  
  --function-name my-function
```

输出：

```
{  
  "Aliases": [  
    {  
      "AliasArn": "arn:aws:lambda:us-west-2:123456789012:function:my-  
function:BETA",  
      "RevisionId": "a410117f-ab16-494e-8035-7e204bb7933b",  
      "FunctionVersion": "2",  
      "Name": "BETA",  
      "Description": "alias for beta version of function"  
    },  
    {  
      "AliasArn": "arn:aws:lambda:us-west-2:123456789012:function:my-  
function:LIVE",  
      "RevisionId": "21d40116-f8b1-40ba-9360-3ea284da1bb5",  
      "FunctionVersion": "1",  
      "Name": "LIVE",  
      "Description": "alias for live version of function"  
    }  
  ]  
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的配置 Lambda AWS 函数别名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAliases](#) 中的。

## list-event-source-mappings

以下代码示例演示了如何使用 list-event-source-mappings。

### AWS CLI

列出函数的事件源映射

以下 list-event-source-mappings 示例显示了 Lambda my-function a 函数的事件源映射列表。

```
aws lambda list-event-source-mappings \
  --function-name my-function
```

输出：

```
{
  "EventSourceMappings": [
    {
      "UUID": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "StateTransitionReason": "USER_INITIATED",
      "LastModified": 1569284520.333,
      "BatchSize": 5,
      "State": "Enabled",
      "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function",
      "EventSourceArn": "arn:aws:sqs:us-west-2:123456789012:mySQSqueue"
    }
  ]
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 Lambda 事件源映射](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListEventSourceMappings](#) 中的。

## list-function-event-invoke-configs

以下代码示例演示了如何使用 list-function-event-invoke-configs。

## AWS CLI

### 查看异步调用配置列表

以下 `list-function-event-invoke-configs` 示例列出了指定函数的异步调用配置。

```
aws lambda list-function-event-invoke-configs \  
  --function-name my-function
```

输出：

```
{  
  "FunctionEventInvokeConfigs": [  
    {  
      "LastModified": 1577824406.719,  
      "FunctionArn": "arn:aws:lambda:us-east-2:123456789012:function:my-  
function:GREEN",  
      "MaximumRetryAttempts": 2,  
      "MaximumEventAgeInSeconds": 1800  
    },  
    {  
      "LastModified": 1577824396.653,  
      "FunctionArn": "arn:aws:lambda:us-east-2:123456789012:function:my-  
function:BLUE",  
      "MaximumRetryAttempts": 0,  
      "MaximumEventAgeInSeconds": 3600  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListFunctionEventInvokeConfigs](#) 中的。

## list-functions

以下代码示例演示了如何使用 `list-functions`。

### AWS CLI

#### 检索 Lambda 函数列表

以下 `list-functions` 示例显示当前用户所有函数的列表。

```
aws lambda list-functions
```

输出：

```
{
  "Functions": [
    {
      "TracingConfig": {
        "Mode": "PassThrough"
      },
      "Version": "$LATEST",
      "CodeSha256": "dBG9m8SGdmlEjw/JYXlhhvCrAv5TxvXsbl/RMr0fT/I=",
      "FunctionName": "helloworld",
      "MemorySize": 128,
      "RevisionId": "1718e831-badf-4253-9518-d0644210af7b",
      "CodeSize": 294,
      "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:helloworld",
      "Handler": "helloworld.handler",
      "Role": "arn:aws:iam::123456789012:role/service-role/MyTestFunction-role-zgur6bf4",
      "Timeout": 3,
      "LastModified": "2023-09-23T18:32:33.857+0000",
      "Runtime": "nodejs18.x",
      "Description": ""
    },
    {
      "TracingConfig": {
        "Mode": "PassThrough"
      },
      "Version": "$LATEST",
      "CodeSha256": "sU0cJ2/h0ZevwV/1TxCuQqK3gDZP3i8gUoqUUVRmY6E=",
      "FunctionName": "my-function",
      "VpcConfig": {
        "SubnetIds": [],
        "VpcId": "",
        "SecurityGroupIds": []
      },
      "MemorySize": 256,
      "RevisionId": "93017fc9-59cb-41dc-901b-4845ce4bf668",
      "CodeSize": 266,
      "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
    }
  ]
}
```

```
    "Handler": "index.handler",
    "Role": "arn:aws:iam::123456789012:role/service-role/helloWorldPython-
role-uy3l9qq",
    "Timeout": 3,
    "LastModified": "2023-10-01T16:47:28.490+0000",
    "Runtime": "nodejs18.x",
    "Description": ""
  },
  {
    "Layers": [
      {
        "CodeSize": 41784542,
        "Arn": "arn:aws:lambda:us-west-2:420165488524:layer:AWSLambda-
Python37-SciPy1x:2"
      },
      {
        "CodeSize": 4121,
        "Arn": "arn:aws:lambda:us-
west-2:123456789012:layer:pythonLayer:1"
      }
    ],
    "TracingConfig": {
      "Mode": "PassThrough"
    },
    "Version": "$LATEST",
    "CodeSha256": "ZQukCqxtkqFgyF2cU41Avj99TKQ/hNihPtDtRcc08mI=",
    "FunctionName": "my-python-function",
    "VpcConfig": {
      "SubnetIds": [],
      "VpcId": "",
      "SecurityGroupIds": []
    },
    "MemorySize": 128,
    "RevisionId": "80b4eabc-acf7-4ea8-919a-e874c213707d",
    "CodeSize": 299,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
python-function",
    "Handler": "lambda_function.lambda_handler",
    "Role": "arn:aws:iam::123456789012:role/service-role/my-python-function-
role-z5g7dr6n",
    "Timeout": 3,
    "LastModified": "2023-10-01T19:40:41.643+0000",
    "Runtime": "python3.11",
    "Description": ""
  }
]
```



```
    }  
  ]  
}
```

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的[配置AWS Lambda 函数选项](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFunctions](#)中的。

## list-layer-versions

以下代码示例演示了如何使用 `list-layer-versions`。

### AWS CLI

列出 AWS Lambda 层的版本

以下`list-layers-versions`示例显示名为的层的版本的相关信息`my-layer`。

```
aws lambda list-layer-versions \  
  --layer-name my-layer
```

输出：

```
{  
  "Layers": [  
    {  
      "LayerVersionArn": "arn:aws:lambda:us-east-2:123456789012:layer:my-  
layer:2",  
      "Version": 2,  
      "Description": "My layer",  
      "CreateDate": "2023-11-15T00:37:46.592+0000",  
      "CompatibleRuntimes": [  
        "python3.10",  
        "python3.11"  
      ]  
    }  
  ]  
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 Lambda AWS 层](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListLayerVersions](#)中的。

## list-layers

以下代码示例演示了如何使用 `list-layers`。

### AWS CLI

列出与函数运行时兼容的图层

以下 `list-layers` 示例显示了与 Python 3.11 运行时兼容的图层的相关信息。

```
aws lambda list-layers \  
  --compatible-runtime python3.11
```

输出：

```
{  
  "Layers": [  
    {  
      "LayerName": "my-layer",  
      "LayerArn": "arn:aws:lambda:us-east-2:123456789012:layer:my-layer",  
      "LatestMatchingVersion": {  
        "LayerVersionArn": "arn:aws:lambda:us-east-2:123456789012:layer:my-layer:2",  
        "Version": 2,  
        "Description": "My layer",  
        "CreateDate": "2023-11-15T00:37:46.592+0000",  
        "CompatibleRuntimes": [  
          "python3.10",  
          "python3.11"  
        ]  
      }  
    }  
  ]  
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 Lambda AWS 层](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListLayers](#) 中的。

## list-provisioned-concurrency-configs

以下代码示例演示了如何使用 `list-provisioned-concurrency-configs`。

## AWS CLI

### 获取预置并发配置列表

以下 `list-provisioned-concurrency-configs` 示例列出了指定函数的预置并发配置。

```
aws lambda list-provisioned-concurrency-configs \  
  --function-name my-function
```

输出：

```
{  
  "ProvisionedConcurrencyConfigs": [  
    {  
      "FunctionArn": "arn:aws:lambda:us-east-2:123456789012:function:my-  
function:GREEN",  
      "RequestedProvisionedConcurrentExecutions": 100,  
      "AvailableProvisionedConcurrentExecutions": 100,  
      "AllocatedProvisionedConcurrentExecutions": 100,  
      "Status": "READY",  
      "LastModified": "2019-12-31T20:29:00+0000"  
    },  
    {  
      "FunctionArn": "arn:aws:lambda:us-east-2:123456789012:function:my-  
function:BLUE",  
      "RequestedProvisionedConcurrentExecutions": 100,  
      "AvailableProvisionedConcurrentExecutions": 100,  
      "AllocatedProvisionedConcurrentExecutions": 100,  
      "Status": "READY",  
      "LastModified": "2019-12-31T20:28:49+0000"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListProvisionedConcurrencyConfigs](#)中的。

## list-tags

以下代码示例演示了如何使用 `list-tags`。

## AWS CLI

### 检索 Lambda 函数的标签列表

以下 `list-tags` 示例展示了附加到 `my-function` Lambda 函数的标签。

```
aws lambda list-tags \  
  --resource arn:aws:lambda:us-west-2:123456789012:function:my-function
```

输出：

```
{  
  "Tags": {  
    "Category": "Web Tools",  
    "Department": "Sales"  
  }  
}
```

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的[标记 Lambda 函数](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTags](#) 中的。

## list-versions-by-function

以下代码示例演示了如何使用 `list-versions-by-function`。

### AWS CLI

检索函数的版本列表

以下 `list-versions-by-function` 示例展示了 `my-function` Lambda 函数的版本列表。

```
aws lambda list-versions-by-function \  
  --function-name my-function
```

输出：

```
{  
  "Versions": [  
    {  
      "TracingConfig": {  
        "Mode": "PassThrough"  
      },  
      "Version": "$LATEST",  
      "CodeSha256": "sU0cJ2/h0ZevwV/1TxCuQqK3gDZP3i8gUoqUUVRmY6E=",  
      "FunctionName": "my-function",
```

```
    "VpcConfig": {
      "SubnetIds": [],
      "VpcId": "",
      "SecurityGroupIds": []
    },
    "MemorySize": 256,
    "RevisionId": "93017fc9-59cb-41dc-901b-4845ce4bf668",
    "CodeSize": 266,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:$LATEST",
    "Handler": "index.handler",
    "Role": "arn:aws:iam::123456789012:role/service-role/helloWorldPython-
role-uy3l9qyq",
    "Timeout": 3,
    "LastModified": "2019-10-01T16:47:28.490+0000",
    "Runtime": "nodejs10.x",
    "Description": ""
  },
  {
    "TracingConfig": {
      "Mode": "PassThrough"
    },
    "Version": "1",
    "CodeSha256": "5tT2qgzYUHoqwR616pZ2dpkn/0J1FrzJmlKidWaaCgk=",
    "FunctionName": "my-function",
    "VpcConfig": {
      "SubnetIds": [],
      "VpcId": "",
      "SecurityGroupIds": []
    },
    "MemorySize": 256,
    "RevisionId": "949c8914-012e-4795-998c-e467121951b1",
    "CodeSize": 304,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:1",
    "Handler": "index.handler",
    "Role": "arn:aws:iam::123456789012:role/service-role/helloWorldPython-
role-uy3l9qyq",
    "Timeout": 3,
    "LastModified": "2019-09-26T20:28:40.438+0000",
    "Runtime": "nodejs10.x",
    "Description": "new version"
  },
  {
```

```
    "TracingConfig": {
      "Mode": "PassThrough"
    },
    "Version": "2",
    "CodeSha256": "sU0cJ2/h0ZevwV/1TxCuQqK3gDZP3i8gUoqUUVRmY6E=",
    "FunctionName": "my-function",
    "VpcConfig": {
      "SubnetIds": [],
      "VpcId": "",
      "SecurityGroupIds": []
    },
    "MemorySize": 256,
    "RevisionId": "cd669f21-0f3d-4e1c-9566-948837f2e2ea",
    "CodeSize": 266,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:2",
    "Handler": "index.handler",
    "Role": "arn:aws:iam::123456789012:role/service-role/helloWorldPython-
role-uy3l9qq",
    "Timeout": 3,
    "LastModified": "2019-10-01T16:47:28.490+0000",
    "Runtime": "nodejs10.x",
    "Description": "newer version"
  }
]
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的配置 Lambda AWS 函数别名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListVersionsByFunction](#) 中的。

## publish-layer-version

以下代码示例演示了如何使用 `publish-layer-version`。

### AWS CLI

#### 创建 Lambda 层版本

以下 `publish-layer-version` 示例创建了一个新的 Python 库层版本。该命令检索在指定 S3 存储桶 `layer.zip` 中命名的文件的图层内容。

```
aws lambda publish-layer-version \
```

```
--layer-name my-layer \  
--description "My Python layer" \  
--license-info "MIT" \  
--content S3Bucket=lambda-layers-us-west-2-123456789012,S3Key=layer.zip \  
--compatible-runtimes python3.10 python3.11
```

输出：

```
{  
  "Content": {  
    "Location": "https://awslambda-us-west-2-layers.s3.us-west-2.amazonaws.com/  
snapshots/123456789012/my-layer-4aaa2fbb-ff77-4b0a-ad92-5b78a716a96a?  
versionId=27iWyA73cCAYqyH...",  
    "CodeSha256": "tv9jJ0+rPbXUUXuRKi7CwHzKtLDkDRJLB3cC3Z/ouXo=",  
    "CodeSize": 169  
  },  
  "LayerArn": "arn:aws:lambda:us-west-2:123456789012:layer:my-layer",  
  "LayerVersionArn": "arn:aws:lambda:us-west-2:123456789012:layer:my-layer:1",  
  "Description": "My Python layer",  
  "CreateDate": "2023-11-14T23:03:52.894+0000",  
  "Version": 1,  
  "LicenseInfo": "MIT",  
  "CompatibleRuntimes": [  
    "python3.10",  
    "python3.11"  
  ]  
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 Lambda AWS 层](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PublishLayerVersion](#) 中的。

## publish-version

以下代码示例演示了如何使用 `publish-version`。

### AWS CLI

发布函数的新版本

以下 `publish-version` 示例发布了 `my-function` Lambda 函数的新版本。

```
aws lambda publish-version \  

```

```
--function-name my-function
```

输出：

```
{
  "TracingConfig": {
    "Mode": "PassThrough"
  },
  "CodeSha256": "dBG9m8SGdmlEjw/JYXlhhvCrAv5TxvXsbl/RMr0fT/I=",
  "FunctionName": "my-function",
  "CodeSize": 294,
  "RevisionId": "f31d3d39-cc63-4520-97d4-43cd44c94c20",
  "MemorySize": 128,
  "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function:3",
  "Version": "2",
  "Role": "arn:aws:iam::123456789012:role/service-role/MyTestFunction-role-zgur6bf4",
  "Timeout": 3,
  "LastModified": "2019-09-23T18:32:33.857+0000",
  "Handler": "my-function.handler",
  "Runtime": "nodejs10.x",
  "Description": ""
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的配置 Lambda AWS 函数别名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PublishVersion](#) 中的。

## put-function-concurrency

以下代码示例演示了如何使用 `put-function-concurrency`。

### AWS CLI

#### 配置函数的预留并发限制

以下 `put-function-concurrency` 示例为 `my-function` 函数配置了 100 个预留并发执行。

```
aws lambda put-function-concurrency \
  --function-name my-function \
  --reserved-concurrent-executions 100
```



输出：

```
{
  "ReservedConcurrentExecutions": 100
}
```

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的[为 Lambda 函数预留并发](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutFunctionConcurrency](#)中的。

## put-function-event-invoke-config

以下代码示例演示了如何使用 `put-function-event-invoke-config`。

### AWS CLI

为异步调用配置错误处理

以下`put-function-event-invoke-config`示例将最长事件持续时间设置为一小时，并禁止对指定函数进行重试。

```
aws lambda put-function-event-invoke-config \
  --function-name my-function \
  --maximum-event-age-in-seconds 3600 \
  --maximum-retry-attempts 0
```

输出：

```
{
  "LastModified": 1573686021.479,
  "FunctionArn": "arn:aws:lambda:us-east-2:123456789012:function:my-function:
$LATEST",
  "MaximumRetryAttempts": 0,
  "MaximumEventAgeInSeconds": 3600,
  "DestinationConfig": {
    "OnSuccess": {},
    "OnFailure": {}
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutFunctionEventInvokeConfig](#)中的。

## put-provisioned-concurrency-config

以下代码示例演示了如何使用 `put-provisioned-concurrency-config`。

### AWS CLI

#### 分配预置并发

以下 `put-provisioned-concurrency-config` 示例为指定函数的 BLUE 别名分配了 100 个预置并发。

```
aws lambda put-provisioned-concurrency-config \  
  --function-name my-function \  
  --qualifier BLUE \  
  --provisioned-concurrent-executions 100
```

#### 输出：

```
{  
  "Requested ProvisionedConcurrentExecutions": 100,  
  "Allocated ProvisionedConcurrentExecutions": 0,  
  "Status": "IN_PROGRESS",  
  "LastModified": "2019-11-21T19:32:12+0000"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutProvisionedConcurrencyConfig](#)中的。

## remove-layer-version-permission

以下代码示例演示了如何使用 `remove-layer-version-permission`。

### AWS CLI

#### 删除图层版本权限

以下`remove-layer-version-permission`示例删除了账户配置图层版本的权限。

```
aws lambda remove-layer-version-permission \  
  --layer-name my-layer \  
  --statement-id xaccount \  
  --
```

```
--version-number 1
```

此命令不生成任何输出。

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 Lambda AWS 层](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RemoveLayerVersionPermission](#) 中的。

## remove-permission

以下代码示例演示了如何使用 `remove-permission`。

### AWS CLI

从现有 Lambda 函数中删除权限

以下 `remove-permission` 示例删除了调用名为 `my-function` 的函数的权限。

```
aws lambda remove-permission \  
  --function-name my-function \  
  --statement-id sns
```

此命令不生成任何输出。

有关更多信息，请参阅 Lambda 开发人员指南中的 [AWS 对 Lambda 使用基于资源的策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RemovePermission](#) 中的。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

### AWS CLI

将标签添加到现有 Lambda 函数

以下 `tag-resource` 示例向指定的 Lambda 函数添加了键名称为 `DEPARTMENT` 和值为 `Department A` 的标签。

```
aws lambda tag-resource \  
  --resource arn:aws:lambda:us-west-2:123456789012:function:my-function \  
  --tag-key DEPARTMENT --tag-value Department A
```

```
--tags "DEPARTMENT=Department A"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的[标记 Lambda 函数](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从现有 Lambda 函数中删除标签

以下 untag-resource 示例从 my-function Lambda 函数中删除了键名称为 DEPARTMENT 的标签。

```
aws lambda untag-resource \  
  --resource arn:aws:lambda:us-west-2:123456789012:function:my-function \  
  --tag-keys DEPARTMENT
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的[标记 Lambda 函数](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-alias

以下代码示例演示了如何使用 update-alias。

### AWS CLI

更新函数别名

以下 update-alias 示例会更新名为 LIVE 的别名，该别名指向 my-function Lambda 函数的版本 3。

```
aws lambda update-alias \  
  --function-name my-function \  
  --name LIVE \  
  --version 3
```

```
--function-name my-function \  
--function-version 3 \  
--name LIVE
```

输出：

```
{  
  "FunctionVersion": "3",  
  "Name": "LIVE",  
  "AliasArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function:LIVE",  
  "RevisionId": "594f41fb-b85f-4c20-95c7-6ca5f2a92c93",  
  "Description": "alias for live version of function"  
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的配置 Lambda AWS 函数别名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateAlias](#) 中的。

## update-event-source-mapping

以下代码示例演示了如何使用 `update-event-source-mapping`。

### AWS CLI

更新事件源和 AWS Lambda 函数之间的映射

以下 `update-event-source-mapping` 示例将指定映射中的批量大小更新为 8。

```
aws lambda update-event-source-mapping \  
  --uuid "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE" \  
  --batch-size 8
```

输出：

```
{  
  "UUID": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
  "StateTransitionReason": "USER_INITIATED",  
  "LastModified": 1569284520.333,  
  "BatchSize": 8,  
  "State": "Updating",  
  "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
```

```
"EventSourceArn": "arn:aws:sqs:us-west-2:123456789012:mySQSqueue"
}
```

有关更多信息，请参阅 [AWS Lambda 开发人员指南中的 L AWS am bda 事件源映射](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateEventSourceMapping](#) 中的。

## update-function-code

以下代码示例演示了如何使用 update-function-code。

### AWS CLI

#### 更新 Lambda 函数代码

以下 update-function-code 示例将 my-function 函数的未发布 (\$LATEST) 版本的代码替换为指定 zip 文件的内容。

```
aws lambda update-function-code \  
  --function-name my-function \  
  --zip-file fileb://my-function.zip
```

输出：

```
{  
  "FunctionName": "my-function",  
  "LastModified": "2019-09-26T20:28:40.438+0000",  
  "RevisionId": "e52502d4-9320-4688-9cd6-152a6ab7490d",  
  "MemorySize": 256,  
  "Version": "$LATEST",  
  "Role": "arn:aws:iam::123456789012:role/service-role/my-function-role-uy3l9qqq",  
  "Timeout": 3,  
  "Runtime": "nodejs10.x",  
  "TracingConfig": {  
    "Mode": "PassThrough"  
  },  
  "CodeSha256": "5tT2qgzYUHaqwR716pZ2dpkn/0J1FrzJm1KidWoaCgk=",  
  "Description": "",  
  "VpcConfig": {  
    "SubnetIds": [],  
    "VpcId": "",  
    "SecurityGroupIds": []  
  }  
}
```

```
  },
  "CodeSize": 304,
  "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
  "Handler": "index.handler"
}
```

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的[配置AWS Lambda 函数选项](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateFunctionCode](#)中的。

## update-function-configuration

以下代码示例演示了如何使用 update-function-configuration。

### AWS CLI

#### 修改函数的配置

以下 update-function-configuration 示例将 my-function 函数的未发布 (\$LATEST) 版本的内存大小修改为 256 MB。

```
aws lambda update-function-configuration \
  --function-name my-function \
  --memory-size 256
```

输出：

```
{
  "FunctionName": "my-function",
  "LastModified": "2019-09-26T20:28:40.438+0000",
  "RevisionId": "e52502d4-9320-4688-9cd6-152a6ab7490d",
  "MemorySize": 256,
  "Version": "$LATEST",
  "Role": "arn:aws:iam::123456789012:role/service-role/my-function-role-uy3l9qq",
  "Timeout": 3,
  "Runtime": "nodejs10.x",
  "TracingConfig": {
    "Mode": "PassThrough"
  },
  "CodeSha256": "5tT2qgzYUHaqwR716pZ2dpkn/0J1FrzJm1KidWoaCgk=",
  "Description": "",
  "VpcConfig": {
```

```
    "SubnetIds": [],
    "VpcId": "",
    "SecurityGroupIds": []
  },
  "CodeSize": 304,
  "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
  "Handler": "index.handler"
}
```

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的[配置AWS Lambda 函数选项](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateFunctionConfiguration](#)中的。

## update-function-event-invoke-config

以下代码示例演示了如何使用 update-function-event-invoke-config。

### AWS CLI

#### 更新异步调用配置

以下update-function-event-invoke-config示例将失败时目标添加到指定函数的现有异步调用配置中。

```
aws lambda update-function-event-invoke-config \
  --function-name my-function \
  --destination-config '{"OnFailure":{"Destination": "arn:aws:sqs:us-
east-2:123456789012:destination"}}'
```

输出：

```
{
  "LastModified": 1573687896.493,
  "FunctionArn": "arn:aws:lambda:us-east-2:123456789012:function:my-function:
$LATEST",
  "MaximumRetryAttempts": 0,
  "MaximumEventAgeInSeconds": 3600,
  "DestinationConfig": {
    "OnSuccess": {},
    "OnFailure": {
      "Destination": "arn:aws:sqs:us-east-2:123456789012:destination"
    }
  }
}
```



```
}  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateFunctionEventInvokeConfig](#)中的。

## 使用的 License Manager 示例 AWS CLI

以下代码示例向您展示了如何使用与 License Manager AWS Command Line Interface 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **create-license-configuration**

以下代码示例演示了如何使用 create-license-configuration。

#### AWS CLI

##### 示例 1：创建许可证配置

以下create-license-configuration示例创建了硬限制为 10 个内核的许可证配置。

```
aws license-manager create-license-configuration --name my-license-configuration \  
  --license-counting-type Core \  
  --license-count 10 \  
  --license-count-hard-limit
```

输出：

```
{
  "LicenseConfigurationArn": "arn:aws:license-manager:us-
west-2:123456789012:license-configuration:lic-6eb6586f508a786a2ba41EXAMPLE1111"
}
```

## 示例 2：创建许可证配置

以下 `create-license-configuration` 示例创建软限制为 100 个 vCPU 的许可证配置。它使用规则来启用 vCPU 优化。

```
aws license-manager create-license-configuration --name my-license-configuration
--license-counting-type vCPU \
--license-count 100 \
--license-rules "#honorVcpuOptimization=true"
```

输出：

```
{
  "LicenseConfigurationArn": "arn:aws:license-manager:us-
west-2:123456789012:license-configuration:lic-6eb6586f508a786a2ba41EXAMPLE2222"
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateLicenseConfiguration](#) 中的。

## delete-license-configuration

以下代码示例演示了如何使用 `delete-license-configuration`。

### AWS CLI

#### 删除许可证配置

以下 `delete-license-configuration` 示例删除了指定的许可证配置。

```
aws license-manager delete-license-configuration \
--license-configuration-arn arn:aws:license-manager:us-
west-2:123456789012:license-configuration:lic-6eb6586f508a786a2ba4f56c1EXAMPLE
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLicenseConfiguration](#)中的。

## get-license-configuration

以下代码示例演示了如何使用 `get-license-configuration`。

### AWS CLI

获取许可证配置信息

以下`get-license-configuration`示例显示了指定许可证配置的详细信息。

```
aws license-manager get-license-configuration \  
  --license-configuration-arn arn:aws:license-manager:us-  
west-2:123456789012:license-configuration:lic-38b658717b87478aaa7c00883EXAMPLE
```

输出：

```
{  
  "LicenseConfigurationId": "lic-38b658717b87478aaa7c00883EXAMPLE",  
  "LicenseConfigurationArn": "arn:aws:license-manager:us-  
west-2:123456789012:license-configuration:lic-38b658717b87478aaa7c00883EXAMPLE",  
  "Name": "my-license-configuration",  
  "LicenseCountingType": "vCPU",  
  "LicenseRules": [],  
  "LicenseCountHardLimit": false,  
  "ConsumedLicenses": 0,  
  "Status": "AVAILABLE",  
  "OwnerAccountId": "123456789012",  
  "ConsumedLicenseSummaryList": [  
    {  
      "ResourceType": "EC2_INSTANCE",  
      "ConsumedLicenses": 0  
    },  
    {  
      "ResourceType": "EC2_HOST",  
      "ConsumedLicenses": 0  
    },  
    {  
      "ResourceType": "SYSTEMS_MANAGER_MANAGED_INSTANCE",  
      "ConsumedLicenses": 0  
    }  
  ]  
}
```

```
],
  "ManagedResourceSummaryList": [
    {
      "ResourceType": "EC2_INSTANCE",
      "AssociationCount": 0
    },
    {
      "ResourceType": "EC2_HOST",
      "AssociationCount": 0
    },
    {
      "ResourceType": "EC2_AMI",
      "AssociationCount": 2
    },
    {
      "ResourceType": "SYSTEMS_MANAGER_MANAGED_INSTANCE",
      "AssociationCount": 0
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLicenseConfiguration](#)中的。

## get-service-settings

以下代码示例演示了如何使用 get-service-settings。

### AWS CLI

要获取 License Manager 设置

以下get-service-settings示例显示了当前区域中 License Manager 的服务设置。

```
aws license-manager get-service-settings
```

以下显示了禁用跨账户资源发现时的输出示例。

```
{
  "OrganizationConfiguration": {
    "EnableIntegration": false
  },
  "EnableCrossAccountsDiscovery": false
}
```

```
}
```

以下显示了启用跨账户资源发现时的输出示例。

```
{
  "S3BucketArn": "arn:aws:s3::aws-license-manager-service-c22d6279-35c4-47c4-bb",
  "OrganizationConfiguration": {
    "EnableIntegration": true
  },
  "EnableCrossAccountsDiscovery": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetServiceSettings](#)中的。

## list-associations-for-license-configuration

以下代码示例演示了如何使用 `list-associations-for-license-configuration`。

### AWS CLI

获取许可证配置的关联

以下`list-associations-for-license-configuration`示例显示了指定许可证配置的关联的详细信息。

```
aws license-manager list-associations-for-license-configuration \
  --license-configuration-arn arn:aws:license-manager:us-
west-2:123456789012:license-configuration:lic-38b658717b87478aaa7c00883EXAMPLE
```

输出：

```
{
  "LicenseConfigurationAssociations": [
    {
      "ResourceArn": "arn:aws:ec2:us-west-2::image/ami-1234567890abcdef0",
      "ResourceType": "EC2_AMI",
      "ResourceOwnerId": "123456789012",
      "AssociationTime": 1568825118.617
    },
    {
      "ResourceArn": "arn:aws:ec2:us-west-2::image/ami-0abcdef1234567890",
      "ResourceType": "EC2_AMI",

```

```
        "ResourceOwnerId": "123456789012",
        "AssociationTime": 1568825118.946
    }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAssociationsForLicenseConfiguration](#)中的。

## list-license-configurations

以下代码示例演示了如何使用 `list-license-configurations`。

### AWS CLI

示例 1：列出所有许可证配置

以下`list-license-configurations`示例列出了您的所有许可证配置。

```
aws license-manager list-license-configurations
```

输出：

```
{
  "LicenseConfigurations": [
    {
      "LicenseConfigurationId": "lic-6eb6586f508a786a2ba4f56c1EXAMPLE",
      "LicenseConfigurationArn": "arn:aws:license-manager:us-
west-2:123456789012:license-configuration:lic-6eb6586f508a786a2ba4f56c1EXAMPLE",
      "Name": "my-license-configuration",
      "LicenseCountingType": "Core",
      "LicenseRules": [],
      "LicenseCount": 10,
      "LicenseCountHardLimit": true,
      "ConsumedLicenses": 0,
      "Status": "AVAILABLE",
      "OwnerAccountId": "123456789012",
      "ConsumedLicenseSummaryList": [
        {
          "ResourceType": "EC2_INSTANCE",
          "ConsumedLicenses": 0
        }
      ],
    }
  ],
}
```

```

        {
            "ResourceType": "EC2_HOST",
            "ConsumedLicenses": 0
        },
        {
            "ResourceType": "SYSTEMS_MANAGER_MANAGED_INSTANCE",
            "ConsumedLicenses": 0
        }
    ],
    "ManagedResourceSummaryList": [
        {
            "ResourceType": "EC2_INSTANCE",
            "AssociationCount": 0
        },
        {
            "ResourceType": "EC2_HOST",
            "AssociationCount": 0
        },
        {
            "ResourceType": "EC2_AMI",
            "AssociationCount": 0
        },
        {
            "ResourceType": "SYSTEMS_MANAGER_MANAGED_INSTANCE",
            "AssociationCount": 0
        }
    ]
},
{
    ...
}
]
}

```

### 示例 2：列出特定的许可证配置

以下 `list-license-configurations` 示例仅列出了指定的许可证配置。

```

aws license-manager list-license-configurations \
  --license-configuration-arns arn:aws:license-manager:us-
west-2:123456789012:license-configuration:lic-38b658717b87478aaa7c00883EXAMPLE

```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListLicenseConfigurations](#) 中的。

## list-license-specifications-for-resource

以下代码示例演示了如何使用 `list-license-specifications-for-resource`。

### AWS CLI

列出资源的许可证配置

以下 `list-license-specifications-for-resource` 示例列出了与指定的 Amazon 系统映像 (AMI) 关联的许可配置。

```
aws license-manager list-license-specifications-for-resource \  
  --resource-arn arn:aws:ec2:us-west-2::image/ami-1234567890abcdef0
```

输出：

```
{  
  "LicenseConfigurationArn": "arn:aws:license-manager:us-  
west-2:123456789012:license-configuration:lic-38b658717b87478aaa7c00883EXAMPLE"  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListLicenseSpecificationsForResource](#) 中的。

## list-resource-inventory

以下代码示例演示了如何使用 `list-resource-inventory`。

### AWS CLI

在资源清单中列出资源

以下 `list-resource-inventory` 示例列出了使用 Systems Manager 清单管理的资源。

```
aws license-manager list-resource-inventory
```

输出：

```
{  
  "ResourceInventoryList": [  
    {  
      "Platform": "Red Hat Enterprise Linux Server",  
      "ResourceType": "EC2Instance",
```



```

    "PlatformVersion": "7.4",
    "ResourceArn": "arn:aws:ec2:us-west-2:1234567890129:instance/
i-05d3cdfb05bd36376",
    "ResourceId": "i-05d3cdfb05bd36376",
    "ResourceOwningAccountId": "1234567890129"
  },
  {
    "Platform": "Amazon Linux",
    "ResourceType": "EC2Instance",
    "PlatformVersion": "2",
    "ResourceArn": "arn:aws:ec2:us-west-2:1234567890129:instance/
i-0b1d036cfd4594808",
    "ResourceId": "i-0b1d036cfd4594808",
    "ResourceOwningAccountId": "1234567890129"
  },
  {
    "Platform": "Microsoft Windows Server 2019 Datacenter",
    "ResourceType": "EC2Instance",
    "PlatformVersion": "10.0.17763",
    "ResourceArn": "arn:aws:ec2:us-west-2:1234567890129:instance/
i-0cdb3b54a2a8246ad",
    "ResourceId": "i-0cdb3b54a2a8246ad",
    "ResourceOwningAccountId": "1234567890129"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResourceInventory](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出许可证配置的标签

以下`list-tags-for-resource`示例列出了指定许可证配置的标签。

```

aws license-manager list-tags-for-resource \
  --resource-arn arn:aws:license-manager:us-west-2:123456789012:license-
configuration:lic-6eb6586f508a786a2ba4f56c1EXAMPLE

```

输出：

```
{
  "Tags": [
    {
      "Key": "project",
      "Value": "lima"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## list-usage-for-license-configuration

以下代码示例演示了如何使用 list-usage-for-license-configuration。

AWS CLI

列出用于许可证配置的许可证

以下list-usage-for-license-configuration示例列出了有关使用指定许可证配置的许可证的资源的信息。例如，如果许可类型为 vCPU，则任何实例每个 vCPU 消耗一个许可。

```
aws license-manager list-usage-for-license-configuration \
  --license-configuration-arn arn:aws:license-manager:us-
west-2:123456789012:license-configuration:lic-38b658717b87478aaa7c00883EXAMPLE
```

输出：

```
{
  "LicenseConfigurationUsageList": [
    {
      "ResourceArn": "arn:aws:ec2:us-west-2:123456789012:instance/
i-04a636d18e83cfac",
      "ResourceType": "EC2_INSTANCE",
      "ResourceStatus": "running",
      "ResourceOwnerId": "123456789012",
      "AssociationTime": 1570892850.519,
      "ConsumedLicenses": 2
    }
  ]
}
```

```
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUsageForLicenseConfiguration](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

要添加标签，请执行许可证配置

以下tag-resource示例将指定的标签（密钥名称和值）添加到指定的许可证配置中。

```
aws license-manager tag-resource \
  --tags Key=project,Value=lima \
  --resource-arn arn:aws:license-manager:us-west-2:123456789012:license-
configuration:lic-6eb6586f508a786a2ba4f56c1EXAMPLE
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从许可证配置中删除标签

以下untag-resource示例从指定的许可证配置中删除了指定的标签（密钥名称和资源）。

```
aws license-manager untag-resource \
  --tag-keys project \
  --resource-arn arn:aws:license-manager:us-west-2:123456789012:license-
configuration:lic-6eb6586f508a786a2ba4f56c1EXAMPLE
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-license-configuration

以下代码示例演示了如何使用 `update-license-configuration`。

### AWS CLI

#### 更新许可证配置

以下 `update-license-configuration` 示例更新了指定的许可证配置以移除硬限制。

```
aws license-manager update-license-configuration \  
  --no-license-count-hard-limit \  
  --license-configuration-arn arn:aws:license-manager:us-  
west-2:880185128111:license-configuration:lic-6eb6586f508a786a2ba4f56c1EXAMPLE
```

此命令不生成任何输出。

以下 `update-license-configuration` 示例更新了指定的许可证配置以将其状态更改为 `DISABLED`。

```
aws license-manager update-license-configuration \  
  --license-configuration-status DISABLED  
  --license-configuration-arn arn:aws:license-manager:us-  
west-2:880185128111:license-configuration:lic-6eb6586f508a786a2ba4f56c1EXAMPLE
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateLicenseConfiguration](#) 中的。

## update-license-specifications-for-resource

以下代码示例演示了如何使用 `update-license-specifications-for-resource`。

### AWS CLI

#### 更新资源的许可证配置

以下 `update-license-specifications-for-resource` 示例通过删除一个许可证配置并添加另一个许可配置来替换与指定的 Amazon 系统映像 (AMI) 关联的许可配置。

```
aws license-manager update-license-specifications-for-resource \  
  --license-specification-arn arn:aws:license-manager:us-  
west-2:880185128111:license-specification:lic-6eb6586f508a786a2ba4f56c1EXAMPLE
```

```
--resource-arn arn:aws:ec2:us-west-2::image/ami-1234567890abcdef0 \  
--remove-license-specifications LicenseConfigurationArn=arn:aws:license-  
manager:us-west-2:123456789012:license-  
configuration:lic-38b658717b87478aaa7c00883EXAMPLE \  
--add-license-specifications LicenseConfigurationArn=arn:aws:license-manager:us-  
west-2:123456789012:license-configuration:lic-42b6deb06e5399a980d555927EXAMPLE
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLicenseSpecificationsForResource](#)中的。

## update-service-settings

以下代码示例演示了如何使用 `update-service-settings`。

### AWS CLI

#### 更新 License Manager 设置

以下 `update-service-settings` 示例为当前 AWS 区域的 License Manager 启用跨账户资源发现。Amazon S3 存储桶是 Systems Manager 清单所需的资源数据同步。

```
aws license-manager update-service-settings \  
--organization-configuration EnableIntegration=true \  
--enable-cross-accounts-discovery \  
--s3-bucket-arn arn:aws:s3::aws-license-manager-service-abcd1234EXAMPLE
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateServiceSettings](#)中的。

## 使用 Lightsail 示例 AWS CLI

以下代码示例向您展示了如何使用 `with Lightsail` 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## allocate-static-ip

以下代码示例演示了如何使用 `allocate-static-ip`。

AWS CLI

创建静态 IP

以下 `allocate-static-ip` 示例创建了可以附加到实例的指定静态 IP。

```
aws lightsail allocate-static-ip \  
  --static-ip-name StaticIp-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "b5d06d13-2f19-4683-889f-dEXAMPLEed79",  
      "resourceName": "StaticIp-1",  
      "resourceType": "StaticIp",  
      "createdAt": 1571071325.076,  
      "location": {  
        "availabilityZone": "all",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": true,  
      "operationType": "AllocateStaticIp",  
      "status": "Succeeded",  
      "statusChangedAt": 1571071325.274  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AllocateStaticIp](#)中的。

## attach-disk

以下代码示例演示了如何使用 attach-disk。

### AWS CLI

将块存储磁盘挂接到实例

以下attach-disk示例将磁盘挂载Disk-1到实例上WordPress\_Multisite-1，磁盘路径为 /dev/xvdf

```
aws lightsail attach-disk \  
  --disk-name Disk-1 \  
  --disk-path /dev/xvdf \  
  --instance-name WordPress_Multisite-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "10a08267-19ce-43be-b913-6EXAMPLE7e80",  
      "resourceName": "Disk-1",  
      "resourceType": "Disk",  
      "createdAt": 1571071465.472,  
      "location": {  
        "availabilityZone": "us-west-2a",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationDetails": "WordPress_Multisite-1",  
      "operationType": "AttachDisk",  
      "status": "Started",  
      "statusChangedAt": 1571071465.472  
    },  
    {  
      "id": "2912c477-5295-4539-88c9-bEXAMPLEd1f0",  
      "resourceName": "WordPress_Multisite-1",  
      "resourceType": "Instance",  
      "createdAt": 1571071465.474,  
      "location": {
```

```

        "availabilityZone": "us-west-2a",
        "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationDetails": "Disk-1",
    "operationType": "AttachDisk",
    "status": "Started",
    "statusChangedAt": 1571071465.474
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachDisk](#)中的。

## attach-instances-to-load-balancer

以下代码示例演示了如何使用 `attach-instances-to-load-balancer`。

### AWS CLI

将实例连接到负载均衡器

以下 `attach-instances-to-load-balancer` 示例将实例 MEAN-1、MEAN-2、和附加 MEAN-3 到负载均衡器 LoadBalancer-1。

```

aws lightsail attach-instances-to-load-balancer \
  --instance-names {"MEAN-1","MEAN-2","MEAN-3"} \
  --load-balancer-name LoadBalancer-1

```

输出：

```

{
  "operations": [
    {
      "id": "8055d19d-abb2-40b9-b527-1EXAMPLE3c7b",
      "resourceName": "LoadBalancer-1",
      "resourceType": "LoadBalancer",
      "createdAt": 1571071699.892,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      }
    },
  ],
}

```



```
    "isTerminal": false,
    "operationDetails": "MEAN-2",
    "operationType": "AttachInstancesToLoadBalancer",
    "status": "Started",
    "statusChangedAt": 1571071699.892
  },
  {
    "id": "c35048eb-8538-456a-a118-0EXAMPLEfb73",
    "resourceName": "MEAN-2",
    "resourceType": "Instance",
    "createdAt": 1571071699.887,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationDetails": "LoadBalancer-1",
    "operationType": "AttachInstancesToLoadBalancer",
    "status": "Started",
    "statusChangedAt": 1571071699.887
  },
  {
    "id": "910d09e0-adc5-4372-bc2e-0EXAMPLEd891",
    "resourceName": "LoadBalancer-1",
    "resourceType": "LoadBalancer",
    "createdAt": 1571071699.882,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationDetails": "MEAN-3",
    "operationType": "AttachInstancesToLoadBalancer",
    "status": "Started",
    "statusChangedAt": 1571071699.882
  },
  {
    "id": "178b18ac-43e8-478c-9bed-1EXAMPLE4755",
    "resourceName": "MEAN-3",
    "resourceType": "Instance",
    "createdAt": 1571071699.901,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    }
  }
}
```

```
    },
    "isTerminal": false,
    "operationDetails": "LoadBalancer-1",
    "operationType": "AttachInstancesToLoadBalancer",
    "status": "Started",
    "statusChangedAt": 1571071699.901
  },
  {
    "id": "fb62536d-2a98-4190-a6fc-4EXAMPLE7470",
    "resourceName": "LoadBalancer-1",
    "resourceType": "LoadBalancer",
    "createdAt": 1571071699.885,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationDetails": "MEAN-1",
    "operationType": "AttachInstancesToLoadBalancer",
    "status": "Started",
    "statusChangedAt": 1571071699.885
  },
  {
    "id": "787dac0d-f98d-46c3-8571-3EXAMPLE5a85",
    "resourceName": "MEAN-1",
    "resourceType": "Instance",
    "createdAt": 1571071699.901,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationDetails": "LoadBalancer-1",
    "operationType": "AttachInstancesToLoadBalancer",
    "status": "Started",
    "statusChangedAt": 1571071699.901
  }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachInstancesToLoadBalancer](#)中的。

## attach-load-balancer-tls-certificate

以下代码示例演示了如何使用 `attach-load-balancer-tls-certificate`。

### AWS CLI

将 TLS 证书附加到负载均衡器

以下 `attach-load-balancer-tls-certificate` 示例将负载均衡器 TLS 证书附加到 `Certificate2` 到负载均衡器 `LoadBalancer-1`。

```
aws lightsail attach-load-balancer-tls-certificate \  
  --certificate-name Certificate2 \  
  --load-balancer-name LoadBalancer-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "cf1ad6e3-3cbb-4b8a-a7f2-3EXAMPLEa118",  
      "resourceName": "LoadBalancer-1",  
      "resourceType": "LoadBalancer",  
      "createdAt": 1571072255.416,  
      "location": {  
        "availabilityZone": "all",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": true,  
      "operationDetails": "Certificate2",  
      "operationType": "AttachLoadBalancerTlsCertificate",  
      "status": "Succeeded",  
      "statusChangedAt": 1571072255.416  
    },  
    {  
      "id": "dae1bcfb-d531-4c06-b4ea-bEXAMPLEc04e",  
      "resourceName": "Certificate2",  
      "resourceType": "LoadBalancerTlsCertificate",  
      "createdAt": 1571072255.416,  
      "location": {  
        "availabilityZone": "all",  
        "regionName": "us-west-2"  
      },  
    },  
  ],  
}
```

```

        "isTerminal": true,
        "operationDetails": "LoadBalancer-1",
        "operationType": "AttachLoadBalancerTlsCertificate",
        "status": "Succeeded",
        "statusChangedAt": 1571072255.416
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachLoadBalancerTlsCertificate](#)中的。

## attach-static-ip

以下代码示例演示了如何使用 attach-static-ip。

### AWS CLI

将静态 IP 附加到实例

以下attach-static-ip示例将静态 IP 附加StaticIp-1到实例MEAN-1。

```

aws lightsail attach-static-ip \
  --static-ip-name StaticIp-1 \
  --instance-name MEAN-1

```

输出：

```

{
  "operations": [
    {
      "id": "45e6fa13-4808-4b8d-9292-bEXAMPLE20b2",
      "resourceName": "StaticIp-1",
      "resourceType": "StaticIp",
      "createdAt": 1571072569.375,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      },
      "isTerminal": true,
      "operationDetails": "MEAN-1",
      "operationType": "AttachStaticIp",
      "status": "Succeeded",
    }
  ]
}

```

```

        "statusChangedAt": 1571072569.375
    },
    {
        "id": "9ee09a17-863c-4e51-8a6d-3EXAMPLE5475",
        "resourceName": "MEAN-1",
        "resourceType": "Instance",
        "createdAt": 1571072569.376,
        "location": {
            "availabilityZone": "us-west-2a",
            "regionName": "us-west-2"
        },
        "isTerminal": true,
        "operationDetails": "StaticIp-1",
        "operationType": "AttachStaticIp",
        "status": "Succeeded",
        "statusChangedAt": 1571072569.376
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachStaticIp](#)中的。

## close-instance-public-ports

以下代码示例演示了如何使用 `close-instance-public-ports`。

### AWS CLI

关闭实例的防火墙端口

以下`close-instance-public-ports`示例关闭实例22上的 TCP 端口MEAN-2。

```

aws lightsail close-instance-public-ports \
  --instance-name MEAN-2 \
  --port-info fromPort=22,protocol=TCP,toPort=22

```

输出：

```

{
  "operation": {
    "id": "4f328636-1c96-4649-ae6d-1EXAMPLEf446",
    "resourceName": "MEAN-2",
    "resourceType": "Instance",

```

```

    "createdAt": 1571072845.737,
    "location": {
      "availabilityZone": "us-west-2a",
      "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationDetails": "22/tcp",
    "operationType": "CloseInstancePublicPorts",
    "status": "Succeeded",
    "statusChangedAt": 1571072845.737
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CloseInstancePublicPorts](#)中的。

## copy-snapshot

以下代码示例演示了如何使用 copy-snapshot。

### AWS CLI

示例 1：在同一 AWS 区域内复制快照

以下copy-snapshot示例将实例快照复制MEAN-1-1571075291为同一 AWS 区域MEAN-1-Copy内的实例快照us-west-2。

```

aws lightsail copy-snapshot \
  --source-snapshot-name MEAN-1-1571075291 \
  --target-snapshot-name MEAN-1-Copy \
  --source-region us-west-2

```

输出：

```

{
  "operations": [
    {
      "id": "ced16fc1-f401-4556-8d82-1EXAMPLEb982",
      "resourceName": "MEAN-1-Copy",
      "resourceType": "InstanceSnapshot",
      "createdAt": 1571075581.498,
      "location": {
        "availabilityZone": "all",

```

```

        "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationDetails": "us-west-2:MEAN-1-1571075291",
    "operationType": "CopySnapshot",
    "status": "Started",
    "statusChangedAt": 1571075581.498
  }
]
}

```

有关更多信息，请参阅《Lightsail 开发者指南》中的 [Amazon Lightsail 中将快照从一个 AWS 区域复制到另一个区域](#)。

示例 2：将快照从一个 AWS 区域复制到另一个区域

以下 copy-snapshot 示例将实例快照 MEAN-1-1571075291 作为实例快照 MEAN-1-1571075291-Copy 从 AWS Region 复制 us-west-2 到 us-east-1。

```

aws lightsail copy-snapshot \
  --source-snapshot-name MEAN-1-1571075291 \
  --target-snapshot-name MEAN-1-1571075291-Copy \
  --source-region us-west-2 \
  --region us-east-1

```

输出：

```

{
  "operations": [
    {
      "id": "91116b79-119c-4451-b44a-dEXAMPLEd97b",
      "resourceName": "MEAN-1-1571075291-Copy",
      "resourceType": "InstanceSnapshot",
      "createdAt": 1571075695.069,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-east-1"
      },
      "isTerminal": false,
      "operationDetails": "us-west-2:MEAN-1-1571075291",
      "operationType": "CopySnapshot",
      "status": "Started",
      "statusChangedAt": 1571075695.069
    }
  ]
}

```

```
    }  
  ]  
}
```

有关更多信息，请参阅《Lightsail 开发者指南》中的 [Amazon Lightsail 中将快照从一个 AWS 区域复制到另一个区域](#)。

示例 3：在同一 AWS 区域内复制自动快照

以下 copy-snapshot 示例将实例 WordPress-1 的自动快照复制到 WordPress-1 为 AWS 区域 WordPress-1-10142019 中的手动快照 us-west-2。

```
aws lightsail copy-snapshot \  
  --source-resource-name WordPress-1 \  
  --restore-date 2019-10-14 \  
  --target-snapshot-name WordPress-1-10142019 \  
  --source-region us-west-2
```

输出：

```
{  
  "operations": [  
    {  
      "id": "be3e6754-cd1d-48e6-ad9f-2EXAMPLE1805",  
      "resourceName": "WordPress-1-10142019",  
      "resourceType": "InstanceSnapshot",  
      "createdAt": 1571082412.311,  
      "location": {  
        "availabilityZone": "all",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationDetails": "us-west-2:WordPress-1",  
      "operationType": "CopySnapshot",  
      "status": "Started",  
      "statusChangedAt": 1571082412.311  
    }  
  ]  
}
```

有关更多信息，请参阅 Lightsail 开发者指南中的 [在 Amazon Lightsail 中保存实例或磁盘的自动快照](#)。



#### 示例 4：将自动快照从一个 AWS 区域复制到另一个区域

以下copy-snapshot示例将实例2019-10-14的自动快照WordPress-1作为手动快照WordPress-1-10142019从 AWS 区域复制us-west-2到us-east-1。

```
aws lightsail copy-snapshot \  
  --source-resource-name WordPress-1 \  
  --restore-date 2019-10-14 \  
  --target-snapshot-name WordPress-1-10142019 \  
  --source-region us-west-2 \  
  --region us-east-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "dfffa128b-0b07-476e-b390-bEXAMPLE3775",  
      "resourceName": "WordPress-1-10142019",  
      "resourceType": "InstanceSnapshot",  
      "createdAt": 1571082493.422,  
      "location": {  
        "availabilityZone": "all",  
        "regionName": "us-east-1"  
      },  
      "isTerminal": false,  
      "operationDetails": "us-west-2:WordPress-1",  
      "operationType": "CopySnapshot",  
      "status": "Started",  
      "statusChangedAt": 1571082493.422  
    }  
  ]  
}
```

有关更多信息，请参阅 Lightsail 开发[者指南中的在 Amazon Lightsail 中保存实例或磁盘的自动快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CopySnapshot](#)中的。

## create-disk-from-snapshot

以下代码示例演示了如何使用 create-disk-from-snapshot。

## AWS CLI

### 使用磁盘快照创建磁盘

以下create-disk-from-snapshot示例根据指定的块存储磁盘快照创建了一个Disk-2名为的块存储磁盘。磁盘在指定的 AWS 区域和可用区中创建，存储空间为 32 GB。

```
aws lightsail create-disk-from-snapshot \  
  --disk-name Disk-2 \  
  --disk-snapshot-name Disk-1-1566839161 \  
  --availability-zone us-west-2a \  
  --size-in-gb 32
```

输出：

```
{  
  "operations": [  
    {  
      "id": "d42b605d-5ef1-4b4a-8791-7a3e8b66b5e7",  
      "resourceName": "Disk-2",  
      "resourceType": "Disk",  
      "createdAt": 1569624941.471,  
      "location": {  
        "availabilityZone": "us-west-2a",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationType": "CreateDiskFromSnapshot",  
      "status": "Started",  
      "statusChangedAt": 1569624941.791  
    }  
  ]  
}
```

有关更多信息，请参阅 [Lightsail 开发者指南中的在 Amazon Lightsail 中使用快照创建块存储磁盘](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDiskFromSnapshot](#)中的。

## create-disk-snapshot

以下代码示例演示了如何使用 create-disk-snapshot。

## AWS CLI

### 示例 1：创建磁盘快照

以下create-disk-snapshot示例为指定的块存储磁盘创建名为DiskSnapshot-1的快照。

```
aws lightsail create-disk-snapshot \  
  --disk-name Disk-1 \  
  --disk-snapshot-name DiskSnapshot-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "fa74c6d2-03a3-4f42-a7c7-792f124d534b",  
      "resourceName": "DiskSnapshot-1",  
      "resourceType": "DiskSnapshot",  
      "createdAt": 1569625129.739,  
      "location": {  
        "availabilityZone": "all",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationDetails": "Disk-1",  
      "operationType": "CreateDiskSnapshot",  
      "status": "Started",  
      "statusChangedAt": 1569625129.739  
    },  
    {  
      "id": "920a25df-185c-4528-87cd-7b85f5488c06",  
      "resourceName": "Disk-1",  
      "resourceType": "Disk",  
      "createdAt": 1569625129.739,  
      "location": {  
        "availabilityZone": "us-west-2a",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationDetails": "DiskSnapshot-1",  
      "operationType": "CreateDiskSnapshot",  
      "status": "Started",  
      "statusChangedAt": 1569625129.739  
    }  
  ]  
}
```

```
    }  
  ]  
}
```

## 示例 2：创建实例系统磁盘的快照

以下 `create-disk-snapshot` 示例创建指定实例系统磁盘的快照。

```
aws lightsail create-disk-snapshot \  
  --instance-name WordPress-1 \  
  --disk-snapshot-name SystemDiskSnapshot-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "f508cf1c-6597-42a6-a4c3-4aebd75af0d9",  
      "resourceName": "SystemDiskSnapshot-1",  
      "resourceType": "DiskSnapshot",  
      "createdAt": 1569625294.685,  
      "location": {  
        "availabilityZone": "all",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationDetails": "WordPress-1",  
      "operationType": "CreateDiskSnapshot",  
      "status": "Started",  
      "statusChangedAt": 1569625294.685  
    },  
    {  
      "id": "0bb9f712-da3b-4d99-b508-3bf871d989e5",  
      "resourceName": "WordPress-1",  
      "resourceType": "Instance",  
      "createdAt": 1569625294.685,  
      "location": {  
        "availabilityZone": "us-west-2a",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationDetails": "SystemDiskSnapshot-1",  
      "operationType": "CreateDiskSnapshot",
```

```
        "status": "Started",
        "statusChangedAt": 1569625294.685
      }
    ]
  }
```

有关更多信息，请参阅 Lightsail 开发者[指南中的亚马逊 Lightsail](#) 中的[快照和在 Amazon Lightsail 中创建实例根卷的快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateDiskSnapshot](#) 中的。

## create-disk

以下代码示例演示了如何使用 create-disk。

### AWS CLI

#### 创建块存储磁盘

以下 create-disk 示例在指定的 AWS 区域和可用区 Disk-1 中创建一个具有 32 GB 存储空间的块存储磁盘。

```
aws lightsail create-disk \  
  --disk-name Disk-1 \  
  --availability-zone us-west-2a \  
  --size-in-gb 32
```

输出：

```
{  
  "operations": [  
    {  
      "id": "1c85e2ec-86ba-4697-b936-77f4d3dc013a",  
      "resourceName": "Disk-1",  
      "resourceType": "Disk",  
      "createdAt": 1569449220.36,  
      "location": {  
        "availabilityZone": "us-west-2a",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationType": "CreateDisk",
```

```

        "status": "Started",
        "statusChangedAt": 1569449220.588
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDisk](#)中的。

## create-domain-entry

以下代码示例演示了如何使用 create-domain-entry。

### AWS CLI

创建域条目 ( DNS 记录 )

以下create-domain-entry示例为指定域的顶点创建指向实例 IP 地址的 DNS 记录 (A)。

注意：Lightsail 与域名相关的 API 操作仅在该地区可用。us-east-1如果您的 CLI 配置文件配置为使用其他区域，则必须包含--region us-east-1参数，否则命令将失败。

```

aws lightsail create-domain-entry \
  --region us-east-1 \
  --domain-name example.com \
  --domain-entry name=example.com,type=A,target=192.0.2.0

```

输出：

```

{
  "operation": {
    "id": "5be4494d-56f4-41fc-8730-693dcd0ef9e2",
    "resourceName": "example.com",
    "resourceType": "Domain",
    "createdAt": 1569865296.519,
    "location": {
      "availabilityZone": "all",
      "regionName": "global"
    },
    "isTerminal": true,
    "operationType": "CreateDomainEntry",
    "status": "Succeeded",
    "statusChangedAt": 1569865296.519
  }
}

```

```
}  
}
```

有关更多信息，请参阅《Lightsail 开发者指南》中的 [Amazon Lightsail](#) 中的 [DNS 和 Amazon Lightsail 中创建 DNS 区域以管理您的域名的 DNS 记录](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDomainEntry](#) 中的。

## create-domain

以下代码示例演示了如何使用 create-domain。

### AWS CLI

创建域 ( DNS 区域 )

以下 create-domain 示例为指定域创建 DNS 区域。

注意：Lightsail 与域名相关的 API 操作仅在该地区可用。us-east-1 如果您的 CLI 配置文件配置为使用其他区域，则必须包含 `--region us-east-1` 参数，否则命令将失败。

```
aws lightsail create-domain \  
  --region us-east-1 \  
  --domain-name example.com
```

输出：

```
{  
  "operation": {  
    "id": "64e522c8-9ae1-4c05-9b65-3f237324dc34",  
    "resourceName": "example.com",  
    "resourceType": "Domain",  
    "createdAt": 1569864291.92,  
    "location": {  
      "availabilityZone": "all",  
      "regionName": "global"  
    },  
    "isTerminal": true,  
    "operationType": "CreateDomain",  
    "status": "Succeeded",  
    "statusChangedAt": 1569864292.109  
  }  
}
```

```
}
```

有关更多信息，请参阅《Lightsail 开发者指南》中的 [Amazon Lightsail 中的 DNS 和 Amazon Lightsail 中创建 DNS 区域以管理您的域名的 DNS 记录](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDomain](#) 中的。

## create-instance-snapshot

以下代码示例演示了如何使用 create-instance-snapshot。

### AWS CLI

#### 创建实例的快照

以下 create-instance-snapshot 示例从指定实例创建快照。

```
aws lightsail create-instance-snapshot \  
  --instance-name WordPress-1 \  
  --instance-snapshot-name WordPress-Snapshot-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "4c3db559-9dd0-41e7-89c0-2cb88c19786f",  
      "resourceName": "WordPress-Snapshot-1",  
      "resourceType": "InstanceSnapshot",  
      "createdAt": 1569866438.48,  
      "location": {  
        "availabilityZone": "all",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationDetails": "WordPress-1",  
      "operationType": "CreateInstanceSnapshot",  
      "status": "Started",  
      "statusChangedAt": 1569866438.48  
    },  
    {  
      "id": "c04fdc45-2981-488c-88b5-d6d2fd759a6a",
```



```

    "resourceName": "WordPress-1",
    "resourceType": "Instance",
    "createdAt": 1569866438.48,
    "location": {
      "availabilityZone": "us-west-2a",
      "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationDetails": "WordPress-Snapshot-1",
    "operationType": "CreateInstanceSnapshot",
    "status": "Started",
    "statusChangedAt": 1569866438.48
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateInstanceSnapshot](#)中的。

## create-instances-from-snapshot

以下代码示例演示了如何使用 `create-instances-from-snapshot`。

### AWS CLI

#### 使用快照创建实例

以下`create-instances-from-snapshot`示例使用价值 10 美元的捆绑包，在指定的 AWS 区域和可用区根据指定的实例快照创建实例。

注意：您指定的捆绑包的规格必须等于或大于用于创建快照的原始源实例的捆绑包。

```

aws lightsail create-instances-from-snapshot \
  --instance-snapshot-name WordPress-1-1569866208 \
  --instance-names WordPress-2 \
  --availability-zone us-west-2a \
  --bundle-id medium_2_0

```

输出：

```

{
  "operations": [
    {

```

```

    "id": "003f8271-b711-464d-b9b8-7f3806cb496e",
    "resourceName": "WordPress-2",
    "resourceType": "Instance",
    "createdAt": 1569865914.908,
    "location": {
      "availabilityZone": "us-west-2a",
      "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationType": "CreateInstancesFromSnapshot",
    "status": "Started",
    "statusChangedAt": 1569865914.908
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateInstancesFromSnapshot](#)中的。

## create-instances

以下代码示例演示了如何使用 create-instances。

### AWS CLI

#### 示例 1：创建单个实例

以下create-instances示例使用 WordPress 蓝图和价值 3.50 美元的捆绑包在指定的 AWS 区域和可用区创建实例。

```

aws lightsail create-instances \
  --instance-names Instance-1 \
  --availability-zone us-west-2a \
  --blueprint-id wordpress_5_1_1_2 \
  --bundle-id nano_2_0

```

输出：

```

{
  "operations": [
    {
      "id": "9a77158f-7be3-4d6d-8054-cf5ae2b720cc",
      "resourceName": "Instance-1",

```

```

    "resourceType": "Instance",
    "createdAt": 1569447986.061,
    "location": {
      "availabilityZone": "us-west-2a",
      "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationType": "CreateInstance",
    "status": "Started",
    "statusChangedAt": 1569447986.061
  }
]
}

```

## 示例 2：一次创建多个实例

以下 `create-instances` 示例使用 WordPress 蓝图和价值 3.50 美元的捆绑包，在指定的 AWS 区域和可用区创建三个实例。

```

aws lightsail create-instances \
  --instance-names {"Instance1","Instance2","Instance3"} \
  --availability-zone us-west-2a \
  --blueprint-id wordpress_5_1_1_2 \
  --bundle-id nano_2_0

```

输出：

```

{
  "operations": [
    {
      "id": "5492f015-9d2e-48c6-8eea-b516840e6903",
      "resourceName": "Instance1",
      "resourceType": "Instance",
      "createdAt": 1569448780.054,
      "location": {
        "availabilityZone": "us-west-2a",
        "regionName": "us-west-2"
      },
      "isTerminal": false,
      "operationType": "CreateInstance",
      "status": "Started",
      "statusChangedAt": 1569448780.054
    },

```

```
{
  "id": "c58b5f46-2676-44c8-b95c-3ad375898515",
  "resourceName": "Instance2",
  "resourceType": "Instance",
  "createdAt": 1569448780.054,
  "location": {
    "availabilityZone": "us-west-2a",
    "regionName": "us-west-2"
  },
  "isTerminal": false,
  "operationType": "CreateInstance",
  "status": "Started",
  "statusChangedAt": 1569448780.054
},
{
  "id": "a5ad8006-9bee-4499-9eb7-75e42e6f5882",
  "resourceName": "Instance3",
  "resourceType": "Instance",
  "createdAt": 1569448780.054,
  "location": {
    "availabilityZone": "us-west-2a",
    "regionName": "us-west-2"
  },
  "isTerminal": false,
  "operationType": "CreateInstance",
  "status": "Started",
  "statusChangedAt": 1569448780.054
}
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateInstances](#)中的。

## create-key-pair

以下代码示例演示了如何使用 create-key-pair。

### AWS CLI

#### 创建密钥对

以下create-key-pair示例创建了一个 key pair，您可以使用该密钥对进行身份验证和连接到实例。

```
aws lightsail create-key-pair \
  --key-pair-name MyPersonalKeyPair
```

输出提供了私钥 base64 值，您可以使用该值对使用已创建密钥对的实例进行身份验证。注意：将私钥 base64 值复制并粘贴到安全位置，因为以后无法检索。

```
{
  "keyPair": {
    "name": "MyPersonalKeyPair",
    "arn": "arn:aws:lightsail:us-west-2:111122223333:KeyPair/55025c71-198f-403b-
b42f-a69433e724fb",
    "supportCode": "621291663362/MyPersonalKeyPair",
    "createdAt": 1569866556.567,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "resourceType": "KeyPair"
  },
  "publicKeyBase64": "ssh-rsa ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCV0xUEwx96amPERH7K1bVT1tTF190mNk6o7m5YVhk9x10dMbDRbFvhtXvw4jz
+BUgedGUXno6uF7agqxZN01kPLJBIVTW26SSYBJ0tE
+y804UyVsjrBqUqCaMXDhmfXpWuLMPwuXhwcKh7e8hwoTfkiX0E6Ql
+KqF/MiA3w6DCjEqvvdI07SiEZJFsuGNfYDDN3w60Re15MUhmn30Jdn4y/
A7Nwb3IxL4pFvE4rgFRKU8n1jp9kwRnLVMVB0WuGXk6n+H6M2f1 ",
  "privateKeyBase64": "-----BEGIN RSA PRIVATE KEY-----
EXAMPLETCCafICCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMakGA1UEBhMC
\nVVMxCzAJBgNVBAgTAldBMRAwDgYDVQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6\nnb24xFDASBgNVBAAsTC01BTSBD
\nBgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
\nMTIwNDI0MjA0NTIxWjCBiDELMakGA1UEBhMCMVVMxCzAJBgNVBAgTAldBMRAwDgYD
\nVQHEwdTZWF0dGx1MQ8wDQEXAMPEwZBbWF6b24xFDASBgNVBAAsTC01BTSBDb25z
\nb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWxhZAdBgkqhkiG9w0BCQEWEG5vb251QGft
\nYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMEXAMPLE4GmWIWJ
\n21uUSfwfEvySwTc2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
\nrDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
\nIbb30hjZnzcVQAaREXAMPLEm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4\nnnUHVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
\nFFBjvSfpJI1J00zbhNYS5f6GuoEDmFJ10ZxBHjJnyp3780EXAMPLELvjx79LjStB
\nNYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=\n-----END RSA PRIVATE KEY-----",
  "operation": {
    "id": "67f984db-9994-45fe-ad38-59bafcaf82ef",
    "resourceName": "MyPersonalKeyPair",
    "resourceType": "KeyPair",
  }
}
```

```

    "createdAt": 1569866556.567,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationType": "CreateKeyPair",
    "status": "Succeeded",
    "statusChangedAt": 1569866556.704
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateKeyPair](#)中的。

## create-load-balancer-tls-certificate

以下代码示例演示了如何使用 `create-load-balancer-tls-certificate`。

### AWS CLI

为负载均衡器创建 TLS 证书

以下`create-load-balancer-tls-certificate`示例创建了一个附加到指定负载均衡器的 TLS 证书。创建的证书适用于指定的域。注意：只能为负载均衡器创建两个证书。

```

aws lightsail create-load-balancer-tls-certificate \
  --certificate-alternative-names abc.example.com \
  --certificate-domain-name example.com \
  --certificate-name MySecondCertificate \
  --load-balancer-name MyFirstLoadBalancer

```

输出：

```

{
  "operations": [
    {
      "id": "be663aed-cb46-41e2-9b23-e2f747245bd4",
      "resourceName": "MySecondCertificate",
      "resourceType": "LoadBalancerTlsCertificate",
      "createdAt": 1569867364.971,
      "location": {
        "availabilityZone": "all",

```

```

        "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationDetails": "MyFirstLoadBalancer",
    "operationType": "CreateLoadBalancerTlsCertificate",
    "status": "Succeeded",
    "statusChangedAt": 1569867365.219
},
{
    "id": "f3dfa930-969e-41cc-ac7d-337178716f6d",
    "resourceName": "MyFirstLoadBalancer",
    "resourceType": "LoadBalancer",
    "createdAt": 1569867364.971,
    "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationDetails": "MySecondCertificate",
    "operationType": "CreateLoadBalancerTlsCertificate",
    "status": "Succeeded",
    "statusChangedAt": 1569867365.219
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLoadBalancerTlsCertificate](#)中的。

## create-load-balancer

以下代码示例演示了如何使用 create-load-balancer。

### AWS CLI

#### 创建负载均衡器

以下create-load-balancer示例使用 TLS 证书创建负载均衡器。TLS 证书适用于指定的域，并将流量路由到端口 80 上的实例。

```

aws lightsail create-load-balancer \
  --certificate-alternative-names www.example.com test.example.com \
  --certificate-domain-name example.com \
  --certificate-name Certificate-1 \

```

```
--instance-port 80 \  
--load-balancer-name LoadBalancer-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "cc7b920a-83d8-4762-a74e-9174fe1540be",  
      "resourceName": "LoadBalancer-1",  
      "resourceType": "LoadBalancer",  
      "createdAt": 1569867169.406,  
      "location": {  
        "availabilityZone": "all",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationType": "CreateLoadBalancer",  
      "status": "Started",  
      "statusChangedAt": 1569867169.406  
    },  
    {  
      "id": "658ed43b-f729-42f3-a8e4-3f8024d3c98d",  
      "resourceName": "LoadBalancer-1",  
      "resourceType": "LoadBalancerTlsCertificate",  
      "createdAt": 1569867170.193,  
      "location": {  
        "availabilityZone": "all",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": true,  
      "operationDetails": "LoadBalancer-1",  
      "operationType": "CreateLoadBalancerTlsCertificate",  
      "status": "Succeeded",  
      "statusChangedAt": 1569867170.54  
    },  
    {  
      "id": "4757a342-5181-4870-b1e0-227eebc35ab5",  
      "resourceName": "LoadBalancer-1",  
      "resourceType": "LoadBalancer",  
      "createdAt": 1569867170.193,  
      "location": {  
        "availabilityZone": "all",
```



```

        "regionName": "us-west-2"
      },
      "isTerminal": true,
      "operationDetails": "Certificate-1",
      "operationType": "CreateLoadBalancerTlsCertificate",
      "status": "Succeeded",
      "statusChangedAt": 1569867170.54
    }
  ]
}

```

有关更多信息，请参阅 [Lightsail 开发者指南中的 Lightsail 负载均衡器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateLoadBalancer](#) 中的。

## create-relational-database-from-snapshot

以下代码示例演示了如何使用 `create-relational-database-from-snapshot`。

### AWS CLI

#### 从快照创建托管数据库

以下 `create-relational-database-from-snapshot` 示例使用价值 15 美元的标准数据库捆绑包，从指定 AWS 区域和可用区的指定快照创建托管数据库。注：您指定的捆绑包的规格必须等于或大于用于创建快照的原始源数据库的捆绑包。

```

aws lightsail create-relational-database-from-snapshot \
  --relational-database-snapshot-name Database-Oregon-1-1566839359 \
  --relational-database-name Database-1 \
  --availability-zone us-west-2a \
  --relational-database-bundle-id micro_1_0 \
  --no-publicly-accessible

```

输出：

```

{
  "operations": [
    {
      "id": "ad6d9193-9d5c-4ea1-97ae-8fe6de600b4c",
      "resourceName": "Database-1",

```

```

    "resourceType": "RelationalDatabase",
    "createdAt": 1569867916.938,
    "location": {
      "availabilityZone": "us-west-2a",
      "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationType": "CreateRelationalDatabaseFromSnapshot",
    "status": "Started",
    "statusChangedAt": 1569867918.643
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRelationalDatabaseFromSnapshot](#)中的。

## create-relational-database-snapshot

以下代码示例演示了如何使用 create-relational-database-snapshot。

### AWS CLI

#### 创建托管数据库的快照

以下create-relational-database-snapshot示例创建了指定托管数据库的快照。

```

aws lightsail create-relational-database-snapshot \
  --relational-database-name Database1 \
  --relational-database-snapshot-name RelationalDatabaseSnapshot1

```

输出：

```

{
  "operations": [
    {
      "id": "853667fb-ea91-4c02-8d20-8fc5fd43b9eb",
      "resourceName": "RelationalDatabaseSnapshot1",
      "resourceType": "RelationalDatabaseSnapshot",
      "createdAt": 1569868074.645,
      "location": {

```

```
        "availabilityZone": "all",
        "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationDetails": "Database1",
    "operationType": "CreateRelationalDatabaseSnapshot",
    "status": "Started",
    "statusChangedAt": 1569868074.645
},
{
    "id": "fbafa521-3cac-4be8-9773-1c143780b239",
    "resourceName": "Database1",
    "resourceType": "RelationalDatabase",
    "createdAt": 1569868074.645,
    "location": {
        "availabilityZone": "us-west-2a",
        "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationDetails": "RelationalDatabaseSnapshot1",
    "operationType": "CreateRelationalDatabaseSnapshot",
    "status": "Started",
    "statusChangedAt": 1569868074.645
}
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRelationalDatabaseSnapshot](#)中的。

## create-relational-database

以下代码示例演示了如何使用 create-relational-database。

### AWS CLI

#### 创建托管数据库

以下create-relational-database示例使用 MySQL 5.6 数据库引擎 (mysql\_5\_6) 和价值 15 美元的标准数据库捆绑包 (micro\_1\_0) 在指定的 AWS 区域和可用区创建托管数据库。托管数据库已预先填充主用户名，不可公开访问。

```
aws lightsail create-relational-database \
```

```
--relational-database-name Database-1 \  
--availability-zone us-west-2a \  
--relational-database-blueprint-id mysql_5_6 \  
--relational-database-bundle-id micro_1_0 \  
--master-database-name dbmaster \  
--master-username user \  
--no-publicly-accessible
```

输出：

```
{  
  "operations": [  
    {  
      "id": "b52bedee-73ed-4798-8d2a-9c12df89adcd",  
      "resourceName": "Database-1",  
      "resourceType": "RelationalDatabase",  
      "createdAt": 1569450017.244,  
      "location": {  
        "availabilityZone": "us-west-2a",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationType": "CreateRelationalDatabase",  
      "status": "Started",  
      "statusChangedAt": 1569450018.637  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRelationalDatabase](#)中的。

## delete-auto-snapshot

以下代码示例演示了如何使用 delete-auto-snapshot。

### AWS CLI

#### 删除自动快照

以下delete-auto-snapshot示例删除了实例2019-10-10的自动快照WordPress-1。

```
aws lightsail delete-auto-snapshot \  

```

```
--resource-name WordPress-1 \  
--date 2019-10-10
```

输出：

```
{  
  "operations": [  
    {  
      "id": "31c36e09-3d52-46d5-b6d8-7EXAMPLE534a",  
      "resourceName": "WordPress-1",  
      "resourceType": "Instance",  
      "createdAt": 1571088141.501,  
      "location": {  
        "availabilityZone": "us-west-2",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": true,  
      "operationDetails": "DeleteAutoSnapshot-2019-10-10",  
      "operationType": "DeleteAutoSnapshot",  
      "status": "Succeeded"  
    }  
  ]  
}
```

有关更多信息，请参阅 [Lightsail 开发者指南中的在 Amazon Lightsail 中删除实例或磁盘的自动快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteAutoSnapshot](#) 中的。

## delete-disk-snapshot

以下代码示例演示了如何使用 delete-disk-snapshot。

### AWS CLI

删除块存储磁盘的快照

以下 delete-disk-snapshot 示例删除块存储磁盘的指定快照

```
aws lightsail delete-disk-snapshot \  
--disk-snapshot-name DiskSnapshot-1
```

输出：

```
{
  "operations": [
    {
      "id": "d1e5766d-b81e-4595-ad5d-02afbcccfd5d",
      "resourceName": "DiskSnapshot-1",
      "resourceType": "DiskSnapshot",
      "createdAt": 1569873552.79,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      },
      "isTerminal": true,
      "operationType": "DeleteDiskSnapshot",
      "status": "Succeeded",
      "statusChangedAt": 1569873552.79
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDiskSnapshot](#)中的。

## delete-disk

以下代码示例演示了如何使用 delete-disk。

### AWS CLI

#### 删除块存储磁盘

以下delete-disk示例删除了指定的块存储磁盘。

```
aws lightsail delete-disk \
  --disk-name Disk-1
```

输出：

```
{
  "operations": [
    {
```

```

    "id": "6378c70f-4d75-4f7a-ab66-730fca0bb2fc",
    "resourceName": "Disk-1",
    "resourceType": "Disk",
    "createdAt": 1569872887.864,
    "location": {
      "availabilityZone": "us-west-2a",
      "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationType": "DeleteDisk",
    "status": "Succeeded",
    "statusChangedAt": 1569872887.864
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDisk](#)中的。

## delete-domain-entry

以下代码示例演示了如何使用 delete-domain-entry。

### AWS CLI

删除域名条目 ( DNS 记录 )

以下delete-domain-entry示例从现有域中删除指定的域条目。

注意：Lightsail 与域名相关的 API 操作仅在该地区可用。us-east-1如果您的 CLI 配置文件配置为使用其他区域，则必须包含--region us-east-1参数，否则命令将失败。

```

aws lightsail delete-domain-entry \
  --region us-east-1 \
  --domain-name example.com \
  --domain-entry name=123.example.com,target=192.0.2.0,type=A

```

输出：

```

{
  "operation": {
    "id": "06eacd01-d785-420e-8daa-823150c7dca1",

```

```
    "resourceName": "example.com ",
    "resourceType": "Domain",
    "createdAt": 1569874157.005,
    "location": {
      "availabilityZone": "all",
      "regionName": "global"
    },
    "isTerminal": true,
    "operationType": "DeleteDomainEntry",
    "status": "Succeeded",
    "statusChangedAt": 1569874157.005
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDomainEntry](#)中的。

## delete-domain

以下代码示例演示了如何使用 delete-domain。

### AWS CLI

#### 删除域 ( DNS 区域 )

以下delete-domain示例删除了指定的域和该域中的所有条目 ( DNS 记录 )。

注意：Lightsail 与域名相关的 API 操作仅在该地区可用。us-east-1如果您的 CLI 配置文件配置为使用其他区域，则必须包含--region us-east-1参数，否则命令将失败。

```
aws lightsail delete-domain \
  --region us-east-1 \
  --domain-name example.com
```

输出：

```
{
  "operation": {
    "id": "fcef5265-5af1-4a46-a3d7-90b5e18b9b32",
    "resourceName": "example.com",
    "resourceType": "Domain",
    "createdAt": 1569873788.13,
    "location": {
```



```
        "availabilityZone": "all",
        "regionName": "global"
    },
    "isTerminal": true,
    "operationType": "DeleteDomain",
    "status": "Succeeded",
    "statusChangedAt": 1569873788.13
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDomain](#)中的。

## delete-instance-snapshot

以下代码示例演示了如何使用 delete-instance-snapshot。

### AWS CLI

title

以下delete-instance-snapshot示例删除了实例的指定快照。

```
aws lightsail delete-instance-snapshot \
  --instance-snapshot-name WordPress-1-Snapshot-1
```

输出：

```
{
  "operations": [
    {
      "id": "14dad182-976a-46c6-bfd4-9480482bf0ea",
      "resourceName": "WordPress-1-Snapshot-1",
      "resourceType": "InstanceSnapshot",
      "createdAt": 1569874524.562,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      },
      "isTerminal": true,
      "operationType": "DeleteInstanceSnapshot",
      "status": "Succeeded",
      "statusChangedAt": 1569874524.562
    }
  ]
}
```

```
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInstanceSnapshot](#)中的。

## delete-instance

以下代码示例演示了如何使用 delete-instance。

### AWS CLI

#### 删除实例

以下delete-instance示例删除了指定的实例。

```
aws lightsail delete-instance \  
  --instance-name WordPress-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "d77345a3-8f80-4d2e-b47d-aaa622718df2",  
      "resourceName": "Disk-1",  
      "resourceType": "Disk",  
      "createdAt": 1569874357.469,  
      "location": {  
        "availabilityZone": "us-west-2a",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationDetails": "WordPress-1",  
      "operationType": "DetachDisk",  
      "status": "Started",  
      "statusChangedAt": 1569874357.469  
    },  
    {  
      "id": "708fa606-2bfd-4e48-a2c1-0b856585b5b1",  
      "resourceName": "WordPress-1",  
      "resourceType": "Instance",
```

```
    "createdAt": 1569874357.465,
    "location": {
      "availabilityZone": "us-west-2a",
      "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationDetails": "Disk-1",
    "operationType": "DetachDisk",
    "status": "Started",
    "statusChangedAt": 1569874357.465
  },
  {
    "id": "3187e823-8acb-405d-b098-fad5ceb17bec",
    "resourceName": "WordPress-1",
    "resourceType": "Instance",
    "createdAt": 1569874357.829,
    "location": {
      "availabilityZone": "us-west-2a",
      "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationType": "DeleteInstance",
    "status": "Succeeded",
    "statusChangedAt": 1569874357.829
  }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInstance](#)中的。

## delete-key-pair

以下代码示例演示了如何使用 delete-key-pair。

### AWS CLI

#### 删除密钥对

以下delete-key-pair示例删除指定的 key pair。

```
aws lightsail delete-key-pair \  
  --key-pair-name MyPersonalKeyPair
```

输出：

```
{
  "operation": {
    "id": "81621463-df38-4810-b866-6e801a15abbf",
    "resourceName": "MyPersonalKeyPair",
    "resourceType": "KeyPair",
    "createdAt": 1569874626.466,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationType": "DeleteKeyPair",
    "status": "Succeeded",
    "statusChangedAt": 1569874626.685
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteKeyPair](#)中的。

## delete-known-host-keys

以下代码示例演示了如何使用 delete-known-host-keys。

### AWS CLI

从实例中删除已知的主机密钥

以下delete-known-host-keys示例从指定实例中删除已知的主机密钥。

```
aws lightsail delete-known-host-keys \
  --instance-name Instance-1
```

输出：

```
{
  "operations": [
    {
      "id": "c61afe9c-45a4-41e6-a97e-d212364da3f5",
      "resourceName": "Instance-1",
```

```
    "resourceType": "Instance",
    "createdAt": 1569874760.201,
    "location": {
      "availabilityZone": "us-west-2a",
      "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationType": "DeleteKnownHostKeys",
    "status": "Succeeded",
    "statusChangedAt": 1569874760.201
  }
]
```

有关更多信息，请参阅 [Lightsail 开发者指南中的解决基于亚马逊 Lightsail 浏览器的 SSH 或 RDP 客户端的连接问题](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteKnownHostKeys](#) 中的。

## delete-load-balancer-tls-certificate

以下代码示例演示了如何使用 `delete-load-balancer-tls-certificate`。

### AWS CLI

删除负载均衡器的 TLS 证书

以下 `delete-load-balancer-tls-certificate` 示例从指定的负载均衡器中删除指定的 TLS 证书。

```
aws lightsail delete-load-balancer-tls-certificate \
  --load-balancer-name MyFirstLoadBalancer \
  --certificate-name MyFirstCertificate
```

输出：

```
{
  "operations": [
    {
      "id": "50bec274-e45e-4caa-8a69-b763ef636583",
      "resourceName": "MyFirstCertificate",
```

```

    "resourceType": "LoadBalancerTlsCertificate",
    "createdAt": 1569874989.48,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationType": "DeleteLoadBalancerTlsCertificate",
    "status": "Started",
    "statusChangedAt": 1569874989.48
  },
  {
    "id": "78c58cdc-a59a-4b27-8213-500638634a8f",
    "resourceName": "MyFirstLoadBalancer",
    "resourceType": "LoadBalancer",
    "createdAt": 1569874989.48,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationType": "DeleteLoadBalancerTlsCertificate",
    "status": "Started",
    "statusChangedAt": 1569874989.48
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLoadBalancerTlsCertificate](#)中的。

## delete-load-balancer

以下代码示例演示了如何使用 delete-load-balancer。

### AWS CLI

#### 删除负载均衡器

以下delete-load-balancer示例删除指定的负载均衡器和所有关联的 TLS 证书。

```

aws lightsail delete-load-balancer \
  --load-balancer-name MyFirstLoadBalancer

```

输出：

```
{
  "operations": [
    {
      "id": "a8c968c7-72a3-4680-a714-af8f03eea535",
      "resourceName": "MyFirstLoadBalancer",
      "resourceType": "LoadBalancer",
      "createdAt": 1569875092.125,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      },
      "isTerminal": true,
      "operationType": "DeleteLoadBalancer",
      "status": "Succeeded",
      "statusChangedAt": 1569875092.125
    },
    {
      "id": "f91a29fc-8ce3-4e69-a227-ea70ca890bf5",
      "resourceName": "MySecondCertificate",
      "resourceType": "LoadBalancerTlsCertificate",
      "createdAt": 1569875091.938,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      },
      "isTerminal": false,
      "operationType": "DeleteLoadBalancerTlsCertificate",
      "status": "Started",
      "statusChangedAt": 1569875091.938
    },
    {
      "id": "cf64c060-154b-4eb4-ba57-84e2e41563d6",
      "resourceName": "MyFirstLoadBalancer",
      "resourceType": "LoadBalancer",
      "createdAt": 1569875091.94,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      },
      "isTerminal": false,
      "operationType": "DeleteLoadBalancerTlsCertificate",
      "status": "Started",
    }
  ]
}
```

```

        "statusChangedAt": 1569875091.94
      }
    ]
  }

```

有关更多信息，请参阅指南中的标题。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLoadBalancer](#)中的。

## delete-relational-database-snapshot

以下代码示例演示了如何使用 delete-relational-database-snapshot。

### AWS CLI

删除托管数据库的快照

以下delete-relational-database-snapshot示例删除托管数据库的指定快照。

```

aws lightsail delete-relational-database-snapshot \
  --relational-database-snapshot-name Database-Oregon-1-1566839359

```

输出：

```

{
  "operations": [
    {
      "id": "b99acae8-735b-4823-922f-30af580e3729",
      "resourceName": "Database-Oregon-1-1566839359",
      "resourceType": "RelationalDatabaseSnapshot",
      "createdAt": 1569875293.58,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      },
      "isTerminal": true,
      "operationType": "DeleteRelationalDatabaseSnapshot",
      "status": "Succeeded",
      "statusChangedAt": 1569875293.58
    }
  ]
}

```



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRelationalDatabaseSnapshot](#)中的。

## delete-relational-database

以下代码示例演示了如何使用 delete-relational-database。

### AWS CLI

#### 删除托管数据库

以下delete-relational-database示例删除了指定的托管数据库。

```
aws lightsail delete-relational-database \  
  --relational-database-name Database-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "3b0c41c1-053d-46f0-92a3-14f76141dc86",  
      "resourceName": "Database-1",  
      "resourceType": "RelationalDatabase",  
      "createdAt": 1569875210.999,  
      "location": {  
        "availabilityZone": "us-west-2a",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationType": "DeleteRelationalDatabase",  
      "status": "Started",  
      "statusChangedAt": 1569875210.999  
    },  
    {  
      "id": "01ddeae8-a87a-4a4b-a1f3-092c71bf9180",  
      "resourceName": "Database-1",  
      "resourceType": "RelationalDatabase",  
      "createdAt": 1569875211.029,  
      "location": {  
        "availabilityZone": "us-west-2a",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,
```

```

    "operationDetails": "Database-1-FinalSnapshot-1569875210793",
    "operationType": "CreateRelationalDatabaseSnapshot",
    "status": "Started",
    "statusChangedAt": 1569875211.029
  },
  {
    "id": "74d73681-30e8-4532-974e-1f23cd3f9f73",
    "resourceName": "Database-1-FinalSnapshot-1569875210793",
    "resourceType": "RelationalDatabaseSnapshot",
    "createdAt": 1569875211.029,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": false,
    "operationDetails": "Database-1",
    "operationType": "CreateRelationalDatabaseSnapshot",
    "status": "Started",
    "statusChangedAt": 1569875211.029
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRelationalDatabase](#)中的。

## detach-static-ip

以下代码示例演示了如何使用 detach-static-ip。

### AWS CLI

将静态 IP 与实例分离

以下detach-static-ip示例将静态 IP 与任何连接StaticIp-1的实例分离。

```
aws lightsail detach-static-ip \
  --static-ip-name StaticIp-1
```

输出：

```
{
  "operations": [
    {
```

```
    "id": "2a43d8a3-9f2d-4fe7-bdd0-eEXAMPLE3cf3",
    "resourceName": "StaticIp-1",
    "resourceType": "StaticIp",
    "createdAt": 1571088261.999,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationDetails": "MEAN-1",
    "operationType": "DetachStaticIp",
    "status": "Succeeded",
    "statusChangedAt": 1571088261.999
  },
  {
    "id": "41a7d40c-74e8-4d2e-a837-cEXAMPLEf747",
    "resourceName": "MEAN-1",
    "resourceType": "Instance",
    "createdAt": 1571088262.022,
    "location": {
      "availabilityZone": "us-west-2a",
      "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationDetails": "StaticIp-1",
    "operationType": "DetachStaticIp",
    "status": "Succeeded",
    "statusChangedAt": 1571088262.022
  }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachStaticIp](#)中的。

## get-active-names

以下代码示例演示了如何使用 get-active-names。

### AWS CLI

获取活动资源名称

以下get-active-names示例返回已配置 AWS 区域中的活动资源名称。

```
aws lightsail get-active-names
```

输出：

```
{
  "activeNames": [
    "WordPress-1",
    "StaticIp-1",
    "MEAN-1",
    "Plesk_Hosting_Stack_on_Ubuntu-1"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetActiveNames](#)中的。

## get-auto-snapshots

以下代码示例演示了如何使用 get-auto-snapshots。

### AWS CLI

获取实例的可用自动快照

例如，以下get-auto-snapshots示例返回可用的自动快照WordPress-1。

```
aws lightsail get-auto-snapshots \
  --resource-name WordPress-1
```

输出：

```
{
  "resourceName": "WordPress-1",
  "resourceType": "Instance",
  "autoSnapshots": [
    {
      "date": "2019-10-14",
      "createdAt": 1571033872.0,
      "status": "Success",
      "fromAttachedDisks": []
    },
    {
      "date": "2019-10-13",
```

```
    "createdAt": 1570947473.0,
    "status": "Success",
    "fromAttachedDisks": []
  },
  {
    "date": "2019-10-12",
    "createdAt": 1570861072.0,
    "status": "Success",
    "fromAttachedDisks": []
  },
  {
    "date": "2019-10-11",
    "createdAt": 1570774672.0,
    "status": "Success",
    "fromAttachedDisks": []
  }
]
}
```

有关更多信息，请参阅 [Lightsail 开发者指南中的在 Amazon Lightsail 中保存实例或磁盘的自动快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetAutoSnapshots](#) 中的。

## get-blueprints

以下代码示例演示了如何使用 get-blueprints。

### AWS CLI

#### 获取新实例的蓝图

以下 get-blueprints 示例显示了可用于在 Amazon Lightsail 中创建新实例的所有可用蓝图的详细信息。

```
aws lightsail get-blueprints
```

输出：

```
{
  "blueprints": [
    {
      "blueprintId": "wordpress",
```

```
    "name": "WordPress",
    "group": "wordpress",
    "type": "app",
    "description": "Bitnami, the leaders in application packaging, and Automattic, the experts behind WordPress, have teamed up to offer this official WordPress image. This image is a pre-configured, ready-to-run image for running WordPress on Amazon Lightsail. WordPress is the world's most popular content management platform. Whether it's for an enterprise or small business website, or a personal or corporate blog, content authors can easily create content using its new Gutenberg editor, and developers can extend the base platform with additional features. Popular plugins like Jetpack, Akismet, All in One SEO Pack, WP Mail, Google Analytics for WordPress, and Amazon Polly are all pre-installed in this image. Let's Encrypt SSL certificates are supported through an auto-configuration script.",
    "isActive": true,
    "minPower": 0,
    "version": "5.2.2-3",
    "versionCode": "1",
    "productUrl": "https://aws.amazon.com/marketplace/pp/B00NN8Y43U",
    "licenseUrl": "https://d7umqicpi7263.cloudfront.net/eula/product/7d426cb7-9522-4dd7-a56b-55dd8cc1c8d0/588fd495-6492-4610-b3e8-d15ce864454c.txt",
    "platform": "LINUX_UNIX"
  },
  {
    "blueprintId": "lamp_7_1_28",
    "name": "LAMP (PHP 7)",
    "group": "lamp_7",
    "type": "app",
    "description": "LAMP with PHP 7.x certified by Bitnami greatly simplifies the development and deployment of PHP applications. It includes the latest versions of PHP 7.x, Apache and MySQL together with phpMyAdmin and popular PHP frameworks Zend, Symfony, CodeIgniter, CakePHP, Smarty, and Laravel. Other pre-configured components and PHP modules include FastCGI, ModSecurity, SQLite, Varnish, ImageMagick, xDebug, Xcache, OpenLDAP, Memcache, OAuth, PEAR, PECL, APC, GD and cURL. It is secure by default and supports multiple applications, each with its own virtual host and project directory. Let's Encrypt SSL certificates are supported through an auto-configuration script.",
    "isActive": true,
    "minPower": 0,
    "version": "7.1.28",
    "versionCode": "1",
    "productUrl": "https://aws.amazon.com/marketplace/pp/B072JNJZ5C",
```

```

        "licenseUrl": "https://d7umqicpi7263.cloudfront.net/eula/product/
cb6afd05-a3b2-4916-a3e6-bccd414f5f21/12ab56cc-6a8c-4977-9611-dcd770824aad.txt",
        "platform": "LINUX_UNIX"
    },
    {
        "blueprintId": "nodejs",
        "name": "Node.js",
        "group": "node",
        "type": "app",
        "description": "Node.js certified by Bitnami is a pre-configured, ready
to run image for Node.js on Amazon EC2. It includes the latest version of Node.js,
Apache, Python and Redis. The image supports multiple Node.js applications, each
with its own virtual host and project directory. It is configured for production
use and is secure by default, as all ports except HTTP, HTTPS and SSH ports are
closed. Let's Encrypt SSL certificates are supported through an auto-configuration
script. Developers benefit from instant access to a secure, update and consistent
Node.js environment without having to manually install and configure multiple
components and libraries.",
        "isActive": true,
        "minPower": 0,
        "version": "12.7.0",
        "versionCode": "1",
        "productUrl": "https://aws.amazon.com/marketplace/pp/B00NNZUAKO",
        "licenseUrl": "https://d7umqicpi7263.cloudfront.net/
eula/product/033793fe-951d-47d0-aa94-5fbd0afb3582/25f8fa66-c868-4d80-
adf8-4a2b602064ae.txt",
        "platform": "LINUX_UNIX"
    },
    ...
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBlueprints](#)中的。

## get-bundles

以下代码示例演示了如何使用 get-bundles。

### AWS CLI

#### 获取新实例的捆绑包

以下`get-bundles`示例显示了可用于在 Amazon Lightsail 中创建新实例的所有可用捆绑包的详细信息。

```
aws lightsail get-bundles
```

输出：

```
{
  "bundles": [
    {
      "price": 3.5,
      "cpuCount": 1,
      "diskSizeInGb": 20,
      "bundleId": "nano_2_0",
      "instanceType": "nano",
      "isActive": true,
      "name": "Nano",
      "power": 300,
      "ramSizeInGb": 0.5,
      "transferPerMonthInGb": 1024,
      "supportedPlatforms": [
        "LINUX_UNIX"
      ]
    },
    {
      "price": 5.0,
      "cpuCount": 1,
      "diskSizeInGb": 40,
      "bundleId": "micro_2_0",
      "instanceType": "micro",
      "isActive": true,
      "name": "Micro",
      "power": 500,
      "ramSizeInGb": 1.0,
      "transferPerMonthInGb": 2048,
      "supportedPlatforms": [
        "LINUX_UNIX"
      ]
    },
    {
      "price": 10.0,
      "cpuCount": 1,
      "diskSizeInGb": 60,
```



```

        "bundleId": "small_2_0",
        "instanceType": "small",
        "isActive": true,
        "name": "Small",
        "power": 1000,
        "ramSizeInGb": 2.0,
        "transferPerMonthInGb": 3072,
        "supportedPlatforms": [
            "LINUX_UNIX"
        ]
    },
    ...
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBundles](#)中的。

## get-cloud-formation-stack-records

以下代码示例演示了如何使用 `get-cloud-formation-stack-records`。

### AWS CLI

获取 CloudFormation 堆栈记录及其关联堆栈

以下`get-cloud-formation-stack-records`示例显示了有关 CloudFormation 堆栈记录及其关联堆栈的详细信息，这些堆栈用于根据导出的 Amazon Lightsail 快照创建 Amazon EC2 资源。

```
aws lightsail get-cloud-formation-stack-records
```

输出：

```

{
  "cloudFormationStackRecords": [
    {
      "name": "CloudFormationStackRecord-588a4243-
e2d1-490d-8200-3a7513ecebdf",
      "arn": "arn:aws:lightsail:us-
west-2:111122223333:CloudFormationStackRecord/28d646ab-27bc-48d9-a422-1EXAMPLE6d37",
      "createdAt": 1565301666.586,
      "location": {

```

```

        "availabilityZone": "all",
        "regionName": "us-west-2"
    },
    "resourceType": "CloudFormationStackRecord",
    "state": "Succeeded",
    "sourceInfo": [
        {
            "resourceType": "ExportSnapshotRecord",
            "name": "ExportSnapshotRecord-
e02f23d7-0453-4aa9-9c95-91aa01a141dd",
            "arn": "arn:aws:lightsail:us-
west-2:111122223333:ExportSnapshotRecord/f12b8792-f3ea-4d6f-b547-2EXAMPLE8796"
        }
    ],
    "destinationInfo": {
        "id": "arn:aws:cloudformation:us-west-2:111122223333:stack/
Lightsail-Stack-588a4243-e2d1-490d-8200-3EXAMPLEebdf/063203b0-
ba28-11e9-838b-0EXAMPLE8b00",
        "service": "Aws::CloudFormation::Stack"
    }
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCloudFormationStackRecords](#)中的。

## get-disk-snapshot

以下代码示例演示了如何使用 `get-disk-snapshot`。

### AWS CLI

获取有关磁盘快照的信息

以下`get-disk-snapshot`示例显示了有关磁盘快照的详细信息Disk-1-1566839161。

```
aws lightsail get-disk-snapshot \
  --disk-snapshot-name Disk-1-1566839161
```

输出：

```
{
```

```
    "diskSnapshot": {
      "name": "Disk-1-1566839161",
      "arn": "arn:aws:lightsail:us-west-2:111122223333:DiskSnapshot/
e2d0fa53-8ee0-41a0-8e56-0EXAMPLE1051",
      "supportCode": "6EXAMPLE3362/snap-0EXAMPLE06100d09",
      "createdAt": 1566839163.749,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      },
      "resourceType": "DiskSnapshot",
      "tags": [],
      "sizeInGb": 8,
      "state": "completed",
      "progress": "100%",
      "fromDiskName": "Disk-1",
      "fromDiskArn": "arn:aws:lightsail:us-west-2:111122223333:Disk/
c21cfb0a-07f2-44ae-9a23-bEXAMPLE8096",
      "isFromAutoSnapshot": false
    }
  }
}
```

有关更多信息，请参阅指南中的标题。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDiskSnapshot](#)中的。

## get-disk-snapshots

以下代码示例演示了如何使用 get-disk-snapshots。

### AWS CLI

获取有关所有磁盘快照的信息

以下get-disk-snapshots示例显示了有关已配置 AWS 区域中所有磁盘快照的详细信息。

```
aws lightsail get-disk-snapshots
```

输出：

```
{
  "diskSnapshots": [
```

```
{
  "name": "Disk-2-1571090588",
  "arn": "arn:aws:lightsail:us-
west-2:111122223333:DiskSnapshot/32e889a9-38d4-4687-9f21-eEXAMPLE7839",
  "supportCode": "6EXAMPLE3362/snap-0EXAMPLE1ca192a4",
  "createdAt": 1571090591.226,
  "location": {
    "availabilityZone": "all",
    "regionName": "us-west-2"
  },
  "resourceType": "DiskSnapshot",
  "tags": [],
  "sizeInGb": 8,
  "state": "completed",
  "progress": "100%",
  "fromDiskName": "Disk-2",
  "fromDiskArn": "arn:aws:lightsail:us-
west-2:111122223333:Disk/6a343ff8-6341-422d-86e2-bEXAMPLE16c2",
  "isFromAutoSnapshot": false
},
{
  "name": "Disk-1-1566839161",
  "arn": "arn:aws:lightsail:us-west-2:111122223333:DiskSnapshot/
e2d0fa53-8ee0-41a0-8e56-0EXAMPLE1051",
  "supportCode": "6EXAMPLE3362/snap-0EXAMPLEe06100d09",
  "createdAt": 1566839163.749,
  "location": {
    "availabilityZone": "all",
    "regionName": "us-west-2"
  },
  "resourceType": "DiskSnapshot",
  "tags": [],
  "sizeInGb": 8,
  "state": "completed",
  "progress": "100%",
  "fromDiskName": "Disk-1",
  "fromDiskArn": "arn:aws:lightsail:us-west-2:111122223333:Disk/
c21cfb0a-07f2-44ae-9a23-bEXAMPLE8096",
  "isFromAutoSnapshot": false
}
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDiskSnapshots](#)中的。

## get-disk

以下代码示例演示了如何使用 `get-disk`。

### AWS CLI

获取有关块存储磁盘的信息

以下 `get-disk` 示例显示了有关磁盘的详细信息 `Disk-1`。

```
aws lightsail get-disk \  
  --disk-name Disk-1
```

输出：

```
{  
  "disk": {  
    "name": "Disk-1",  
    "arn": "arn:aws:lightsail:us-west-2:111122223333:Disk/  
c21cfb0a-07f2-44ae-9a23-bEXAMPLE8096",  
    "supportCode": "6EXAMPLE3362/vol-0EXAMPLEf2f88b32f",  
    "createdAt": 1566585439.587,  
    "location": {  
      "availabilityZone": "us-west-2a",  
      "regionName": "us-west-2"  
    },  
    "resourceType": "Disk",  
    "tags": [],  
    "sizeInGb": 8,  
    "isSystemDisk": false,  
    "iops": 100,  
    "path": "/dev/xvdf",  
    "state": "in-use",  
    "attachedTo": "WordPress_Multisite-1",  
    "isAttached": true,  
    "attachmentState": "attached"  
  }  
}
```

有关更多信息，请参阅指南中的标题。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDisk](#) 中的。

## get-disks

以下代码示例演示了如何使用 get-disks。

### AWS CLI

获取有关所有块存储磁盘的信息

以下get-disks示例显示了有关已配置 AWS 区域中所有磁盘的详细信息。

```
aws lightsail get-disks
```

输出：

```
{
  "disks": [
    {
      "name": "Disk-2",
      "arn": "arn:aws:lightsail:us-west-2:111122223333:Disk/6a343ff8-6341-422d-86e2-bEXAMPLE16c2",
      "supportCode": "6EXAMPLE3362/vol-0EXAMPLE929602087",
      "createdAt": 1571090461.634,
      "location": {
        "availabilityZone": "us-west-2a",
        "regionName": "us-west-2"
      },
      "resourceType": "Disk",
      "tags": [],
      "sizeInGb": 8,
      "isSystemDisk": false,
      "iops": 100,
      "state": "available",
      "isAttached": false,
      "attachmentState": "detached"
    },
    {
      "name": "Disk-1",
      "arn": "arn:aws:lightsail:us-west-2:111122223333:Disk/c21cfb0a-07f2-44ae-9a23-bEXAMPLE8096",
      "supportCode": "6EXAMPLE3362/vol-0EXAMPLEf2f88b32f",
      "createdAt": 1566585439.587,
      "location": {
        "availabilityZone": "us-west-2a",
        "regionName": "us-west-2"
      }
    }
  ]
}
```

```
    },
    "resourceType": "Disk",
    "tags": [],
    "sizeInGb": 8,
    "isSystemDisk": false,
    "iops": 100,
    "path": "/dev/xvdf",
    "state": "in-use",
    "attachedTo": "WordPress_Multisite-1",
    "isAttached": true,
    "attachmentState": "attached"
  }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDisks](#)中的。

## get-domain

以下代码示例演示了如何使用 `get-domain`。

### AWS CLI

获取有关域名的信息

以下`get-domain`示例显示有关该域的详细信息`example.com`。

**注意：** Lightsail 与域名相关的 API 操作仅在该地区可用。us-east-1 AWS 如果您的 CLI 配置文件配置为使用其他区域，则必须包含“`--region us-east-1``”参数，否则命令将失败。

```
aws lightsail get-domain \
  --domain-name example.com \
  --region us-east-1
```

输出：

```
{
  "domain": {
    "name": "example.com",
    "arn":
      "arn:aws:lightsail:global:111122223333:Domain/28cda903-3f15-44b2-9baf-3EXAMPLEb304",
    "supportCode": "6EXAMPLE3362//hostedzone/ZEXAMPLEONGSC1",
    "createdAt": 1570728588.6,
  }
}
```

```
"location": {
  "availabilityZone": "all",
  "regionName": "global"
},
"resourceType": "Domain",
"tags": [],
"domainEntries": [
  {
    "id": "-1682899164",
    "name": "example.com",
    "target": "192.0.2.0",
    "isAlias": false,
    "type": "A"
  },
  {
    "id": "1703104243",
    "name": "example.com",
    "target": "ns-137.awsdns-17.com",
    "isAlias": false,
    "type": "NS"
  },
  {
    "id": "-1038331153",
    "name": "example.com",
    "target": "ns-1710.awsdns-21.co.uk",
    "isAlias": false,
    "type": "NS"
  },
  {
    "id": "-2107289565",
    "name": "example.com",
    "target": "ns-692.awsdns-22.net",
    "isAlias": false,
    "type": "NS"
  },
  {
    "id": "1582095705",
    "name": "example.com",
    "target": "ns-1436.awsdns-51.org",
    "isAlias": false,
    "type": "NS"
  },
  {
    "id": "-1769796132",
```



```

        "name": "example.com",
        "target": "ns-1710.awsdns-21.co.uk. awsdns-hostmaster.amazon.com. 1
7200 900 1209600 86400",
        "isAlias": false,
        "type": "SOA"
    }
]
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDomain](#)中的。

## get-domains

以下代码示例演示了如何使用 get-domains。

### AWS CLI

获取有关所有域的信息

以下get-domains示例显示了有关已配置 AWS 区域中所有域的详细信息。

注意：Lightsail 与域名相关的 API 操作仅在该地区可用。us-east-1 AWS 如果您的 CLI 配置文件配置为使用其他区域，则必须包含--region us-east-1参数，否则命令将失败。

```
aws lightsail get-domains \
  --region us-east-1
```

输出：

```

{
  "domains": [
    {
      "name": "example.com",
      "arn":
"arn:aws:lightsail:global:111122223333:Domain/28cda903-3f15-44b2-9baf-3EXAMPLEb304",
      "supportCode": "6EXAMPLE3362//hostedzone/ZEXAMPLEONGSC1",
      "createdAt": 1570728588.6,
      "location": {
        "availabilityZone": "all",
        "regionName": "global"
      },
      "resourceType": "Domain",
    }
  ]
}

```

```
"tags": [],
"domainEntries": [
  {
    "id": "-1682899164",
    "name": "example.com",
    "target": "192.0.2.0",
    "isAlias": false,
    "type": "A"
  },
  {
    "id": "1703104243",
    "name": "example.com",
    "target": "ns-137.awsdns-17.com",
    "isAlias": false,
    "type": "NS"
  },
  {
    "id": "-1038331153",
    "name": "example.com",
    "target": "ns-4567.awsdns-21.co.uk",
    "isAlias": false,
    "type": "NS"
  },
  {
    "id": "-2107289565",
    "name": "example.com",
    "target": "ns-333.awsdns-22.net",
    "isAlias": false,
    "type": "NS"
  },
  {
    "id": "1582095705",
    "name": "example.com",
    "target": "ns-1111.awsdns-51.org",
    "isAlias": false,
    "type": "NS"
  },
  {
    "id": "-1769796132",
    "name": "example.com",
    "target": "ns-1234.awsdns-21.co.uk. awsdns-
hostmaster.amazon.com. 1 7200 900 1209600 86400",
    "isAlias": false,
    "type": "SOA"
  }
]
```

```
    },
    {
      "id": "1029454894",
      "name": "_dead6a124ede046a0319eb44a4eb3cbc.example.com",
      "target": "_be133b0a0899fb7b6bf79d9741d1a383.hkvuiqjoua.acm-
validations.aws",
      "isAlias": false,
      "type": "CNAME"
    }
  ]
},
{
  "name": "example.net",
  "arn": "arn:aws:lightsail:global:111122223333:Domain/9c9f0d70-
c92e-4753-86c2-6EXAMPLE029d",
  "supportCode": "6EXAMPLE3362//hostedzone/ZEXAMPLE5TPKMV",
  "createdAt": 1556661071.384,
  "location": {
    "availabilityZone": "all",
    "regionName": "global"
  },
  "resourceType": "Domain",
  "tags": [],
  "domainEntries": [
    {
      "id": "-766320943",
      "name": "example.net",
      "target": "192.0.2.2",
      "isAlias": false,
      "type": "A"
    },
    {
      "id": "-453913825",
      "name": "example.net",
      "target": "ns-123.awsdns-10.net",
      "isAlias": false,
      "type": "NS"
    },
    {
      "id": "1553601564",
      "name": "example.net",
      "target": "ns-4444.awsdns-47.co.uk",
      "isAlias": false,
      "type": "NS"
    }
  ]
}
```

```
    },
    {
      "id": "1653797661",
      "name": "example.net",
      "target": "ns-7890.awsdns-61.org",
      "isAlias": false,
      "type": "NS"
    },
    {
      "id": "706414698",
      "name": "example.net",
      "target": "ns-123.awsdns-44.com",
      "isAlias": false,
      "type": "NS"
    },
    {
      "id": "337271745",
      "name": "example.net",
      "target": "ns-4444.awsdns-47.co.uk. awsdns-
hostmaster.amazon.com. 1 7200 900 1209600 86400",
      "isAlias": false,
      "type": "SOA"
    },
    {
      "id": "-1785431096",
      "name": "www.example.net",
      "target": "192.0.2.2",
      "isAlias": false,
      "type": "A"
    }
  ]
},
{
  "name": "example.org",
  "arn": "arn:aws:lightsail:global:111122223333:Domain/
f0f13ba3-3df0-4fdc-8ebb-1EXAMPLEf26e",
  "supportCode": "6EXAMPLE3362//hostedzone/ZEXAMPLEAF038",
  "createdAt": 1556661199.106,
  "location": {
    "availabilityZone": "all",
    "regionName": "global"
  },
  "resourceType": "Domain",
  "tags": [],
```

```
"domainEntries": [  
  {  
    "id": "2065301345",  
    "name": "example.org",  
    "target": "192.0.2.4",  
    "isAlias": false,  
    "type": "A"  
  },  
  {  
    "id": "-447198516",  
    "name": "example.org",  
    "target": "ns-123.awsdns-45.com",  
    "isAlias": false,  
    "type": "NS"  
  },  
  {  
    "id": "136463022",  
    "name": "example.org",  
    "target": "ns-9999.awsdns-15.co.uk",  
    "isAlias": false,  
    "type": "NS"  
  },  
  {  
    "id": "1395941679",  
    "name": "example.org",  
    "target": "ns-555.awsdns-01.net",  
    "isAlias": false,  
    "type": "NS"  
  },  
  {  
    "id": "872052569",  
    "name": "example.org",  
    "target": "ns-6543.awsdns-38.org",  
    "isAlias": false,  
    "type": "NS"  
  },  
  {  
    "id": "1001949377",  
    "name": "example.org",  
    "target": "ns-1234.awsdns-15.co.uk. awsdns-  
hostmaster.amazon.com. 1 7200 900 1209600 86400",  
    "isAlias": false,  
    "type": "SOA"  
  },  
]
```

```
    {
      "id": "1046191192",
      "name": "www.example.org",
      "target": "192.0.2.4",
      "isAlias": false,
      "type": "A"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDomains](#)中的。

## get-export-snapshot-record

以下代码示例演示了如何使用 `get-export-snapshot-record`。

### AWS CLI

获取导出到 Amazon EC2 的快照记录

以下`get-export-snapshot-record`示例显示了有关导出到亚马逊 EC2 的 Amazon Lightsail 实例或磁盘快照的详细信息。

```
aws lightsail get-export-snapshot-records
```

输出：

```
{
  "exportSnapshotRecords": [
    {
      "name": "ExportSnapshotRecord-d2da10ce-0b3c-4ae1-ab3a-2EXAMPLEa586",
      "arn": "arn:aws:lightsail:us-west-2:111122223333:ExportSnapshotRecord/076c7060-b0cc-4162-98f0-2EXAMPLEe28e",
      "createdAt": 1543534665.678,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      },
      "resourceType": "ExportSnapshotRecord",
      "state": "Succeeded",
    }
  ]
}
```

```

    "sourceInfo": {
      "resourceType": "InstanceSnapshot",
      "createdAt": 1540339310.706,
      "name": "WordPress-512MB-0regon-1-1540339219",
      "arn": "arn:aws:lightsail:us-
west-2:111122223333:InstanceSnapshot/5446f534-ed60-4c17-b4a5-bEXAMPLEf8b7",
      "fromResourceName": "WordPress-512MB-0regon-1",
      "fromResourceArn": "arn:aws:lightsail:us-
west-2:111122223333:Instance/4b8f1f24-e4d1-4cf3-88ff-cEXAMPLEa397",
      "instanceSnapshotInfo": {
        "fromBundleId": "nano_2_0",
        "fromBlueprintId": "wordpress_4_9_8",
        "fromDiskInfo": [
          {
            "path": "/dev/sda1",
            "sizeInGb": 20,
            "isSystemDisk": true
          }
        ]
      }
    },
    "destinationInfo": {
      "id": "ami-0EXAMPLEc0d65058e",
      "service": "Aws::EC2::Image"
    }
  },
  {
    "name": "ExportSnapshotRecord-1c94e884-40ff-4fe1-9302-0EXAMPLE14c2",
    "arn": "arn:aws:lightsail:us-west-2:111122223333:ExportSnapshotRecord/
fb392ce8-6567-4013-9bfd-3EXAMPLE5b4c",
    "createdAt": 1543432110.2,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "resourceType": "ExportSnapshotRecord",
    "state": "Succeeded",
    "sourceInfo": {
      "resourceType": "InstanceSnapshot",
      "createdAt": 1540833603.545,
      "name": "LAMP_PHP_5-512MB-0regon-1-1540833565",
      "arn": "arn:aws:lightsail:us-
west-2:111122223333:InstanceSnapshot/82334399-b5f2-49ec-8382-0EXAMPLEe45f",
      "fromResourceName": "LAMP_PHP_5-512MB-0regon-1",

```

```

    "fromResourceArn": "arn:aws:lightsail:us-
west-2:111122223333:Instance/863b9f35-ab1e-4418-bdd2-1EXAMPLEbab2",
    "instanceSnapshotInfo": {
      "fromBundleId": "nano_2_0",
      "fromBlueprintId": "lamp_5_6_37_2",
      "fromDiskInfo": [
        {
          "path": "/dev/sda1",
          "sizeInGb": 20,
          "isSystemDisk": true
        }
      ]
    },
    "destinationInfo": {
      "id": "ami-0EXAMPLE7c5ec84e2",
      "service": "Aws::EC2::Image"
    }
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetExportSnapshotRecord](#)中的。

## get-instance-access-details

以下代码示例演示了如何使用 `get-instance-access-details`。

### AWS CLI

获取实例的主机密钥信息

例如，以下`get-instance-access-details`示例显示了主机密钥信息WordPress\_Multisite-1。

```
aws lightsail get-instance-access-details \
  --instance-name WordPress_Multisite-1
```

输出：

```
{
  "accessDetails": {
```



```

    "certKey": "ssh-rsa-cert-v01@openssh.com
    AEXAMPLEaC1yc2EtY2VydC12MDFab3B1bnNzaC5jb20AAAAGNf076Dt3ppmPd0fPxZVMmS491aEAYYH9cHqAJ3fNML8
vEXAMPLE2eBWJyQvn7o1/
i0+s966h5sx8qUD791PB7q5UESd5VZGFtytrykfQJnjiwqe7EV5agzvjb1Lj26Fb37EKda9HVfCOu8pWbvky7Tyn9w29
+xMfQM9xVz0rXZmqx8uJidJpRgLCMTviofwQJU/
K1EXAMPLEAAAAAAAAABAAAALS0MzMzMdu4MzA40Dg1MTY2NjM40np6UWlndHk4UElRSG9Stit0TG5QSEE9PQAAAAAsAAA
+LiB+ozNbUA0cdNL9Y67x7qPv/R7XhTc21+2A+8+GuVpK/Kz9dqDMKNAEXAMPLE+YYN
+tiXm7Y80gziK+7iDB7xUuQ4vghmn4+qgz9mKwYgWvVe2+0XLuV7cnWPB7iUlHQg
+E3LUKrV4ZFw9pj7X2dFdNKfMxwWgI1ISWKimEXAMPLEeHjrf1Rqc/
QH6TpWcVpfcx8uvwVqdwTfke/SfA5BCzbGGI1UmIUadh8nHcb5FamQ1hK7kECy47K/x9FMn/
KwmM7pCwJbSLDM07n9bnbvck6m8ZoB2N2YLMG5dW7BerEXAMPLEeobqfdtyYJHHe11EyyEJs1fWNU3D5JIGlgzcPAV
+Z1bQyUCZXf0os1Sa+HE85f0/
FRq9SVSBSHrmb0fr1PhgMzgSmqLeyhlbr6wwWIDbREXAMPLEJZ49H7RdQxdKyYrZPwvRgcr0qI2EL0tAajnpQQ8UZo
Aqter0xN5PhFL0J490WTacwCGRAjLhibAx7K1t/1ZXWo6c+ijq8c1l1327EXAMPLE/
e89GC89KcmKCxfGQniDAUgF8UqofIbq3Z0UgiAAYCVXc1I4L68NhVXyoWuQXPBRQSEXAMPLEWm74tDL9tFN3c7tSe/
Oz0cTR+4sAAAIPAAAAB3NzaC1yc2EAAAIAQnG/
L0DqiSnLrWhEox4aHqMgd0m0oLLAYx60QH9F0TM9EXAMPLE961rzSCMon7ZgswNnL00wZQgDG
+rtJ4N0B7H0Vwns4ynUFbzNq3qFGGeE31KwX1L41vV1iSy7sDk8aI0LmrKJi1LE1Qc1l8uborLwoX0YEXAMPLEaUCeX
+10+WEXAMPLEeg6Y4U4ZvE2B3xyRdpvysb5TGFntk5qPslacnVkoL0GsZZXmpLGJnG40BpQLLtpj9sNMxAgZPCAUjhkqk
+nx0904NUZ2pTwbVSUaV1gm6pug9xbwN01Im21t34JeLlKTqxcJ6zzS8W0c0KKpAm5c4hWkseMbyutS2jav/4hiS
+BhrYgptzfwe5qRXEXAMPLEEHZQr3YfGzYoBJ/
lLK3NHhx0ihhsfAYwMei0BFZT1F/7CT3IH4iitEkIgodio6/
Mw6UDqMPozyQCK11EA6LFhYCOZG9drWcoRa741M4ky9TP028Za8gDMh1WpkXLq9Gixon50HP8aM/
sEXAMPLEr2+fnkw+1Bto05L6+VKoPlXaGqZ/fBYEXAMPLEAMQHjnLM1JYNvtEEPhp+TNzXHzuixWf/
Ht04m0AVpXrzIDXaS102tXY=",
    "ipAddress": "192.0.2.0",
    "privateKey": "-----BEGIN RSA PRIVATE KEY-----
\nEXAMPLEBAAKCAQEA+AD3qeU2toBy505v7wnRLVo/tngVickL5+6Jf4tPrPeuoebM
\nfK1A+/ZTwe6uVBEneVWRhbcra8pH0CZ44sKnuxFeWoM7425S49uhW9+xCnWvR1Xw
\njrvKVm75Mu08p/cNvfWugrBuaPB65DspgxNn0fZWMVxpIpSq0SPWmSwQHV597d6C
\nrEXAMPLEe08hJmqz2KFQ09X7fB21BruGgr9aXiNPmWmovYKqwFmrnFvR7odFmDecq
\n5EXAMPLE9dyU1ZsrWhGby77eYrVaF10GNGQ8qy1HGUiscquZ9NDIL49n4mXbfsTH
\n0EXAMPLE12ZqsfliYnSaUYCwjE74qH8ECVPytQIDAQABaoIBAHeZV9Z58JHAjifz
\nCEXAMPLEEqC3do0VDgXS1kKI92qNo4z2VcUEho878paCuVvXVHCcGgSnGeyIh2tN
\nMEXAMPLESohR427BhH3YLA+3Z5SivnejbTgYPfLC37B8khTaYqkqMvdZiFVZK5qn
\nIEXAMPLEM93oF9eSZCjclKB/jGHsfb0eCDMP8BshHE2beuqzVMoK1Dx0nvoP3+Fp
\nAEXAMPLESg6pDpCo9YVUX8g1u3Ro9cP12LXHDy+oVEY5KhbZQJ7VU1I72W0vppWW
\n0EXAMPLEkgY1q7p6qYtYcSgTEjz14gDiMfQ7SyHB3alkIoN0NQ9ZPaWHyJvymeud
\noQTNuz0CgYEA/LFWNTEZrzdZdR1kJmyNRmAermU0B6utyNENChAlHGSHkB+11VSh
\nbEXAMPLEQo9ooUeW5Ux03YwacZLoDT1mwxw1Ptc1+PNycZoLe1fE9UdARrdmGTob
\n817CPLSXp3xuR8VqSp2fnIc7hfiQs/NrPX9gm/E0rB0we0RKyDSzWScGyEA+z/r
\niob+nJZq0Ybn0SuP6oMULP4vnWniWj8MIhUJU53LwSAM8DeJd0NKDdkui0d52aAL
\nVgn7nLo88rVwKhJwVc4tu/rNgZLcR3bP4+kL6zand0KQnMLy0zNA2Ys26aa5udH1\nnqw10WTt9WEm/
h10ndC1kn0MectrvsG17b38y5sMCgYEA54NiRGGz8oCPW6GN/FZA

```

```

\nKEXAMPLE5tw34GEH3Uxlcn3CejDaQmcz0ATwX4nIwRZDEqWyYZcS0btg1jhGiBD\nYEXAMPLEkc8Z71L/
agZEAaVCEog9FqfSqwB
+XTfoKh8qur74X1yCu9p6gof1q6k9\nEXAMPLEchJcNN0g4ETIfMkCgYBdVORRHE4mqvWp0dzA7v66FdEz2YSkjAXKK
\naEXAMPLE8Z/8yBSmuBv1Qv03XA12my462uB92uzzGAuW
+1yBc2Kn1sXqYTy0y1z0\nngEXAMPLEBogjw4MqHKL1bPKMHYQU8/
q24PaYgzHPzy13wLH6pTYf1XqLHdE2D6Vv\nnyEXAMPLEgQC3i/
kVVhky/2XRwRV1C7J02Bg3QGTx38hpmDa5IuofKANjA+Wa3/zy\nnbEXAMPLE6ytQgD9GN/YtBq+uh0
+2ZkvXPL+CWRi0ZRXPwYDBBFU9Cw0AuWWG1L8\nnwEXAMPLExMlcysRgcWB9RNgf3Au0pFd2i6XT/
riNsvvKpmJ+VooU8g==\n-----END RSA PRIVATE KEY-----\n",
    "protocol": "ssh",
    "instanceName": "WordPress_Multisite-1",
    "username": "bitnami",
    "hostKeys": [
      {
        "algorithm": "ssh-rsa",
        "publicKey":
"AEXAMPLEaC1yc2EAAAADAQABAAQCoer9ieZTjQ3pXCHczuAYZFj1F7t
+uBkXuqeGMRex78pCvmS+DiEXAMPLEeuJ1Q8dcKhrQL4HpXbD9dosVCTaJnJwb4MQqsuSVFdhFzy3guP
+BKclWqtXJEXAMPLEsBGqZz1rIv6a9bTA0TCplZ8AD+hSRTaSXXqg6FT
+Qf16IktH0X1Ms7xIEXAMPLEmNtjCpzZiGXDHzytoMvUgwa8uHPp440g36EUu4VqQxoUHPJKoXvcQizyk3K8ym0hP0Tp
0t6y9HwvykEXAMPLEAfbKjbr42+u6+0S1kr4d339q2U1sTDytJhhs8HUel1wTfGRfp",
        "witnessedAt": 1570744377.699,
        "fingerprintSHA1": "SHA1:GEXAMPLEMoYgUg0ucadqU9Bt3Lk",
        "fingerprintSHA256": "SHA256:IEEXAMPLEcB5vgxnAUoJawbdZ
+MwELhIp6FUxuwq/LIU"
      },
      {
        "algorithm": "ssh-ed25519",
        "publicKey":
"AEXAMPLEaC1lZDI1NTE5AAAAIC1gwGPDfGa0NxEXAMPLEJX3UNap781QxHQmn8nzlrUv",
        "witnessedAt": 1570744377.697,
        "fingerprintSHA1": "SHA1:VEXAMPLE5ReqSmTgv03sSUw9toU",
        "fingerprintSHA256": "SHA256:0EXAMPLEdE6tI95k3TJpG
+qhJbAoknB0yz9nAEaDt3A"
      },
      {
        "algorithm": "ecdsa-sha2-nistp256",
        "publicKey":
"AEXAMPLEZHNhLXNoYTIitbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABEXAMPLE9B4mZy8YSsZW7cixCDq5yHSAAxjJkDo5
+EnK1DCsYtUkxxEXAMPLE6V0WL2z63RTKa2AUPgd8irjxWI=",
        "witnessedAt": 1570744377.707,
        "fingerprintSHA1": "SHA1:UEEXAMPLE0YCFxScf2G6tDg+7YG0",
        "fingerprintSHA256": "SHA256:wEXAMPLEQ9a/
iEXAMPLEhRufm6U9vFU4cPKMPHnBsNA"
      }
    ]
  }
}

```

```
    }
  ]
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInstanceAccessDetails](#)中的。

## get-instance-metric-data

以下代码示例演示了如何使用 `get-instance-metric-data`。

### AWS CLI

获取实例的指标数据

例如，以下`get-instance-metric-data`示例返回与之间CPUUtilization每7200秒（2小时）15713424001571428800的平均百分比MEAN-1。

我们建议您使用 unix 时间转换器来识别开始和结束时间。

```
aws lightsail get-instance-metric-data \
  --instance-name MEAN-1 \
  --metric-name CPUUtilization \
  --period 7200 \
  --start-time 1571342400 \
  --end-time 1571428800 \
  --unit Percent \
  --statistics Average
```

输出：

```
{
  "metricName": "CPUUtilization",
  "metricData": [
    {
      "average": 0.26113718770120725,
      "timestamp": 1571342400.0,
      "unit": "Percent"
    },
    {
      "average": 0.26861268928111953,
      "timestamp": 1571392800.0,
      "unit": "Percent"
    }
  ]
}
```

```
  },
  {
    "average": 0.28187475104748777,
    "timestamp": 1571378400.0,
    "unit": "Percent"
  },
  {
    "average": 0.2651936960458352,
    "timestamp": 1571421600.0,
    "unit": "Percent"
  },
  {
    "average": 0.2561856213712188,
    "timestamp": 1571371200.0,
    "unit": "Percent"
  },
  {
    "average": 0.3021383254607764,
    "timestamp": 1571356800.0,
    "unit": "Percent"
  },
  {
    "average": 0.2618381649223539,
    "timestamp": 1571407200.0,
    "unit": "Percent"
  },
  {
    "average": 0.26331929394825787,
    "timestamp": 1571400000.0,
    "unit": "Percent"
  },
  {
    "average": 0.2576348407007818,
    "timestamp": 1571385600.0,
    "unit": "Percent"
  },
  {
    "average": 0.2513008454658378,
    "timestamp": 1571364000.0,
    "unit": "Percent"
  },
  {
    "average": 0.26329974562758346,
    "timestamp": 1571414400.0,
```

```
        "unit": "Percent"
      },
      {
        "average": 0.2667092536656445,
        "timestamp": 1571349600.0,
        "unit": "Percent"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInstanceMetricData](#)中的。

## get-instance-port-states

以下代码示例演示了如何使用 `get-instance-port-states`。

### AWS CLI

获取实例的防火墙信息

例如，以下`get-instance-port-states`示例返回配置的防火墙端口MEAN-1。

```
aws lightsail get-instance-port-states \
  --instance-name MEAN-1
```

输出：

```
{
  "portStates": [
    {
      "fromPort": 80,
      "toPort": 80,
      "protocol": "tcp",
      "state": "open"
    },
    {
      "fromPort": 22,
      "toPort": 22,
      "protocol": "tcp",
      "state": "open"
    },
    {
      "fromPort": 443,
```

```

        "toPort": 443,
        "protocol": "tcp",
        "state": "open"
      }
    ]
  }

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInstancePortStates](#)中的。

## get-instance-snapshot

以下代码示例演示了如何使用 get-instance-snapshot。

### AWS CLI

获取有关指定实例快照的信息

以下get-instance-snapshot示例显示有关指定实例快照的详细信息。

```

aws lightsail get-instance-snapshot \
  --instance-snapshot-name MEAN-1-1571419854

```

输出：

```

{
  "instanceSnapshot": {
    "name": "MEAN-1-1571419854",
    "arn": "arn:aws:lightsail:us-west-2:111122223333:InstanceSnapshot/
ac54700c-48a8-40fd-b065-2EXAMPLEac8f",
    "supportCode": "6EXAMPLE3362/ami-0EXAMPLE67a73020d",
    "createdAt": 1571419891.927,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "resourceType": "InstanceSnapshot",
    "tags": [],
    "state": "available",
    "fromAttachedDisks": [],
    "fromInstanceName": "MEAN-1",
    "fromInstanceArn": "arn:aws:lightsail:us-west-2:111122223333:Instance/
bd470fc5-a68b-44c5-8dbc-8EXAMPLEebada",
    "fromBlueprintId": "mean_4_0_9",

```

```
    "fromBundleId": "medium_2_0",
    "isFromAutoSnapshot": false,
    "sizeInGb": 80
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInstanceSnapshot](#)中的。

## get-instance-snapshots

以下代码示例演示了如何使用 get-instance-snapshots。

### AWS CLI

获取有关您的所有实例快照的信息

以下get-instance-snapshots示例显示了有关已配置 AWS 区域中所有实例快照的详细信息。

```
aws lightsail get-instance-snapshots
```

输出：

```
{
  "instanceSnapshots": [
    {
      "name": "MEAN-1-1571421498",
      "arn": "arn:aws:lightsail:us-west-2:111122223333:InstanceSnapshot/a20e6ebe-b0ee-4ae4-a750-3EXAMPLEcb0c",
      "supportCode": "6EXAMPLE3362/ami-0EXAMPLEEe33cabfa1",
      "createdAt": 1571421527.755,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      },
      "resourceType": "InstanceSnapshot",
      "tags": [
        {
          "key": "no_delete"
        }
      ],
      "state": "available",
      "fromAttachedDisks": [],
      "fromInstanceName": "MEAN-1",
    }
  ]
}
```

```

        "fromInstanceArn": "arn:aws:lightsail:us-
west-2:111122223333:Instance/1761aa0a-6038-4f25-8b94-2EXAMPLE19fd",
        "fromBlueprintId": "wordpress_5_1_1_2",
        "fromBundleId": "micro_2_0",
        "isFromAutoSnapshot": false,
        "sizeInGb": 40
    },
    {
        "name": "MEAN-1-1571419854",
        "arn": "arn:aws:lightsail:us-west-2:111122223333:InstanceSnapshot/
ac54700c-48a8-40fd-b065-2EXAMPLEac8f",
        "supportCode": "6EXAMPLE3362/ami-0EXAMPLE67a73020d",
        "createdAt": 1571419891.927,
        "location": {
            "availabilityZone": "all",
            "regionName": "us-west-2"
        },
        "resourceType": "InstanceSnapshot",
        "tags": [],
        "state": "available",
        "fromAttachedDisks": [],
        "fromInstanceName": "MEAN-1",
        "fromInstanceArn": "arn:aws:lightsail:us-west-2:111122223333:Instance/
bd470fc5-a68b-44c5-8dbc-8EXAMPLEbada",
        "fromBlueprintId": "mean_4_0_9",
        "fromBundleId": "medium_2_0",
        "isFromAutoSnapshot": false,
        "sizeInGb": 80
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInstanceSnapshots](#)中的。

## get-instance-state

以下代码示例演示了如何使用 get-instance-state。

### AWS CLI

获取有关实例状态的信息

以下get-instance-state示例返回指定实例的状态。



```
aws lightsail get-instance-state \  
  --instance-name MEAN-1
```

输出：

```
{  
  "state": {  
    "code": 16,  
    "name": "running"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInstanceState](#)中的。

## get-instance

以下代码示例演示了如何使用 get-instance。

### AWS CLI

获取有关实例的信息

以下get-instance示例显示了有关该实例的详细信息MEAN-1。

```
aws lightsail get-instance \  
  --instance-name MEAN-1
```

输出：

```
{  
  "instance": {  
    "name": "MEAN-1",  
    "arn": "arn:aws:lightsail:us-west-2:111122223333:Instance/bd470fc5-  
a68b-44c5-8dbc-EXAMPLE4bada",  
    "supportCode": "6EXAMPLE3362/i-05EXAMPLE407c97d3",  
    "createdAt": 1570635023.124,  
    "location": {  
      "availabilityZone": "us-west-2a",  
      "regionName": "us-west-2"  
    },  
    "resourceType": "Instance",  
    "tags": [],
```

```
"blueprintId": "mean_4_0_9",
"blueprintName": "MEAN",
"bundleId": "medium_2_0",
"isStaticIp": false,
"privateIpAddress": "192.0.2.0",
"publicIpAddress": "192.0.2.0",
"hardware": {
  "cpuCount": 2,
  "disks": [
    {
      "createdAt": 1570635023.124,
      "sizeInGb": 80,
      "isSystemDisk": true,
      "iops": 240,
      "path": "/dev/sda1",
      "attachedTo": "MEAN-1",
      "attachmentState": "attached"
    }
  ],
  "ramSizeInGb": 4.0
},
"networking": {
  "monthlyTransfer": {
    "gbPerMonthAllocated": 4096
  },
  "ports": [
    {
      "fromPort": 80,
      "toPort": 80,
      "protocol": "tcp",
      "accessFrom": "Anywhere (0.0.0.0/0)",
      "accessType": "public",
      "commonName": "",
      "accessDirection": "inbound"
    },
    {
      "fromPort": 22,
      "toPort": 22,
      "protocol": "tcp",
      "accessFrom": "Anywhere (0.0.0.0/0)",
      "accessType": "public",
      "commonName": "",
      "accessDirection": "inbound"
    }
  ],
}
```

```

        {
            "fromPort": 443,
            "toPort": 443,
            "protocol": "tcp",
            "accessFrom": "Anywhere (0.0.0.0/0)",
            "accessType": "public",
            "commonName": "",
            "accessDirection": "inbound"
        }
    ]
},
"state": {
    "code": 16,
    "name": "running"
},
"username": "bitnami",
"sshKeyName": "MyKey"
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInstance](#)中的。

## get-instances

以下代码示例演示了如何使用 get-instances。

### AWS CLI

获取有关所有实例的信息

以下get-instances示例显示了有关已配置 AWS 区域中所有实例的详细信息。

```
aws lightsail get-instances
```

输出：

```

{
  "instances": [
    {
      "name": "Windows_Server_2016-1",
      "arn": "arn:aws:lightsail:us-
west-2:111122223333:Instance/0f44fbb9-8f55-4e47-a25e-EXAMPLE04763",
      "supportCode": "62EXAMPLE362/i-0bEXAMPLE71a686b9",

```

```
"createdAt": 1571332358.665,
"location": {
  "availabilityZone": "us-west-2a",
  "regionName": "us-west-2"
},
"resourceType": "Instance",
"tags": [],
"blueprintId": "windows_server_2016",
"blueprintName": "Windows Server 2016",
"bundleId": "small_win_2_0",
"isStaticIp": false,
"privateIpAddress": "192.0.2.0",
"publicIpAddress": "192.0.2.0",
"hardware": {
  "cpuCount": 1,
  "disks": [
    {
      "createdAt": 1571332358.665,
      "sizeInGb": 60,
      "isSystemDisk": true,
      "iops": 180,
      "path": "/dev/sda1",
      "attachedTo": "Windows_Server_2016-1",
      "attachmentState": "attached"
    },
    {
      "name": "my-disk-for-windows-server",
      "arn": "arn:aws:lightsail:us-
west-2:111122223333:Disk/4123a81c-484c-49ea-afea-5EXAMPLEda87",
      "supportCode": "6EXAMPLE3362/vol-0EXAMPLEb2b99ca3d",
      "createdAt": 1571355063.494,
      "location": {
        "availabilityZone": "us-west-2a",
        "regionName": "us-west-2"
      },
      "resourceType": "Disk",
      "tags": [],
      "sizeInGb": 128,
      "isSystemDisk": false,
      "iops": 384,
      "path": "/dev/xvdf",
      "state": "in-use",
      "attachedTo": "Windows_Server_2016-1",
      "isAttached": true,
```

```
        "attachmentState": "attached"
      }
    ],
    "ramSizeInGb": 2.0
  },
  "networking": {
    "monthlyTransfer": {
      "gbPerMonthAllocated": 3072
    },
    "ports": [
      {
        "fromPort": 80,
        "toPort": 80,
        "protocol": "tcp",
        "accessFrom": "Anywhere (0.0.0.0/0)",
        "accessType": "public",
        "commonName": "",
        "accessDirection": "inbound"
      },
      {
        "fromPort": 22,
        "toPort": 22,
        "protocol": "tcp",
        "accessFrom": "Anywhere (0.0.0.0/0)",
        "accessType": "public",
        "commonName": "",
        "accessDirection": "inbound"
      },
      {
        "fromPort": 3389,
        "toPort": 3389,
        "protocol": "tcp",
        "accessFrom": "Anywhere (0.0.0.0/0)",
        "accessType": "public",
        "commonName": "",
        "accessDirection": "inbound"
      }
    ]
  },
  "state": {
    "code": 16,
    "name": "running"
  },
  "username": "Administrator",
```

```
    "sshKeyName": "LightsailDefaultKeyPair"
  },
  {
    "name": "MEAN-1",
    "arn": "arn:aws:lightsail:us-west-2:111122223333:Instance/bd470fc5-
a68b-44c5-8dbc-8EXAMPLEbada",
    "supportCode": "6EXAMPLE3362/i-0EXAMPLEa407c97d3",
    "createdAt": 1570635023.124,
    "location": {
      "availabilityZone": "us-west-2a",
      "regionName": "us-west-2"
    },
    "resourceType": "Instance",
    "tags": [],
    "blueprintId": "mean_4_0_9",
    "blueprintName": "MEAN",
    "bundleId": "medium_2_0",
    "isStaticIp": false,
    "privateIpAddress": "192.0.2.0",
    "publicIpAddress": "192.0.2.0",
    "hardware": {
      "cpuCount": 2,
      "disks": [
        {
          "name": "Disk-1",
          "arn": "arn:aws:lightsail:us-west-2:111122223333:Disk/
c21cfb0a-07f2-44ae-9a23-bEXAMPLE8096",
          "supportCode": "6EXAMPLE3362/vol-0EXAMPLEf2f88b32f",
          "createdAt": 1566585439.587,
          "location": {
            "availabilityZone": "us-west-2a",
            "regionName": "us-west-2"
          },
          "resourceType": "Disk",
          "tags": [
            {
              "key": "test"
            }
          ],
          "sizeInGb": 8,
          "isSystemDisk": false,
          "iops": 100,
          "path": "/dev/xvdf",
          "state": "in-use",
```

```
        "attachedTo": "MEAN-1",
        "isAttached": true,
        "attachmentState": "attached"
    },
    {
        "createdAt": 1570635023.124,
        "sizeInGb": 80,
        "isSystemDisk": true,
        "iops": 240,
        "path": "/dev/sda1",
        "attachedTo": "MEAN-1",
        "attachmentState": "attached"
    }
],
"ramSizeInGb": 4.0
},
"networking": {
    "monthlyTransfer": {
        "gbPerMonthAllocated": 4096
    },
    "ports": [
        {
            "fromPort": 80,
            "toPort": 80,
            "protocol": "tcp",
            "accessFrom": "Anywhere (0.0.0.0/0)",
            "accessType": "public",
            "commonName": "",
            "accessDirection": "inbound"
        },
        {
            "fromPort": 22,
            "toPort": 22,
            "protocol": "tcp",
            "accessFrom": "Anywhere (0.0.0.0/0)",
            "accessType": "public",
            "commonName": "",
            "accessDirection": "inbound"
        },
        {
            "fromPort": 443,
            "toPort": 443,
            "protocol": "tcp",
            "accessFrom": "Anywhere (0.0.0.0/0)",
```

```

        "accessType": "public",
        "commonName": "",
        "accessDirection": "inbound"
      }
    ]
  },
  "state": {
    "code": 16,
    "name": "running"
  },
  "username": "bitnami",
  "sshKeyName": "MyTestKey"
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInstances](#)中的。

## get-key-pair

以下代码示例演示了如何使用 `get-key-pair`。

### AWS CLI

获取有关 key pair 的信息

以下`get-key-pair`示例显示有关指定 key pair 的详细信息。

```
aws lightsail get-key-pair \
  --key-pair-name MyKey1
```

输出：

```
{
  "keyPair": {
    "name": "MyKey1",
    "arn": "arn:aws:lightsail:us-
west-2:111122223333:KeyPair/19a4efdf-3054-43d6-91fd-eEXAMPLE21bf",
    "supportCode": "6EXAMPLE3362/MyKey1",
    "createdAt": 1571255026.975,
    "location": {
      "availabilityZone": "all",

```



```
    "regionName": "us-west-2"
  },
  "resourceType": "KeyPair",
  "tags": [],
  "fingerprint": "00:11:22:33:44:55:66:77:88:99:aa:bb:cc:dd:ee:ff:gg:hh:ii:jj"
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetKeyPair](#)中的。

## get-key-pairs

以下代码示例演示了如何使用 get-key-pairs。

### AWS CLI

获取有关所有密钥对的信息

以下get-key-pairs示例显示了有关已配置 AWS 区域中所有密钥对的详细信息。

```
aws lightsail get-key-pairs
```

输出：

```
{
  "keyPairs": [
    {
      "name": "MyKey1",
      "arn": "arn:aws:lightsail:us-west-2:111122223333:KeyPair/19a4efdf-3054-43d6-91fd-eEXAMPLE21bf",
      "supportCode": "6EXAMPLE3362/MyKey1",
      "createdAt": 1571255026.975,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      },
      "resourceType": "KeyPair",
      "tags": [],
      "fingerprint": "00:11:22:33:44:55:66:77:88:99:aa:bb:cc:dd:ee:ff:gg:hh:ii:jj"
    }
  ]
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetKeyPairs](#)中的。

## get-load-balancer-tls-certificates

以下代码示例演示了如何使用 `get-load-balancer-tls-certificates`。

### AWS CLI

获取有关负载均衡器的 TLS 证书的信息

以下`get-load-balancer-tls-certificates`示例显示了有关指定负载均衡器的 TLS 证书的详细信息。

```
aws lightsail get-load-balancer-tls-certificates \  
  --load-balancer-name LoadBalancer-1
```

输出：

```
{  
  "tlsCertificates": [  
    {  
      "name": "example-com",  
      "arn": "arn:aws:lightsail:us-west-2:111122223333:LoadBalancerTlsCertificate/d7bf4643-6a02-4cd4-b3c4-fEXAMPLE9b4d",  
      "supportCode": "6EXAMPLE3362/arn:aws:acm:us-west-2:333322221111:certificate/9af8e32c-a54e-4a67-8c63-cEXAMPLEb314",  
      "createdAt": 1571678025.3,  
      "location": {  
        "availabilityZone": "all",  
        "regionName": "us-west-2"  
      },  
      "resourceType": "LoadBalancerTlsCertificate",  
      "loadBalancerName": "LoadBalancer-1",  
      "isAttached": false,  
      "status": "ISSUED",  
      "domainName": "example.com",  
      "domainValidationRecords": [  
        {  
          "name": "_dEXAMPLE4ede046a0319eb44a4eb3cbc.example.com.",
```

```

        "type": "CNAME",
        "value": "_bEXAMPLE0899fb7b6bf79d9741d1a383.hkvuijqjoua.acm-
validations.aws.",
        "validationStatus": "SUCCESS",
        "domainName": "example.com"
    }
],
"issuedAt": 1571678070.0,
"issuer": "Amazon",
"keyAlgorithm": "RSA-2048",
"notAfter": 1605960000.0,
"notBefore": 1571616000.0,
"serial": "00:11:22:33:44:55:66:77:88:99:aa:bb:cc:dd:ee:ff",
"signatureAlgorithm": "SHA256WITHRSA",
"subject": "CN=example.com",
"subjectAlternativeNames": [
    "example.com"
]
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLoadBalancerTlsCertificates](#)中的。

## get-load-balancer

以下代码示例演示了如何使用 get-load-balancer。

### AWS CLI

获取有关负载均衡器的信息

以下get-load-balancer示例显示了有关指定负载均衡器的详细信息。

```
aws lightsail get-load-balancer \
  --load-balancer-name LoadBalancer-1
```

输出：

```
{
  "loadBalancer": {
    "name": "LoadBalancer-1",
```

```
    "arn": "arn:aws:lightsail:us-  
west-2:111122223333:LoadBalancer/40486b2b-1ad0-4152-83e4-cEXAMPLE6f4b",  
    "supportCode": "6EXAMPLE3362/arn:aws:elasticloadbalancing:us-  
west-2:333322221111:loadbalancer/app/  
bEXAMPLE128cb59d86f946a9395dd304/1EXAMPLE8dd9d77e",  
    "createdAt": 1571677906.723,  
    "location": {  
      "availabilityZone": "all",  
      "regionName": "us-west-2"  
    },  
    "resourceType": "LoadBalancer",  
    "tags": [],  
    "dnsName": "bEXAMPLE128cb59d86f946a9395dd304-1486911371.us-  
west-2.elb.amazonaws.com",  
    "state": "active",  
    "protocol": "HTTP",  
    "publicPorts": [  
      80  
    ],  
    "healthCheckPath": "/",  
    "instancePort": 80,  
    "instanceHealthSummary": [  
      {  
        "instanceName": "MEAN-3",  
        "instanceHealth": "healthy"  
      },  
      {  
        "instanceName": "MEAN-1",  
        "instanceHealth": "healthy"  
      },  
      {  
        "instanceName": "MEAN-2",  
        "instanceHealth": "healthy"  
      }  
    ],  
    "tlsCertificateSummaries": [  
      {  
        "name": "example-com",  
        "isAttached": false  
      }  
    ],  
    "configurationOptions": {  
      "SessionStickinessEnabled": "false",  
      "SessionStickiness_LB_CookieDurationSeconds": "86400"
```

```
    }  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLoadBalancer](#)中的。

## get-load-balancers

以下代码示例演示了如何使用 `get-load-balancers`。

### AWS CLI

获取有关所有负载均衡器的信息

以下`get-load-balancers`示例显示了有关已配置 AWS 区域中所有负载均衡器的详细信息。

```
aws lightsail get-load-balancers
```

输出：

```
{  
  "loadBalancers": [  
    {  
      "name": "LoadBalancer-1",  
      "arn": "arn:aws:lightsail:us-west-2:111122223333:LoadBalancer/40486b2b-1ad0-4152-83e4-cEXAMPLE6f4b",  
      "supportCode": "6EXAMPLE3362/arn:aws:elasticloadbalancing:us-west-2:333322221111:loadbalancer/app/bEXAMPLE128cb59d86f946a9395dd304/1EXAMPLE8dd9d77e",  
      "createdAt": 1571677906.723,  
      "location": {  
        "availabilityZone": "all",  
        "regionName": "us-west-2"  
      },  
      "resourceType": "LoadBalancer",  
      "tags": [],  
      "dnsName": "bEXAMPLE128cb59d86f946a9395dd304-1486911371.us-west-2.elb.amazonaws.com",  
      "state": "active",  
      "protocol": "HTTP",  
      "publicPorts": [  
        80  
      ],  
    },  
  ],  
}
```

```
    "healthCheckPath": "/",
    "instancePort": 80,
    "instanceHealthSummary": [
      {
        "instanceName": "MEAN-3",
        "instanceHealth": "healthy"
      },
      {
        "instanceName": "MEAN-1",
        "instanceHealth": "healthy"
      },
      {
        "instanceName": "MEAN-2",
        "instanceHealth": "healthy"
      }
    ],
    "tlsCertificateSummaries": [
      {
        "name": "example-com",
        "isAttached": false
      }
    ],
    "configurationOptions": {
      "SessionStickinessEnabled": "false",
      "SessionStickiness_LB_CookieDurationSeconds": "86400"
    }
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLoadBalancers](#)中的。

## get-operation

以下代码示例演示了如何使用 get-operation。

### AWS CLI

获取有关单个操作的信息

以下get-operation示例显示有关指定操作的详细信息。

```
aws lightsail get-operation \
```

```
--operation-id e5700e8a-daf2-4b49-bc01-3EXAMPLE910a
```

输出：

```
{
  "operation": {
    "id": "e5700e8a-daf2-4b49-bc01-3EXAMPLE910a",
    "resourceName": "Instance-1",
    "resourceType": "Instance",
    "createdAt": 1571679872.404,
    "location": {
      "availabilityZone": "us-west-2a",
      "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationType": "CreateInstance",
    "status": "Succeeded",
    "statusChangedAt": 1571679890.304
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetOperation](#)中的。

## get-operations-for-resource

以下代码示例演示了如何使用 `get-operations-for-resource`。

AWS CLI

获取资源的所有操作

以下`get-operations-for-resource`示例显示有关指定资源的所有操作的详细信息。

```
aws lightsail get-operations-for-resource \
  --resource-name LoadBalancer-1
```

输出：

```
{
  "operations": [
    {
      "id": "e2973046-43f8-4252-a4b4-9EXAMPLE69ce",
```

```
    "resourceName": "LoadBalancer-1",
    "resourceType": "LoadBalancer",
    "createdAt": 1571678786.071,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationDetails": "MEAN-1",
    "operationType": "DetachInstancesFromLoadBalancer",
    "status": "Succeeded",
    "statusChangedAt": 1571679087.57
  },
  {
    "id": "2d742a18-0e7f-48c8-9705-3EXAMPLEf98a",
    "resourceName": "LoadBalancer-1",
    "resourceType": "LoadBalancer",
    "createdAt": 1571678782.784,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationDetails": "MEAN-1",
    "operationType": "AttachInstancesToLoadBalancer",
    "status": "Succeeded",
    "statusChangedAt": 1571678798.465
  },
  {
    "id": "6c700fcc-4246-40ab-952b-1EXAMPLEdac2",
    "resourceName": "LoadBalancer-1",
    "resourceType": "LoadBalancer",
    "createdAt": 1571678775.297,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationDetails": "MEAN-3",
    "operationType": "AttachInstancesToLoadBalancer",
    "status": "Succeeded",
    "statusChangedAt": 1571678842.806
  },
  ...
```



```
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetOperationsForResource](#)中的。

## get-operations

以下代码示例演示了如何使用 get-operations。

### AWS CLI

获取有关所有操作的信息

以下get-operations示例显示了有关已配置 AWS 区域中所有操作的详细信息。

```
aws lightsail get-operations
```

输出：

```
{
  "operations": [
    {
      "id": "e5700e8a-daf2-4b49-bc01-3EXAMPLE910a",
      "resourceName": "Instance-1",
      "resourceType": "Instance",
      "createdAt": 1571679872.404,
      "location": {
        "availabilityZone": "us-west-2a",
        "regionName": "us-west-2"
      },
      "isTerminal": true,
      "operationType": "CreateInstance",
      "status": "Succeeded",
      "statusChangedAt": 1571679890.304
    },
    {
      "id": "701a3339-930e-4914-a9f9-7EXAMPLE68d7",
      "resourceName": "WordPress-1",
      "resourceType": "Instance",
      "createdAt": 1571678786.072,
      "location": {
        "availabilityZone": "all",
```

```
        "regionName": "us-west-2"
      },
      "isTerminal": true,
      "operationDetails": "LoadBalancer-1",
      "operationType": "DetachInstancesFromLoadBalancer",
      "status": "Succeeded",
      "statusChangedAt": 1571679086.399
    },
    {
      "id": "e2973046-43f8-4252-a4b4-9EXAMPLE69ce",
      "resourceName": "LoadBalancer-1",
      "resourceType": "LoadBalancer",
      "createdAt": 1571678786.071,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      },
      "isTerminal": true,
      "operationDetails": "WordPress-1",
      "operationType": "DetachInstancesFromLoadBalancer",
      "status": "Succeeded",
      "statusChangedAt": 1571679087.57
    },
    ...
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetOperations](#)中的。

## get-regions

以下代码示例演示了如何使用 get-regions。

### AWS CLI

要获取亚马逊 Lightsail 的所有 AWS 区域

以下get-regions示例显示了有关亚马逊 Lightsail 所有 AWS 区域的详细信息。

```
aws lightsail get-regions
```

输出：

```
{
  "regions": [
    {
      "continentCode": "NA",
      "description": "This region is recommended to serve users in the eastern
United States",
      "displayName": "Virginia",
      "name": "us-east-1",
      "availabilityZones": [],
      "relationalDatabaseAvailabilityZones": []
    },
    {
      "continentCode": "NA",
      "description": "This region is recommended to serve users in the eastern
United States",
      "displayName": "Ohio",
      "name": "us-east-2",
      "availabilityZones": [],
      "relationalDatabaseAvailabilityZones": []
    },
    {
      "continentCode": "NA",
      "description": "This region is recommended to serve users in the
northwestern United States, Alaska, and western Canada",
      "displayName": "Oregon",
      "name": "us-west-2",
      "availabilityZones": [],
      "relationalDatabaseAvailabilityZones": []
    },
    ...
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRegions](#)中的。

## get-relational-database-blueprints

以下代码示例演示了如何使用 get-relational-database-blueprints。

### AWS CLI

#### 获取新关系数据库的蓝图

以下`get-relational-database-blueprints`示例显示了有关所有可用关系数据库蓝图的详细信息，这些蓝图可用于在 Amazon Lightsail 中创建新的关系数据库。

```
aws lightsail get-relational-database-blueprints
```

输出：

```
{
  "blueprints": [
    {
      "blueprintId": "mysql_5_6",
      "engine": "mysql",
      "engineVersion": "5.6.44",
      "engineDescription": "MySQL Community Edition",
      "engineVersionDescription": "MySQL 5.6.44",
      "isEngineDefault": false
    },
    {
      "blueprintId": "mysql_5_7",
      "engine": "mysql",
      "engineVersion": "5.7.26",
      "engineDescription": "MySQL Community Edition",
      "engineVersionDescription": "MySQL 5.7.26",
      "isEngineDefault": true
    },
    {
      "blueprintId": "mysql_8_0",
      "engine": "mysql",
      "engineVersion": "8.0.16",
      "engineDescription": "MySQL Community Edition",
      "engineVersionDescription": "MySQL 8.0.16",
      "isEngineDefault": false
    },
    {
      "blueprintId": "postgres_9_6",
      "engine": "postgres",
      "engineVersion": "9.6.15",
      "engineDescription": "PostgreSQL",
      "engineVersionDescription": "PostgreSQL 9.6.15-R1",
      "isEngineDefault": false
    },
    {
      "blueprintId": "postgres_10",
```

```
    "engine": "postgres",
    "engineVersion": "10.10",
    "engineDescription": "PostgreSQL",
    "engineVersionDescription": "PostgreSQL 10.10-R1",
    "isEngineDefault": false
  },
  {
    "blueprintId": "postgres_11",
    "engine": "postgres",
    "engineVersion": "11.5",
    "engineDescription": "PostgreSQL",
    "engineVersionDescription": "PostgreSQL 11.5-R1",
    "isEngineDefault": true
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRelationalDatabaseBlueprints](#)中的。

## get-relational-database-bundles

以下代码示例演示了如何使用 `get-relational-database-bundles`。

### AWS CLI

获取新关系数据库的捆绑包

以下`get-relational-database-bundles`示例显示了有关所有可用关系数据库捆绑包的详细信息，这些捆绑包可用于在 Amazon Lightsail 中创建新的关系数据库。请注意，响应中不包括非活动捆绑包，因为命令中未指定该`--include-inactive`标志。您不能使用非活动分发包来创建新的关系数据库。

```
aws lightsail get-relational-database-bundles
```

输出：

```
{
  "bundles": [
    {
      "bundleId": "micro_2_0",
      "name": "Micro",
      "price": 15.0,
```

```
    "ramSizeInGb": 1.0,  
    "diskSizeInGb": 40,  
    "transferPerMonthInGb": 100,  
    "cpuCount": 2,  
    "isEncrypted": true,  
    "isActive": true  
  },  
  {  
    "bundleId": "micro_ha_2_0",  
    "name": "Micro with High Availability",  
    "price": 30.0,  
    "ramSizeInGb": 1.0,  
    "diskSizeInGb": 40,  
    "transferPerMonthInGb": 100,  
    "cpuCount": 2,  
    "isEncrypted": true,  
    "isActive": true  
  },  
  {  
    "bundleId": "small_2_0",  
    "name": "Small",  
    "price": 30.0,  
    "ramSizeInGb": 2.0,  
    "diskSizeInGb": 80,  
    "transferPerMonthInGb": 100,  
    "cpuCount": 2,  
    "isEncrypted": true,  
    "isActive": true  
  },  
  {  
    "bundleId": "small_ha_2_0",  
    "name": "Small with High Availability",  
    "price": 60.0,  
    "ramSizeInGb": 2.0,  
    "diskSizeInGb": 80,  
    "transferPerMonthInGb": 100,  
    "cpuCount": 2,  
    "isEncrypted": true,  
    "isActive": true  
  },  
  {  
    "bundleId": "medium_2_0",  
    "name": "Medium",  
    "price": 60.0,
```

```
    "ramSizeInGb": 4.0,  
    "diskSizeInGb": 120,  
    "transferPerMonthInGb": 100,  
    "cpuCount": 2,  
    "isEncrypted": true,  
    "isActive": true  
  },  
  {  
    "bundleId": "medium_ha_2_0",  
    "name": "Medium with High Availability",  
    "price": 120.0,  
    "ramSizeInGb": 4.0,  
    "diskSizeInGb": 120,  
    "transferPerMonthInGb": 100,  
    "cpuCount": 2,  
    "isEncrypted": true,  
    "isActive": true  
  },  
  {  
    "bundleId": "large_2_0",  
    "name": "Large",  
    "price": 115.0,  
    "ramSizeInGb": 8.0,  
    "diskSizeInGb": 240,  
    "transferPerMonthInGb": 200,  
    "cpuCount": 2,  
    "isEncrypted": true,  
    "isActive": true  
  },  
  {  
    "bundleId": "large_ha_2_0",  
    "name": "Large with High Availability",  
    "price": 230.0,  
    "ramSizeInGb": 8.0,  
    "diskSizeInGb": 240,  
    "transferPerMonthInGb": 200,  
    "cpuCount": 2,  
    "isEncrypted": true,  
    "isActive": true  
  }  
]  
}
```

有关更多信息，请参阅《[亚马逊 Light sail 开发者指南](#)》中的在 [Amazon Light sail 中创建数据库](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRelationalDatabaseBundles](#)中的。

## get-relational-database-events

以下代码示例演示了如何使用 get-relational-database-events。

### AWS CLI

获取关系数据库的事件

以下get-relational-database-events示例显示有关指定关系数据库在过去 17 小时 ( 1020 分钟 ) 内发生的事件的详细信息。

```
aws lightsail get-relational-database-events \  
  --relational-database-name Database-1 \  
  --duration-in-minutes 1020
```

输出：

```
{  
  "relationalDatabaseEvents": [  
    {  
      "resource": "Database-1",  
      "createdAt": 1571654146.553,  
      "message": "Backing up Relational Database",  
      "eventCategories": [  
        "backup"  
      ]  
    },  
    {  
      "resource": "Database-1",  
      "createdAt": 1571654249.98,  
      "message": "Finished Relational Database backup",  
      "eventCategories": [  
        "backup"  
      ]  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRelationalDatabaseEvents](#)中的。



## get-relational-database-log-events

以下代码示例演示了如何使用 `get-relational-database-log-events`。

### AWS CLI

获取关系数据库的日志事件

以下 `get-relational-database-log-events` 示例显示了有关介于 1570733176 和之间的数据库 1571597176 的指定日志的详细信息 Database1。返回的信息配置为从开始 head。

我们建议您使用 `unix` 时间转换器来识别开始和结束时间。

```
aws lightsail get-relational-database-log-events \
  --relational-database-name Database1 \
  --log-stream-name error \
  --start-from-head \
  --start-time 1570733176 \
  --end-time 1571597176
```

输出：

```
{
  "resourceLogEvents": [
    {
      "createdAt": 1570820267.0,
      "message": "2019-10-11 18:57:47 20969 [Warning] IP address '192.0.2.0'
could not be resolved: Name or service not known"
    },
    {
      "createdAt": 1570860974.0,
      "message": "2019-10-12 06:16:14 20969 [Warning] IP address '8192.0.2.0'
could not be resolved: Temporary failure in name resolution"
    },
    {
      "createdAt": 1570860977.0,
      "message": "2019-10-12 06:16:17 20969 [Warning] IP address '192.0.2.0'
could not be resolved: Temporary failure in name resolution"
    },
    {
      "createdAt": 1570860979.0,
      "message": "2019-10-12 06:16:19 20969 [Warning] IP address '192.0.2.0'
could not be resolved: Temporary failure in name resolution"
    },
  ],
}
```

```

    {
      "createdAt": 1570860981.0,
      "message": "2019-10-12 06:16:21 20969 [Warning] IP address '192.0.2.0'
could not be resolved: Temporary failure in name resolution"
    },
    {
      "createdAt": 1570860982.0,
      "message": "2019-10-12 06:16:22 20969 [Warning] IP address '192.0.2.0'
could not be resolved: Temporary failure in name resolution"
    },
    {
      "createdAt": 1570860984.0,
      "message": "2019-10-12 06:16:24 20969 [Warning] IP address '192.0.2.0'
could not be resolved: Temporary failure in name resolution"
    },
    {
      "createdAt": 1570860986.0,
      "message": "2019-10-12 06:16:26 20969 [Warning] IP address '192.0.2.0'
could not be resolved: Temporary failure in name resolution"
    },
    ...
  ],
  "nextBackwardToken":
"eEXAMPLEZXJUZXh0IjoiZnRwb3F3cUpRS1Q5NndMYThxe1RUZlFhR3JGc2dKWEEvM2kvajZMZzVWVWpqRDN0YjFXTj",
  "nextForwardToken":
"eEXAMPLEZXJUZXh0IjoiT09Lb0Z6ZFRJbHhaNEQ5N2tPbkkwRmwwNUxPZjFTbFFwUk1Qbz1SaWgvMwVXbEk4aG56VH"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRelationalDatabaseLogEvents](#)中的。

## get-relational-database-log-streams

以下代码示例演示了如何使用 `get-relational-database-log-streams`。

### AWS CLI

获取关系数据库的日志流

以下`get-relational-database-log-streams`示例返回指定关系数据库的所有可用日志流。

```
aws lightsail get-relational-database-log-streams \
--relational-database-name Database1
```

输出：

```
{
  "logStreams": [
    "audit",
    "error",
    "general",
    "slowquery"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRelationalDatabaseLogStreams](#)中的。

### **get-relational-database-master-user-password**

以下代码示例演示了如何使用 `get-relational-database-master-user-password`。

AWS CLI

获取关系数据库的主用户密码

以下`get-relational-database-master-user-password`示例返回有关指定关系数据库的主用户密码的信息。

```
aws lightsail get-relational-database-master-user-password \
  --relational-database-name Database-1
```

输出：

```
{
  "masterUserPassword": "VEXAMPLEec.9qvX,_t<)Wkf)kwboM,>2",
  "createdAt": 1571259453.959
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRelationalDatabaseMasterUserPassword](#)中的。

### **get-relational-database-metric-data**

以下代码示例演示了如何使用 `get-relational-database-metric-data`。

## AWS CLI

### 获取关系数据库的指标数据

以下`get-relational-database-metric-data`示例返回关系数据库之间的 24 小时 ( 86400秒 ) 内指标`DatabaseConnections`的计数总1570733176和1571597176Database1。

我们建议您使用 `unix` 时间转换器来识别开始和结束时间。

```
aws lightsail get-relational-database-metric-data \  
  --relational-database-name Database1 \  
  --metric-name DatabaseConnections \  
  --period 86400 \  
  --start-time 1570733176 \  
  --end-time 1571597176 \  
  --unit Count \  
  --statistics Sum
```

输出：

```
{  
  "metricName": "DatabaseConnections",  
  "metricData": [  
    {  
      "sum": 1.0,  
      "timestamp": 1571510760.0,  
      "unit": "Count"  
    },  
    {  
      "sum": 1.0,  
      "timestamp": 1570733160.0,  
      "unit": "Count"  
    },  
    {  
      "sum": 1.0,  
      "timestamp": 1570992360.0,  
      "unit": "Count"  
    },  
    {  
      "sum": 0.0,  
      "timestamp": 1571251560.0,  
      "unit": "Count"  
    }  
  ]  
}
```

```
    },
    {
      "sum": 721.0,
      "timestamp": 1570819560.0,
      "unit": "Count"
    },
    {
      "sum": 1.0,
      "timestamp": 1571078760.0,
      "unit": "Count"
    },
    {
      "sum": 2.0,
      "timestamp": 1571337960.0,
      "unit": "Count"
    },
    {
      "sum": 684.0,
      "timestamp": 1570905960.0,
      "unit": "Count"
    },
    {
      "sum": 0.0,
      "timestamp": 1571165160.0,
      "unit": "Count"
    },
    {
      "sum": 1.0,
      "timestamp": 1571424360.0,
      "unit": "Count"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRelationalDatabaseMetricData](#)中的。

## get-relational-database-parameters

以下代码示例演示了如何使用 get-relational-database-parameters。

### AWS CLI

#### 获取关系数据库的参数

以下`get-relational-database-parameters`示例返回有关指定关系数据库的所有可用参数的信息。

```
aws lightsail get-relational-database-parameters \  
  --relational-database-name Database-1
```

输出：

```
{  
  "parameters": [  
    {  
      "allowedValues": "0,1",  
      "applyMethod": "pending-reboot",  
      "applyType": "dynamic",  
      "dataType": "boolean",  
      "description": "Automatically set all granted roles as active after the  
user has authenticated successfully.",  
      "isModifiable": true,  
      "parameterName": "activate_all_roles_on_login",  
      "parameterValue": "0"  
    },  
    {  
      "allowedValues": "0,1",  
      "applyMethod": "pending-reboot",  
      "applyType": "static",  
      "dataType": "boolean",  
      "description": "Controls whether user-defined functions that have only  
an xxx symbol for the main function can be loaded",  
      "isModifiable": false,  
      "parameterName": "allow-suspicious-udfs"  
    },  
    {  
      "allowedValues": "0,1",  
      "applyMethod": "pending-reboot",  
      "applyType": "dynamic",  
      "dataType": "boolean",  
      "description": "Sets the autocommit mode",  
      "isModifiable": true,  
      "parameterName": "autocommit"  
    },  
    {  
      "allowedValues": "0,1",  
      "applyMethod": "pending-reboot",
```

```

        "applyType": "static",
        "dataType": "boolean",
        "description": "Controls whether the server autogenerates SSL key and
certificate files in the data directory, if they do not already exist.",
        "isModifiable": false,
        "parameterName": "auto_generate_certs"
    },
    ...
}
]
}

```

有关更多信息，请参阅 Lightsail 开发者[指南中的更新 Amazon Lightsail 中的数据库参数](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetRelationalDatabaseParameters](#)中的。

## get-relational-database-snapshot

以下代码示例演示了如何使用 get-relational-database-snapshot。

### AWS CLI

获取有关关系数据库快照的信息

以下 get-relational-database-snapshot 示例显示有关指定关系数据库快照的详细信息。

```
aws lightsail get-relational-database-snapshot \
  --relational-database-snapshot-name Database-1-1571350042
```

输出：

```
{
  "relationalDatabaseSnapshot": {
    "name": "Database-1-1571350042",
    "arn": "arn:aws:lightsail:us-
west-2:111122223333:RelationalDatabaseSnapshot/0389bbad-4b85-4c3d-9EXAMPLEaee3643d2",
    "supportCode": "6EXAMPLE3362/1s-8EXAMPLE2ba7ad041451946fafc2ad19cfbd9eb2",
    "createdAt": 1571350046.238,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "resourceType": "RelationalDatabaseSnapshot",

```

```
    "tags": [],
    "engine": "mysql",
    "engineVersion": "8.0.16",
    "sizeInGb": 40,
    "state": "available",
    "fromRelationalDatabaseName": "Database-1",
    "fromRelationalDatabaseArn": "arn:aws:lightsail:us-
west-2:111122223333:RelationalDatabase/7ea932b1-b85a-4bd5-9b3e-bEXAMPLE8cc4",
    "fromRelationalDatabaseBundleId": "micro_1_0",
    "fromRelationalDatabaseBlueprintId": "mysql_8_0"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRelationalDatabaseSnapshot](#)中的。

## get-relational-database-snapshots

以下代码示例演示了如何使用 get-relational-database-snapshots。

### AWS CLI

获取有关所有关系数据库快照的信息

以下get-relational-database-snapshots示例显示了有关已配置 AWS 区域中所有关系数据库快照的详细信息。

```
aws lightsail get-relational-database-snapshots
```

输出：

```
{
  "relationalDatabaseSnapshots": [
    {
      "name": "Database-1-1571350042",
      "arn": "arn:aws:lightsail:us-
west-2:111122223333:RelationalDatabaseSnapshot/0389bbad-4b85-4c3d-9861-6EXAMPLE43d2",
      "supportCode": "6EXAMPLE3362/
1s-8EXAMPLE2ba7ad041451946fafc2ad19cfbd9eb2",
      "createdAt": 1571350046.238,
      "location": {
        "availabilityZone": "all",
        "regionName": "us-west-2"
      }
    }
  ]
}
```



```

    },
    "resourceType": "RelationalDatabaseSnapshot",
    "tags": [],
    "engine": "mysql",
    "engineVersion": "8.0.16",
    "sizeInGb": 40,
    "state": "available",
    "fromRelationalDatabaseName": "Database-1",
    "fromRelationalDatabaseArn": "arn:aws:lightsail:us-
west-2:111122223333:RelationalDatabase/7ea932b1-b85a-4bd5-9b3e-bEXAMPLE8cc4",
    "fromRelationalDatabaseBundleId": "micro_1_0",
    "fromRelationalDatabaseBlueprintId": "mysql_8_0"
  },
  {
    "name": "Database1-Console",
    "arn": "arn:aws:lightsail:us-
west-2:111122223333:RelationalDatabaseSnapshot/8b94136e-06ec-4b1a-
a3fb-5EXAMPLEe1e9",
    "supportCode": "6EXAMPLE3362/
ls-9EXAMPLE14b000d34c8d1c432734e137612d5b5c",
    "createdAt": 1571249981.025,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "resourceType": "RelationalDatabaseSnapshot",
    "tags": [
      {
        "key": "test"
      }
    ],
    "engine": "mysql",
    "engineVersion": "5.6.44",
    "sizeInGb": 40,
    "state": "available",
    "fromRelationalDatabaseName": "Database1",
    "fromRelationalDatabaseArn": "arn:aws:lightsail:us-
west-2:111122223333:RelationalDatabase/a6161cb7-4535-4f16-9dcf-8EXAMPLE3d4e",
    "fromRelationalDatabaseBundleId": "micro_1_0",
    "fromRelationalDatabaseBlueprintId": "mysql_5_6"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRelationalDatabaseSnapshots](#)中的。

## get-relational-database

以下代码示例演示了如何使用 get-relational-database。

### AWS CLI

获取有关关系数据库的信息

以下get-relational-database示例显示有关指定关系数据库的详细信息。

```
aws lightsail get-relational-database \  
  --relational-database-name Database-1
```

输出：

```
{  
  "relationalDatabase": {  
    "name": "Database-1",  
    "arn": "arn:aws:lightsail:us-  
west-2:111122223333:RelationalDatabase/7ea932b1-b85a-4bd5-9b3e-bEXAMPLE8cc4",  
    "supportCode": "6EXAMPLE3362/1s-9EXAMPLE8ad863723b62cc8901a8aa6e794ae0d2",  
    "createdAt": 1571259453.795,  
    "location": {  
      "availabilityZone": "us-west-2a",  
      "regionName": "us-west-2"  
    },  
    "resourceType": "RelationalDatabase",  
    "tags": [],  
    "relationalDatabaseBlueprintId": "mysql_8_0",  
    "relationalDatabaseBundleId": "micro_1_0",  
    "masterDatabaseName": "dbmaster",  
    "hardware": {  
      "cpuCount": 1,  
      "diskSizeInGb": 40,  
      "ramSizeInGb": 1.0  
    },  
    "state": "available",  
    "backupRetentionEnabled": false,  
    "pendingModifiedValues": {},  
    "engine": "mysql",  
    "engineVersion": "8.0.16",
```

```
"masterUsername": "dbmasteruser",
"parameterApplyStatus": "in-sync",
"preferredBackupWindow": "10:01-10:31",
"preferredMaintenanceWindow": "sat:11:14-sat:11:44",
"publiclyAccessible": true,
"masterEndpoint": {
  "port": 3306,
  "address": "ls-9EXAMPLE8ad863723b62ccEXAMPLEa6e794ae0d2.czowadgeezqi.us-
west-2.rds.amazonaws.com"
},
"pendingMaintenanceActions": []
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRelationalDatabase](#)中的。

## get-relational-databases

以下代码示例演示了如何使用 get-relational-databases。

### AWS CLI

获取有关所有关系数据库的信息

以下get-relational-databases示例显示了有关已配置 AWS 区域中所有关系数据库的详细信息。

```
aws lightsail get-relational-databases
```

输出：

```
{
  "relationalDatabases": [
    {
      "name": "MySQL",
      "arn": "arn:aws:lightsail:us-
west-2:111122223333:RelationalDatabase/8529020c-3ab9-4d51-92af-5EXAMPLE8979",
      "supportCode": "6EXAMPLE3362/
ls-3EXAMPLEa995d8c3b06b4501356e5f2f28e1aeba",
      "createdAt": 1554306019.155,
      "location": {
        "availabilityZone": "us-west-2a",
```

```

        "regionName": "us-west-2"
    },
    "resourceType": "RelationalDatabase",
    "tags": [],
    "relationalDatabaseBlueprintId": "mysql_8_0",
    "relationalDatabaseBundleId": "micro_1_0",
    "masterDatabaseName": "dbmaster",
    "hardware": {
        "cpuCount": 1,
        "diskSizeInGb": 40,
        "ramSizeInGb": 1.0
    },
    "state": "available",
    "backupRetentionEnabled": true,
    "pendingModifiedValues": {},
    "engine": "mysql",
    "engineVersion": "8.0.15",
    "latestRestorableTime": 1571686200.0,
    "masterUsername": "dbmasteruser",
    "parameterApplyStatus": "in-sync",
    "preferredBackupWindow": "07:51-08:21",
    "preferredMaintenanceWindow": "tue:12:18-tue:12:48",
    "publiclyAccessible": true,
    "masterEndpoint": {
        "port": 3306,
        "address":
"ls-3EXAMPLEa995d8c3b06b4501356e5f2fEXAMPLEa.czowadgeezqi.us-
west-2.rds.amazonaws.com"
    },
    "pendingMaintenanceActions": []
},
{
    "name": "Postgres",
    "arn": "arn:aws:lightsail:us-west-2:111122223333:RelationalDatabase/
e9780b6b-d0ab-4af2-85f1-1EXAMPLEac68",
    "supportCode": "6EXAMPLE3362/
ls-3EXAMPLEb4ffffb5cec056220c734713e14bd5fcd",
    "createdAt": 1554306000.814,
    "location": {
        "availabilityZone": "us-west-2a",
        "regionName": "us-west-2"
    },
    "resourceType": "RelationalDatabase",
    "tags": [],

```

```

    "relationalDatabaseBlueprintId": "postgres_11",
    "relationalDatabaseBundleId": "micro_1_0",
    "masterDatabaseName": "dbmaster",
    "hardware": {
      "cpuCount": 1,
      "diskSizeInGb": 40,
      "ramSizeInGb": 1.0
    },
    "state": "available",
    "backupRetentionEnabled": true,
    "pendingModifiedValues": {},
    "engine": "postgres",
    "engineVersion": "11.1",
    "latestRestorableTime": 1571686339.0,
    "masterUsername": "dbmasteruser",
    "parameterApplyStatus": "in-sync",
    "preferredBackupWindow": "06:19-06:49",
    "preferredMaintenanceWindow": "sun:10:19-sun:10:49",
    "publiclyAccessible": false,
    "masterEndpoint": {
      "port": 5432,
      "address":
"1s-3EXAMPLEb4ffffb5cec056220c734713eEXAMPLEd.czowadgeezqi.us-
west-2.rds.amazonaws.com"
    },
    "pendingMaintenanceActions": []
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRelationalDatabases](#)中的。

## get-static-ip

以下代码示例演示了如何使用 `get-static-ip`。

### AWS CLI

获取有关静态 IP 的信息

以下`get-static-ip`示例显示有关指定静态 IP 的详细信息。

```
aws lightsail get-static-ip \
```

```
--static-ip-name StaticIp-1
```

输出：

```
{
  "staticIp": {
    "name": "StaticIp-1",
    "arn": "arn:aws:lightsail:us-west-2:111122223333:StaticIp/2257cd76-1f0e-4ac0-82e2-2EXAMPLE23ad",
    "supportCode": "6EXAMPLE3362/192.0.2.0",
    "createdAt": 1571071325.076,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "resourceType": "StaticIp",
    "ipAddress": "192.0.2.0",
    "isAttached": false
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetStaticIp](#)中的。

## get-static-ips

以下代码示例演示了如何使用 get-static-ips。

### AWS CLI

获取有关所有静态 IP 的信息

以下get-static-ips示例显示了有关已配置 AWS 区域中所有静态 IP 的详细信息。

```
aws lightsail get-static-ips
```

输出：

```
{
  "staticIps": [
    {
      "name": "StaticIp-1",
```

```

    "arn": "arn:aws:lightsail:us-
west-2:111122223333:StaticIp/2257cd76-1f0e-4ac0-8EXAMPLE16f9423ad",
    "supportCode": "6EXAMPLE3362/192.0.2.0",
    "createdAt": 1571071325.076,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "resourceType": "StaticIp",
    "ipAddress": "192.0.2.0",
    "isAttached": false
  },
  {
    "name": "StaticIP-2",
    "arn": "arn:aws:lightsail:us-west-2:111122223333:StaticIp/c61edb40-
e5f0-4fd6-ae7c-8EXAMPLE19f8",
    "supportCode": "6EXAMPLE3362/192.0.2.2",
    "createdAt": 1568305385.681,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "resourceType": "StaticIp",
    "ipAddress": "192.0.2.2",
    "attachedTo": "WordPress-1",
    "isAttached": true
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetStaticIps](#)中的。

## is-vpc-peered

以下代码示例演示了如何使用 is-vpc-peered。

### AWS CLI

识别您的 Amazon Lightsail 虚拟私有云是否处于对等状态

以下is-vpc-peered示例返回指定区域的 Amazon Lightsail 虚拟私有云 (VPC) 的对等互连状态。

AWS

```
aws lightsail is-vpc-peered \  
  --region us-west-2
```

输出：

```
{  
  "isPeered": true  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[IsVpcPeered](#)中的。

## open-instance-public-ports

以下代码示例演示了如何使用 open-instance-public-ports。

### AWS CLI

为实例打开防火墙端口

以下open-instance-public-ports示例在指定实例上打开 TCP 端口 22。

```
aws lightsail open-instance-public-ports \  
  --instance-name MEAN-2 \  
  --port-info fromPort=22,protocol=TCP,toPort=22
```

输出：

```
{  
  "operation": {  
    "id": "719744f0-a022-46f2-9f11-6EXAMPLE4642",  
    "resourceName": "MEAN-2",  
    "resourceType": "Instance",  
    "createdAt": 1571072906.849,  
    "location": {  
      "availabilityZone": "us-west-2a",  
      "regionName": "us-west-2"  
    },  
    "isTerminal": true,  
    "operationDetails": "22/tcp",  
    "operationType": "OpenInstancePublicPorts",
```



```
    "status": "Succeeded",
    "statusChangedAt": 1571072906.849
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[OpenInstancePublicPorts](#)中的。

## peer-vpc

以下代码示例演示了如何使用 peer-vpc。

### AWS CLI

对等亚马逊 Lightsail 虚拟私有云

以下peer-vpc示例对指定 AWS 区域的 Amazon Lightsail 虚拟私有云 (VPC) 进行对等。

```
aws lightsail peer-vpc \
  --region us-west-2
```

输出：

```
{
  "operation": {
    "id": "787e846a-54ac-497f-bce2-9EXAMPLE5d91",
    "resourceName": "vpc-0EXAMPLEa5261efb3",
    "resourceType": "PeeredVpc",
    "createdAt": 1571694233.104,
    "location": {
      "availabilityZone": "all",
      "regionName": "us-west-2"
    },
    "isTerminal": true,
    "operationDetails": "vpc-e2b3eb9b",
    "operationType": "PeeredVpc",
    "status": "Succeeded",
    "statusChangedAt": 1571694233.104
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PeerVpc](#)中的。

## reboot-instance

以下代码示例演示了如何使用 `reboot-instance`。

### AWS CLI

#### 重启实例

以下 `reboot-instance` 示例重新启动指定的实例。

```
aws lightsail reboot-instance \  
  --instance-name MEAN-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "2b679f1c-8b71-4bb4-8e97-8EXAMPLEed93",  
      "resourceName": "MEAN-1",  
      "resourceType": "Instance",  
      "createdAt": 1571694445.49,  
      "location": {  
        "availabilityZone": "us-west-2a",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": true,  
      "operationDetails": "",  
      "operationType": "RebootInstance",  
      "status": "Succeeded",  
      "statusChangedAt": 1571694445.49  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RebootInstance](#) 中的。

## reboot-relational-database

以下代码示例演示了如何使用 `reboot-relational-database`。

## AWS CLI

### 重新启动关系数据库

以下 `reboot-relational-database` 示例重新启动指定的关系数据库。

```
aws lightsail reboot-relational-database \  
  --relational-database-name Database-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "e4c980c0-3137-496c-9c91-1EXAMPLEdec2",  
      "resourceName": "Database-1",  
      "resourceType": "RelationalDatabase",  
      "createdAt": 1571694532.91,  
      "location": {  
        "availabilityZone": "us-west-2a",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationDetails": "",  
      "operationType": "RebootRelationalDatabase",  
      "status": "Started",  
      "statusChangedAt": 1571694532.91  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RebootRelationalDatabase](#) 中的。

### **release-static-ip**

以下代码示例演示了如何使用 `release-static-ip`。

## AWS CLI

### 删除静态 IP

以下 `release-static-ip` 示例删除了指定的静态 IP。

```
aws lightsail release-static-ip \  
  --static-ip-name StaticIp-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "e374c002-dc6d-4c7f-919f-2EXAMPLE13ce",  
      "resourceName": "StaticIp-1",  
      "resourceType": "StaticIp",  
      "createdAt": 1571694962.003,  
      "location": {  
        "availabilityZone": "all",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": true,  
      "operationType": "ReleaseStaticIp",  
      "status": "Succeeded",  
      "statusChangedAt": 1571694962.003  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReleaseStaticIp](#)中的。

## start-instance

以下代码示例演示了如何使用 start-instance。

### AWS CLI

#### 启动实例

以下start-instance示例启动指定的实例。

```
aws lightsail start-instance \  
  --instance-name WordPress-1
```

输出：

```
{
```

```
"operations": [  
  {  
    "id": "f88d2a93-7cea-4165-afce-2d688cb18f23",  
    "resourceName": "WordPress-1",  
    "resourceType": "Instance",  
    "createdAt": 1571695583.463,  
    "location": {  
      "availabilityZone": "us-west-2a",  
      "regionName": "us-west-2"  
    },  
    "isTerminal": false,  
    "operationType": "StartInstance",  
    "status": "Started",  
    "statusChangedAt": 1571695583.463  
  }  
]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartInstance](#)中的。

## start-relational-database

以下代码示例演示了如何使用 start-relational-database。

### AWS CLI

#### 启动关系数据库

以下start-relational-database示例启动指定的关系数据库。

```
aws lightsail start-relational-database \  
  --relational-database-name Database-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "4d5294ec-a38a-4fda-9e37-aEXAMPLE0d24",  
      "resourceName": "Database-1",  
      "resourceType": "RelationalDatabase",  
      "createdAt": 1571695998.822,  
      "location": {  
        "availabilityZone": "us-west-2a",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationType": "StartRelationalDatabase",  
      "status": "Starting",  
      "statusChangedAt": 1571695998.822  
    }  
  ]  
}
```

```
        "location": {
            "availabilityZone": "us-west-2a",
            "regionName": "us-west-2"
        },
        "isTerminal": false,
        "operationType": "StartRelationalDatabase",
        "status": "Started",
        "statusChangedAt": 1571695998.822
    }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartRelationalDatabase](#)中的。

## stop-instance

以下代码示例演示了如何使用 stop-instance。

### AWS CLI

#### 停止实例

以下stop-instance示例停止指定的实例。

```
aws lightsail stop-instance \  
--instance-name WordPress-1
```

输出：

```
{  
  "operations": [  
    {  
      "id": "265357e2-2943-4d51-888a-1EXAMPLE7585",  
      "resourceName": "WordPress-1",  
      "resourceType": "Instance",  
      "createdAt": 1571695471.134,  
      "location": {  
        "availabilityZone": "us-west-2a",  
        "regionName": "us-west-2"  
      },  
      "isTerminal": false,  
      "operationType": "StopInstance",  
      "status": "Started",
```

```
        "statusChangedAt": 1571695471.134
      }
    ]
  }
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopInstance](#)中的。

## stop-relational-database

以下代码示例演示了如何使用 stop-relational-database。

### AWS CLI

#### 停止关系数据库

以下stop-relational-database示例停止指定的关系数据库。

```
aws lightsail stop-relational-database \
  --relational-database-name Database-1
```

输出：

```
{
  "operations": [
    {
      "id": "cc559c19-4adb-41e4-b75b-5EXAMPLE4e61",
      "resourceName": "Database-1",
      "resourceType": "RelationalDatabase",
      "createdAt": 1571695526.29,
      "location": {
        "availabilityZone": "us-west-2a",
        "regionName": "us-west-2"
      },
      "isTerminal": false,
      "operationType": "StopRelationalDatabase",
      "status": "Started",
      "statusChangedAt": 1571695526.29
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopRelationalDatabase](#)中的。

## unpeer-vpc

以下代码示例演示了如何使用 unpeer-vpc。

### AWS CLI

取消对等 Amazon Lightsail 虚拟私有云

以下unpeer-vpc示例取消了指定区域的 Amazon Lightsail 虚拟私有云 (VPC) 的 AWS 对等关系。

```
aws lightsail unpeer-vpc \  
  --region us-west-2
```

输出：

```
{  
  "operation": {  
    "id": "531aca64-7157-47ab-84c6-eEXAMPLEd898",  
    "resourceName": "vpc-0EXAMPLEa5261efb3",  
    "resourceType": "PeeredVpc",  
    "createdAt": 1571694109.945,  
    "location": {  
      "availabilityZone": "all",  
      "regionName": "us-west-2"  
    },  
    "isTerminal": true,  
    "operationDetails": "vpc-e2b3eb9b",  
    "operationType": "UnpeeredVpc",  
    "status": "Succeeded",  
    "statusChangedAt": 1571694109.945  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UnpeerVpc](#)中的。

## 使用 Macie 的示例 AWS CLI

以下代码示例向您展示了如何在 Macie 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。



场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 [GitHub](#)，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## describe-buckets

以下代码示例演示了如何使用 `describe-buckets`。

AWS CLI

查询 Amazon Macie 为您的账户监控和分析的一个或多个 S3 存储桶的数据

以下 `describe-buckets` 示例查询名称以 `MY-S3` 开头且位于当前 AWS 区域的所有 S3 存储桶的元数据。

```
aws macie2 describe-buckets \  
  --criteria '{"bucketName":{"prefix":"my-S3"}}'
```

输出：

```
{  
  "buckets": [  
    {  
      "accountId": "123456789012",  
      "allowsUnencryptedObjectUploads": "FALSE",  
      "bucketArn": "arn:aws:s3:::MY-S3-DOC-EXAMPLE-BUCKET1",  
      "bucketCreatedAt": "2020-05-18T19:54:00+00:00",  
      "bucketName": "MY-S3-DOC-EXAMPLE-BUCKET1",  
      "classifiableObjectCount": 13,  
      "classifiableSizeInBytes": 1592088,  
      "jobDetails": {  
        "isDefinedInJob": "TRUE",  
        "isMonitoredByJob": "TRUE",  
        "lastJobId": "08c81dc4a2f3377fae45c9ddaexample",  
        "lastJobRunTime": "2021-04-26T14:55:30.270000+00:00"  
      },  
    },  
  ],  
}
```

```
"lastAutomatedDiscoveryTime": "2022-12-10T19:11:25.364000+00:00",
"lastUpdated": "2022-12-13T07:33:06.337000+00:00",
"objectCount": 13,
"objectCountByEncryptionType": {
  "customerManaged": 0,
  "kmsManaged": 2,
  "s3Managed": 7,
  "unencrypted": 4,
  "unknown": 0
},
"publicAccess": {
  "effectivePermission": "NOT_PUBLIC",
  "permissionConfiguration": {
    "accountLevelPermissions": {
      "blockPublicAccess": {
        "blockPublicAcls": true,
        "blockPublicPolicy": true,
        "ignorePublicAcls": true,
        "restrictPublicBuckets": true
      }
    },
    "bucketLevelPermissions": {
      "accessControlList": {
        "allowsPublicReadAccess": false,
        "allowsPublicWriteAccess": false
      },
      "blockPublicAccess": {
        "blockPublicAcls": true,
        "blockPublicPolicy": true,
        "ignorePublicAcls": true,
        "restrictPublicBuckets": true
      },
      "bucketPolicy": {
        "allowsPublicReadAccess": false,
        "allowsPublicWriteAccess": false
      }
    }
  }
},
"region": "us-west-2",
"replicationDetails": {
  "replicated": false,
  "replicatedExternally": false,
  "replicationAccounts": []
}
```

```
    },
    "sensitivityScore": 78,
    "serverSideEncryption": {
      "kmsMasterKeyId": null,
      "type": "NONE"
    },
    "sharedAccess": "NOT_SHARED",
    "sizeInBytes": 4549746,
    "sizeInBytesCompressed": 0,
    "tags": [
      {
        "key": "Division",
        "value": "HR"
      },
      {
        "key": "Team",
        "value": "Recruiting"
      }
    ],
    "unclassifiableObjectCount": {
      "fileType": 0,
      "storageClass": 0,
      "total": 0
    },
    "unclassifiableObjectSizeInBytes": {
      "fileType": 0,
      "storageClass": 0,
      "total": 0
    },
    "versioning": true
  },
  {
    "accountId": "123456789012",
    "allowsUnencryptedObjectUploads": "TRUE",
    "bucketArn": "arn:aws:s3:::MY-S3-DOC-EXAMPLE-BUCKET2",
    "bucketCreatedAt": "2020-11-25T18:24:38+00:00",
    "bucketName": "MY-S3-DOC-EXAMPLE-BUCKET2",
    "classifiableObjectCount": 8,
    "classifiableSizeInBytes": 133810,
    "jobDetails": {
      "isDefinedInJob": "TRUE",
      "isMonitoredByJob": "FALSE",
      "lastJobId": "188d4f6044d621771ef7d65f2example",
      "lastJobRunTime": "2021-04-09T19:37:11.511000+00:00"
    }
  }
}
```

```
    },
    "lastAutomatedDiscoveryTime": "2022-12-12T19:11:25.364000+00:00",
    "lastUpdated": "2022-12-13T07:33:06.337000+00:00",
    "objectCount": 8,
    "objectCountByEncryptionType": {
      "customerManaged": 0,
      "kmsManaged": 0,
      "s3Managed": 8,
      "unencrypted": 0,
      "unknown": 0
    },
    "publicAccess": {
      "effectivePermission": "NOT_PUBLIC",
      "permissionConfiguration": {
        "accountLevelPermissions": {
          "blockPublicAccess": {
            "blockPublicAcls": true,
            "blockPublicPolicy": true,
            "ignorePublicAcls": true,
            "restrictPublicBuckets": true
          }
        },
        "bucketLevelPermissions": {
          "accessControlList": {
            "allowsPublicReadAccess": false,
            "allowsPublicWriteAccess": false
          },
          "blockPublicAccess": {
            "blockPublicAcls": true,
            "blockPublicPolicy": true,
            "ignorePublicAcls": true,
            "restrictPublicBuckets": true
          },
          "bucketPolicy": {
            "allowsPublicReadAccess": false,
            "allowsPublicWriteAccess": false
          }
        }
      }
    },
    "region": "us-west-2",
    "replicationDetails": {
      "replicated": false,
      "replicatedExternally": false,
```

```
    "replicationAccounts": [],
  },
  "sensitivityScore": 95,
  "serverSideEncryption": {
    "kmsMasterKeyId": null,
    "type": "AES256"
  },
  "sharedAccess": "EXTERNAL",
  "sizeInBytes": 175978,
  "sizeInBytesCompressed": 0,
  "tags": [
    {
      "key": "Division",
      "value": "HR"
    },
    {
      "key": "Team",
      "value": "Recruiting"
    }
  ],
  "unclassifiableObjectCount": {
    "fileType": 3,
    "storageClass": 0,
    "total": 3
  },
  "unclassifiableObjectSizeInBytes": {
    "fileType": 2999826,
    "storageClass": 0,
    "total": 2999826
  },
  "versioning": true
}
]
```

有关更多信息，请参阅 [Amazon Macie 用户指南中的筛选 S3 存储桶清单](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeBuckets](#) 中的。

## 使用 Amazon Managed Grafana 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon Managed Grafana 搭配使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## **list-workspaces**

以下代码示例演示了如何使用 `list-workspaces`。

AWS CLI

在用户凭据指定的区域中列出该账户的工作空间

以下 `list-workspaces` 示例列出了账户所在区域的 Grafana 工作空间。

```
aws grafana list-workspaces
```

输出：

```
{
  "workspaces": [
    {
      "authentication": {
        "providers": [
          "AWS_SSO"
        ]
      },
      "created": "2022-04-04T16:20:21.796000-07:00",
      "description": "to test tags",
      "endpoint": "g-949e7b44df.grafana-workspace.us-east-1.amazonaws.com",
      "grafanaVersion": "8.2",
      "id": "g-949e7b44df",
      "modified": "2022-04-04T16:20:21.796000-07:00",
      "name": "testtag2",
    }
  ]
}
```

```
        "notificationDestinations": [
            "SNS"
        ],
        "status": "ACTIVE"
    },
    {
        "authentication": {
            "providers": [
                "AWS_SSO"
            ]
        },
        "created": "2022-04-20T10:22:15.115000-07:00",
        "description": "ww",
        "endpoint": "g-bffa51ed1b.grafana-workspace.us-east-1.amazonaws.com",
        "grafanaVersion": "8.2",
        "id": "g-bffa51ed1b",
        "modified": "2022-04-20T10:22:15.115000-07:00",
        "name": "ww",
        "notificationDestinations": [
            "SNS"
        ],
        "status": "ACTIVE"
    }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListWorkspaces](#)中的。

## MediaConnect 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 MediaConnect。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### add-flow-outputs

以下代码示例演示了如何使用 add-flow-outputs。

#### AWS CLI

向流程中添加输出

以下add-flow-outputs示例将输出添加到指定流程。

```
aws mediaconnect add-flow-outputs \  
--flow-arn arn:aws:mediacconnect:us-  
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BaseballGame \  
--outputs Description='NYC  
stream',Destination=192.0.2.12,Name=NYC,Port=3333,Protocol=rtp-  
fec,SmoothingLatency=100 Description='LA  
stream',Destination=203.0.113.9,Name=LA,Port=4444,Protocol=rtp-  
fec,SmoothingLatency=100
```

输出：

```
{  
  "Outputs": [  
    {  
      "Port": 3333,  
      "OutputArn": "arn:aws:mediacconnect:us-  
east-1:111122223333:output:2-3aBC45dEF67hiJ89-c34de5fG678h:NYC",  
      "Name": "NYC",  
      "Description": "NYC stream",  
      "Destination": "192.0.2.12",  
      "Transport": {  
        "Protocol": "rtp-fec",  
        "SmoothingLatency": 100  
      }  
    },  
    {  
      "Port": 4444,  
      "OutputArn": "arn:aws:mediacconnect:us-  
east-1:111122223333:output:2-987655dEF67hiJ89-c34de5fG678h:LA",
```



```

        "Name": "LA",
        "Description": "LA stream",
        "Destination": "203.0.113.9",
        "Transport": {
            "Protocol": "rtp-fec",
            "SmoothingLatency": 100
        }
    },
    "FlowArn": "arn:aws:mediacconnect:us-
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BaseballGame"
}

```

有关更多信息，请参阅 AWS Elemental MediaConnect 用户指南中的[向流程添加输出](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[AddFlowOutputs](#)中的。

## create-flow

以下代码示例演示了如何使用 create-flow。

### AWS CLI

#### 创建流程

以下 create-flow 示例使用指定配置创建流程。

```

aws mediacconnect create-flow \
  --availability-zone us-west-2c \
  --name ExampleFlow \
  --source Description='Example source,
backup',IngestPort=1055,Name=BackupSource,Protocol=rtp,WhitelistCidr=10.24.34.0/23

```

输出：

```

{
  "Flow": {
    "FlowArn": "arn:aws:mediacconnect:us-
east-1:123456789012:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:ExampleFlow",
    "AvailabilityZone": "us-west-2c",
    "EgressIp": "54.245.71.21",
    "Source": {
      "IngestPort": 1055,

```

```

    "SourceArn": "arn:aws:mediacconnect:us-
east-1:123456789012:source:2-3aBC45dEF67hiJ89-c34de5fG678h:BackupSource",
    "Transport": {
        "Protocol": "rtsp",
        "MaxBitrate": 80000000
    },
    "Description": "Example source, backup",
    "IngestIp": "54.245.71.21",
    "WhitelistCidr": "10.24.34.0/23",
    "Name": "mySource"
},
"Entitlements": [],
"Name": "ExampleFlow",
"Outputs": [],
"Status": "STANDBY",
"Description": "Example source, backup"
}
}

```

有关更多信息，请参阅 AWS Elemental MediaConnect 用户指南中的[创建流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateFlow](#)中的。

## delete-flow

以下代码示例演示了如何使用 delete-flow。

### AWS CLI

#### 删除流程

以下 delete-flow 示例删除了指定的流程。

```

aws mediacconnect delete-flow \
  --flow-arn arn:aws:mediacconnect:us-
east-1:123456789012:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:AwardsShow

```

#### 输出：

```

{
  "FlowArn": "arn:aws:mediacconnect:us-
east-1:123456789012:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:AwardsShow",
  "Status": "DELETING"
}

```

```
}
```

有关更多信息，请参阅 AWS Elemental MediaConnect 用户指南中的[删除流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteFlow](#)中的。

## describe-flow

以下代码示例演示了如何使用 describe-flow。

### AWS CLI

查看流程的详细信息

以下 describe-flow 示例显示了指定流程的详细信息，例如 ARN、可用区、状态、来源、授权和输出。

```
aws mediaconnect describe-flow \  
  --flow-arn arn:aws:mediaconnect:us-  
east-1:123456789012:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:AwardsShow
```

输出：

```
{  
  "Flow": {  
    "EgressIp": "54.201.4.39",  
    "AvailabilityZone": "us-west-2c",  
    "Status": "ACTIVE",  
    "FlowArn": "arn:aws:mediaconnect:us-  
east-1:123456789012:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:AwardsShow",  
    "Entitlements": [  
      {  
        "EntitlementArn": "arn:aws:mediaconnect:us-  
west-2:123456789012:entitlement:1-AaBb11CcDd22EeFf-34DE5fG12AbC:MyEntitlement",  
        "Description": "Assign to this account",  
        "Name": "MyEntitlement",  
        "Subscribers": [  
          "444455556666"  
        ]  
      }  
    ],  
    "Description": "NYC awards show",
```

```

    "Name": "AwardsShow",
    "Outputs": [
      {
        "Port": 2355,
        "Name": "NYC",
        "Transport": {
          "SmoothingLatency": 0,
          "Protocol": "rtp-fec"
        },
        "OutputArn": "arn:aws:mediacconnect:us-
east-1:123456789012:output:2-3aBC45dEF67hiJ89-c34de5fG678h:NYC",
        "Destination": "192.0.2.0"
      },
      {
        "Port": 3025,
        "Name": "LA",
        "Transport": {
          "SmoothingLatency": 0,
          "Protocol": "rtp-fec"
        },
        "OutputArn": "arn:aws:mediacconnect:us-
east-1:123456789012:output:2-987655dEF67hiJ89-c34de5fG678h:LA",
        "Destination": "192.0.2.0"
      }
    ],
    "Source": {
      "IngestIp": "54.201.4.39",
      "SourceArn": "arn:aws:mediacconnect:us-
east-1:123456789012:source:3-4aBC56dEF78hiJ90-4de5fG6Hi78Jk:ShowSource",
      "Transport": {
        "MaxBitrate": 80000000,
        "Protocol": "rtp"
      },
      "IngestPort": 1069,
      "Description": "Saturday night show",
      "Name": "ShowSource",
      "WhitelistCidr": "10.24.34.0/23"
    }
  }
}

```

有关更多信息，请参阅 AWS Elemental MediaConnect 用户指南中的[查看流程的详细信息](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeFlow](#)中的。

## grant-flow-entitlements

以下代码示例演示了如何使用 grant-flow-entitlements。

### AWS CLI

#### 授予流程权限

以下 grant-flow-entitlements 示例向指定的现有流程授予与其他 AWS 账户共享您的内容的权限。

```
aws mediacconnect grant-flow-entitlements \
  --flow-arn arn:aws:mediacconnect:us-
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BaseballGame \
  --entitlements Description='For
AnyCompany',Encryption={"Algorithm=aes128,KeyType=static-
key,RoleArn=arn:aws:iam::111122223333:role/MediaConnect-
ASM,SecretArn=arn:aws:secretsmanager:us-
west-2:111122223333:secret:mySecret1"},Name=AnyCompany_Entitlement,Subscribers=444455556666
Description='For Example Corp',Name=ExampleCorp,Subscribers=777788889999
```

输出：

```
{
  "Entitlements": [
    {
      "Name": "AnyCompany_Entitlement",
      "EntitlementArn": "arn:aws:mediacconnect:us-
west-2:111122223333:entitlement:1-11aa22bb11aa22bb-3333cccc4444:AnyCompany_Entitlement",
      "Subscribers": [
        "444455556666"
      ],
      "Description": "For AnyCompany",
      "Encryption": {
        "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:mySecret1",
        "Algorithm": "aes128",
        "RoleArn": "arn:aws:iam::111122223333:role/MediaConnect-ASM",
        "KeyType": "static-key"
      }
    },
    {
      "Name": "ExampleCorp",
```

```

        "EntitlementArn": "arn:aws:mediacconnect:us-
west-2:111122223333:entitlement:1-3333cccc4444dddd-1111aaaa2222:ExampleCorp",
        "Subscribers": [
            "777788889999"
        ],
        "Description": "For Example Corp"
    }
],
    "FlowArn": "arn:aws:mediacconnect:us-
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BaseballGame"
}

```

有关更多信息，请参阅 AWS Elemental MediaConnect 用户指南中的[授予流程](#)权限。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GrantFlowEntitlements](#)中的。

## list-entitlements

以下代码示例演示了如何使用 list-entitlements。

### AWS CLI

#### 查看权利列表

以下 list-entitlements 示例显示了已授予该账户的所有权利列表。

```
aws mediacconnect list-entitlements
```

输出：

```

{
  "Entitlements": [
    {
      "EntitlementArn": "arn:aws:mediacconnect:us-
west-2:111122223333:entitlement:1-11aa22bb11aa22bb-3333cccc4444:MyEntitlement",
      "EntitlementName": "MyEntitlement"
    }
  ]
}

```

有关更多信息，请参阅 AWS Elemental MediaConnect API 参考[ListEntitlements](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListEntitlements](#)中的。

## list-flows

以下代码示例演示了如何使用 `list-flows`。

### AWS CLI

#### 查看流程列表

以下`list-flows`示例显示了流程列表。

```
aws mediaconnect list-flows
```

输出：

```
{
  "Flows": [
    {
      "Status": "STANDBY",
      "SourceType": "OWNED",
      "AvailabilityZone": "us-west-2a",
      "Description": "NYC awards show",
      "Name": "AwardsShow",
      "FlowArn": "arn:aws:mediaconnect:us-east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:AwardsShow"
    },
    {
      "Status": "STANDBY",
      "SourceType": "OWNED",
      "AvailabilityZone": "us-west-2c",
      "Description": "LA basketball game",
      "Name": "BasketballGame",
      "FlowArn": "arn:aws:mediaconnect:us-east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BasketballGame"
    }
  ]
}
```

有关更多信息，请参阅 AWS Elemental MediaConnect 用户指南中的[查看流程列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFlows](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出 MediaConnect 资源的标签

以下 `list-tags-for-resource` 示例显示了与指定 MediaConnect 资源关联的标签键和值。

```
aws mediaconnect list-tags-for-resource \  
  --resource-arn arn:aws:mediaconnect:us-  
east-1:123456789012:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BasketballGame
```

输出：

```
{  
  "Tags": {  
    "region": "west",  
    "stage": "prod"  
  }  
}
```

有关更多信息 [ListTagsForResource TagResource](#)，请参阅《AWS 元素 MediaConnect API 参考》UntagResource 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## remove-flow-output

以下代码示例演示了如何使用 `remove-flow-output`。

### AWS CLI

从流程中移除输出

以下 `remove-flow-output` 示例从指定流程中删除输出。

```
aws mediaconnect remove-flow-output \  
  --flow-arn arn:aws:mediaconnect:us-  
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BaseballGame \  
  --output-arn arn:aws:mediaconnect:us-east-1:111122223333:output:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BaseballGame
```



```
--output-arn arn:aws:mediacconnect:us-  
east-1:111122223333:output:2-3aBC45dEF67hiJ89-c34de5fG678h:NYC
```

输出：

```
{  
  "FlowArn": "arn:aws:mediacconnect:us-  
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BaseballGame",  
  "OutputArn": "arn:aws:mediacconnect:us-  
east-1:111122223333:output:2-3aBC45dEF67hiJ89-c34de5fG678h:NYC"  
}
```

有关更多信息，请参阅 AWS Elemental MediaConnect 用户指南中的[从流程中移除输出](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RemoveFlowOutput](#)中的。

## revoke-flow-entitlement

以下代码示例演示了如何使用 revoke-flow-entitlement。

### AWS CLI

#### 撤销授权

以下 revoke-flow-entitlement 示例撤消了指定流程上的授权。

```
aws mediacconnect revoke-flow-entitlement \  
  --flow-arn arn:aws:mediacconnect:us-  
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BaseballGame \  
  --entitlement-arn arn:aws:mediacconnect:us-  
west-2:111122223333:entitlement:1-11aa22bb11aa22bb-3333cccc4444:AnyCompany_Entitlement
```

输出：

```
{  
  "FlowArn": "arn:aws:mediacconnect:us-  
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BaseballGame",  
  "EntitlementArn": "arn:aws:mediacconnect:us-  
west-2:111122223333:entitlement:1-11aa22bb11aa22bb-3333cccc4444:AnyCompany_Entitlement"  
}
```

有关更多信息，请参阅 AWS Elemental MediaConnect 用户指南中的[撤销授权](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RevokeFlowEntitlement](#)中的。

## start-flow

以下代码示例演示了如何使用 start-flow。

### AWS CLI

#### 启动流程

以下start-flow示例启动指定的流程。

```
aws mediaconnect start-flow \  
  --flow-arn arn:aws:mediaconnect:us-  
east-1:123456789012:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:AwardsShow
```

此命令不生成任何输出。输出：

```
{  
  "FlowArn": "arn:aws:mediaconnect:us-  
east-1:123456789012:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:AwardsShow",  
  "Status": "STARTING"  
}
```

有关更多信息，请参阅 AWS Elemental MediaConnect 用户指南中的[启动流程](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartFlow](#)中的。

## stop-flow

以下代码示例演示了如何使用 stop-flow。

### AWS CLI

#### 停止流动

以下stop-flow示例停止指定的流程。

```
aws mediaconnect stop-flow \  
  --flow-arn arn:aws:mediaconnect:us-  
east-1:123456789012:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:AwardsShow
```

输出：

```
{
  "Status": "STOPPING",
  "FlowArn": "arn:aws:mediaconnect:us-
east-1:123456789012:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:AwardsShow"
}
```

有关更多信息，请参阅 AWS Elemental MediaConnect 用户指南中的[停止数据流](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[StopFlow](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为 MediaConnect 资源添加标签

以下 tag-resource 示例向指定 MediaConnect 资源添加带有密钥名称和值的标签。

```
aws mediaconnect tag-resource \
  --resource-arn arn:aws:mediaconnect:us-
east-1:123456789012:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BasketballGame
  --tags region=west
```

此命令不生成任何输出。

有关更多信息 [ListTagsForResource TagResource](#)，请参阅《AWS 元素 MediaConnect API 参考》UntagResource 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从 MediaConnect 资源中移除标签

以下 `untag-resource` 示例从 MediaConnect 资源中移除具有指定密钥名称及其关联值的标签。

```
aws mediacconnect untag-resource \  
  --resource-arn arn:aws:mediacconnect:us-  
east-1:123456789012:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BasketballGame \  
  --tag-keys region
```

此命令不生成任何输出。

有关更多信息 [ListTagsForResource TagResource](#)，请参阅《AWS 元素 MediaConnect API 参考》UntagResource 中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-flow-entitlement

以下代码示例演示了如何使用 `update-flow-entitlement`。

### AWS CLI

#### 更新权利

以下 `update-flow-entitlement` 示例使用新的描述和订阅者更新了指定的权利。

```
aws mediacconnect update-flow-entitlement \  
  --flow-arn arn:aws:mediacconnect:us-  
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BaseballGame \  
  --entitlement-arn arn:aws:mediacconnect:us-  
west-2:111122223333:entitlement:1-11aa22bb11aa22bb-3333cccc4444:AnyCompany_Entitlement  
 \  
  --description 'For AnyCompany Affiliate' \  
  --subscribers 777788889999
```

输出：

```
{  
  "FlowArn": "arn:aws:mediacconnect:us-  
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BaseballGame",  
  "Entitlement": {  
    "Name": "AnyCompany_Entitlement",  
    "Description": "For AnyCompany Affiliate",
```

```

    "EntitlementArn": "arn:aws:mediacconnect:us-
west-2:111122223333:entitlement:1-11aa22bb11aa22bb-3333cccc4444:AnyCompany_Entitlement",
    "Encryption": {
      "KeyType": "static-key",
      "Algorithm": "aes128",
      "RoleArn": "arn:aws:iam::111122223333:role/MediaConnect-ASM",
      "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:mySecret1"
    },
    "Subscribers": [
      "777788889999"
    ]
  }
}

```

有关更多信息，请参阅 AWS Elemental MediaConnect 用户指南中的[更新授权](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateFlowEntitlement](#)中的。

## update-flow-output

以下代码示例演示了如何使用 update-flow-output。

### AWS CLI

#### 更新流程的输出

以下 update-flow-output 示例更新指定流程的输出。

```

aws mediacconnect update-flow-output \
  --flow-arn arn:aws:mediacconnect:us-
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BaseballGame \
  --output-arn arn:aws:mediacconnect:us-
east-1:111122223333:output:2-3aBC45dEF67hiJ89-c34de5fG678h:NYC \
  --port 3331

```

输出：

```

{
  "FlowArn": "arn:aws:mediacconnect:us-
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:BaseballGame",
  "Output": {
    "Name": "NYC",

```

```

    "Port": 3331,
    "Description": "NYC stream",
    "Transport": {
      "Protocol": "rtp-fec",
      "SmoothingLatency": 100
    },
    "OutputArn": "arn:aws:mediacconnect:us-
east-1:111122223333:output:2-3aBC45dEF67hiJ89-c34de5fG678h:NYC",
    "Destination": "192.0.2.12"
  }
}

```

有关更多信息，请参阅 AWS Elemental MediaConnect 用户指南中的[更新流程输出](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateFlowOutput](#)中的。

## update-flow-source

以下代码示例演示了如何使用 update-flow-source。

### AWS CLI

#### 更新现有流程的来源

以下 update-flow-source 示例更新了现有流程的来源。

```

aws mediacconnect update-flow-source \
  --flow-arn arn:aws:mediacconnect:us-
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:AwardsShow \
  --source-arn arn:aws:mediacconnect:us-
east-1:111122223333:source:3-4aBC56dEF78hiJ90-4de5fG6Hi78Jk:ShowSource \
  --description 'Friday night show' \
  --ingest-port 3344 \
  --protocol rtp-fec \
  --whitelist-cidr 10.24.34.0/23

```

输出：

```

{
  "FlowArn": "arn:aws:mediacconnect:us-
east-1:111122223333:flow:1-23aBC45dEF67hiJ8-12AbC34DE5fG:AwardsShow",
  "Source": {
    "IngestIp": "34.210.136.56",

```

```
    "WhitelistCidr": "10.24.34.0/23",
    "Transport": {
      "Protocol": "rtp-fec"
    },
    "IngestPort": 3344,
    "Name": "ShowSource",
    "Description": "Friday night show",
    "SourceArn": "arn:aws:mediacconnect:us-
east-1:111122223333:source:3-4aBC56dEF78hiJ90-4de5fG6Hi78Jk:ShowSource"
  }
}
```

有关更多信息，请参阅 AWS Elemental MediaConnect 用户指南中的[更新流程来源](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateFlowSource](#)中的。

## MediaConvert 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 MediaConvert。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **cancel-job**

以下代码示例演示了如何使用 `cancel-job`。

AWS CLI

取消队列中的作业

以下cancel-job示例取消带有 ID 1234567891234-abc123 的作业。您无法取消服务已开始处理的任务。

```
aws mediaconvert cancel-job \  
  --endpoint-url https://abcd1234.mediaconvert.region-name-1.amazonaws.com \  
  --region region-name-1 \  
  --id 1234567891234-abc123
```

要获取账户特定的端点，请使用 describe-endpoints，或发送不带端点的命令。该服务会返回错误和您的端点。

有关更多信息，请参阅《[AWS 元素 MediaConvert 用户指南](#)》中的“[使用AWS 元素 MediaConvert 任务](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelJob](#)中的。

## create-job-template

以下代码示例演示了如何使用 create-job-template。

### AWS CLI

#### 创建任务模板

以下create-job-template示例使用在系统上的文件job-template.json中指定的转码设置创建作业模板。

```
aws mediaconvert create-job-template \  
  --endpoint-url https://abcd1234.mediaconvert.region-name-1.amazonaws.com \  
  --region region-name-1 \  
  --name JobTemplate1 \  
  --cli-input-json file://~/job-template.json
```

如果您通过使用get-job-template然后修改作业模板文件来创建作业模板 JSON 文件，请移除该JobTemplate对象，但要保留 Settings 子对象。另外，请务必移除以下键值对：LastUpdated、ArnType、和 CreatedAt您可以在 JSON 文件或命令行中指定类别、描述、名称和队列。

要获取账户特定的端点，请使用 describe-endpoints，或发送不带端点的命令。该服务会返回错误和您的端点。



如果您的请求成功，该服务将返回您创建的作业模板的 JSON 规范。

有关更多信息，请参阅 [AWS Elemental MediaConvert 用户指南中的使用AWS 元素 MediaConvert 作业模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateJobTemplate](#)中的。

## create-job

以下代码示例演示了如何使用 create-job。

### AWS CLI

#### 创建作业

以下 create-job 示例使用在位于您发送命令的系统上的文件 job.json 中指定的设置创建转码作业。此 JSON 作业规范可以单独指定每个设置，引用作业模板或引用输出预设。

```
aws mediaconvert create-job \  
  --endpoint-url https://abcd1234.mediaconvert.region-name-1.amazonaws.com \  
  --region region-name-1 \  
  --cli-input-json file://~/job.json
```

您可以使用 AWS Elemental MediaConvert 控制台生成 JSON 作业规范，方法是选择您的作业设置，然后在“作业”部分底部选择“显示作业 JSON”。

要获取账户特定的端点，请使用 describe-endpoints，或发送不带端点的命令。该服务会返回错误和您的端点。

如果您的请求成功，则该服务将返回您在请求中发送的 JSON 作业规范。

有关更多信息，请参阅《[AWS 元素 MediaConvert 用户指南](#)》中的“[使用AWS 元素 MediaConvert 任务](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateJob](#)中的。

## create-preset

以下代码示例演示了如何使用 create-preset。

### AWS CLI

#### 创建自定输出预设

以下 `create-preset` 示例根据文件中指定的输出设置创建自定义输出预设 `preset.json`。您可以在 JSON 文件或命令行中指定类别、描述和名称。

```
aws mediaconvert create-preset \  
  --endpoint-url https://abcd1234.mediaconvert.region-name-1.amazonaws.com \  
  --region region-name-1 \  
  --cli-input-json file://~/preset.json
```

如果您通过使用 `get-preset` 然后修改输出文件来创建预设 JSON 文件，请确保删除以下键值对：`LastUpdated`、`ArnType`、和 `CreatedAt`

要获取账户特定的端点，请使用 `describe-endpoints`，或发送不带端点的命令。该服务会返回错误和您的端点。

有关更多信息，请参阅 [《AWS 元素用户指南》中的使用 AWS 元素 MediaConvert MediaConvert 输出预设](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreatePreset](#) 中的。

## create-queue

以下代码示例演示了如何使用 `create-queue`。

### AWS CLI

#### 创建自定义队列

以下 `create-queue` 示例创建了一个自定义的转码队列。

```
aws mediaconvert create-queue \  
  --endpoint-url https://abcd1234.mediaconvert.region-name-1.amazonaws.com \  
  --region region-name-1 \  
  --name Queue1 \  
  --description "Keep this queue empty unless job is urgent."
```

要获取账户特定的端点，请使用 `describe-endpoints`，或发送不带端点的命令。该服务会返回错误和您的端点。

输出：

```
{
```

```
"Queue": {
  "Status": "ACTIVE",
  "Name": "Queue1",
  "LastUpdated": 1518034928,
  "Arn": "arn:aws:mediaconvert:region-name-1:012345678998:queues/Queue1",
  "Type": "CUSTOM",
  "CreatedAt": 1518034928,
  "Description": "Keep this queue empty unless job is urgent."
}
```

有关更多信息，请参阅 [《AWS 元素 MediaConvert 用户指南》中的使用AWS 元素 MediaConvert 队列](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateQueue](#)中的。

## delete-job-template

以下代码示例演示了如何使用 delete-job-template。

### AWS CLI

#### 删除作业模板

以下delete-job-template示例删除了指定的自定义作业模板。

```
aws mediaconvert delete-job-template \
  --name "DASH Streaming" \
  --endpoint-url https://abcd1234.mediaconvert.us-west-2.amazonaws.com
```

此命令不生成任何输出。运行aws mediaconvert list-job-templates以确认您的模板已被删除。

有关更多信息，请参阅 [AWS Elemental MediaConvert 用户指南中的使用AWS 元素 MediaConvert 作业模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteJobTemplate](#)中的。

## delete-preset

以下代码示例演示了如何使用 delete-preset。

## AWS CLI

### 删除自定义按需队列

以下delete-preset示例删除了指定的自定义预设。

```
aws mediaconvert delete-preset \  
  --name SimpleMP4 \  
  --endpoint-url https://abcd1234.mediaconvert.us-west-2.amazonaws.com
```

此命令不生成任何输出。运行aws mediaconvert list-presets以确认您的预设已删除。

有关更多信息，请参阅《[AWS 元素用户指南](#)》中的[使用AWS 元素 MediaConvert MediaConvert 输出预设](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePreset](#)中的。

## delete-queue

以下代码示例演示了如何使用 delete-queue。

## AWS CLI

### 删除自定义按需队列

以下delete-queue示例删除了指定的自定义点播队列。

您无法删除默认队列。您无法删除具有活动定价计划或包含未处理作业的预留队列。

```
aws mediaconvert delete-queue \  
  --name Customer1 \  
  --endpoint-url https://abcd1234.mediaconvert.us-west-2.amazonaws.com
```

此命令不生成任何输出。运行aws mediaconvert list-queues以确认您的队列已被删除。

有关更多信息，请参阅《[AWS 元素 MediaConvert 用户指南](#)》中的[使用AWS 元素 MediaConvert 队列](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteQueue](#)中的。

## describe-endpoints

以下代码示例演示了如何使用 describe-endpoints。

## AWS CLI

获取账户特定的终端节点

以下describe-endpoints示例检索向服务发送任何其他请求所需的终端节点。

```
aws mediaconvert describe-endpoints
```

输出：

```
{
  "Endpoints": [
    {
      "Url": "https://abcd1234.mediaconvert.region-name-1.amazonaws.com"
    }
  ]
}
```

有关更多信息，请参阅 AWS Elemental API 参考中的 AP MediaConvert | MediaConvert [使用入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEndpoints](#)中的。

## get-job-template

以下代码示例演示了如何使用 get-job-template。

### AWS CLI

获取作业模板的详细信息

以下get-job-template示例显示了指定自定义作业模板的 JSON 定义。

```
aws mediaconvert get-job-template \
  --name "DASH Streaming" \
  --endpoint-url https://abcd1234.mediaconvert.us-east-1.amazonaws.com
```

输出：

```
{
```

```
"JobTemplate": {
  "StatusUpdateInterval": "SECONDS_60",
  "LastUpdated": 1568652998,
  "Description": "Create a DASH streaming ABR stack",
  "CreatedAt": 1568652998,
  "Priority": 0,
  "Name": "DASH Streaming",
  "Settings": {
    ...<truncatedforbrevity>...
  },
  "Arn": "arn:aws:mediaconvert:us-west-2:123456789012:jobTemplates/DASH
Streaming",
  "Type": "CUSTOM"
}
```

有关更多信息，请参阅 [AWS Elemental MediaConvert 用户指南中的使用AWS 元素 MediaConvert 作业模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetJobTemplate](#)中的。

## get-job

以下代码示例演示了如何使用 get-job。

### AWS CLI

获取特定作业的详细信息

以下示例请求 ID 为 1234567890987-1ab2c3 的作业的信息，在本示例中，该作业以错误结尾。

```
aws mediaconvert get-job \
  --endpoint-url https://abcd1234.mediaconvert.region-name-1.amazonaws.com \
  --region region-name-1 \
  --id 1234567890987-1ab2c3
```

要获取账户特定的端点，请使用 describe-endpoints，或发送不带端点的命令。该服务会返回错误和您的端点。

如果您的请求成功，该服务将返回一个含作业信息的 JSON 文件，包括作业设置、任何返回的错误以及其他作业数据，如下所示：

```
{
  "Job": {
    "Status": "ERROR",
    "Queue": "arn:aws:mediaconvert:region-name-1:012345678998:queues/Queue1",
    "Settings": {
      ...<truncated for brevity>...
    },
    "ErrorMessage": "Unable to open input file [s3://my-input-bucket/file-name.mp4]: [Failed probe/open: [Failed to read data: AssumeRole failed]]",
    "ErrorCode": 1434,
    "Role": "arn:aws:iam::012345678998:role/MediaConvertServiceRole",
    "Arn": "arn:aws:mediaconvert:us-west-1:012345678998:jobs/1234567890987-1ab2c3",
    "UserMetadata": {},
    "Timing": {
      "FinishTime": 1517442131,
      "SubmitTime": 1517442103,
      "StartTime": 1517442104
    },
    "Id": "1234567890987-1ab2c3",
    "CreatedAt": 1517442103
  }
}
```

有关更多信息，请参阅 [《AWS 元素 MediaConvert 用户指南》](#) 中的“使用 AWS 元素 MediaConvert 任务”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetJob](#) 中的。

## get-preset

以下代码示例演示了如何使用 get-preset。

### AWS CLI

获取特定预设的详细信息

以下 get-preset 示例请求指定自定义预设的 JSON 定义。

```
aws mediaconvert get-preset \
  --name SimpleMP4 \
  --endpoint-url https://abcd1234.mediaconvert.us-west-2.amazonaws.com
```

输出：

```
{
  "Preset": {
    "Description": "Creates basic MP4 file. No filtering or preprocessing.",
    "Arn": "arn:aws:mediaconvert:us-west-2:123456789012:presets/SimpleMP4",
    "LastUpdated": 1568843141,
    "Name": "SimpleMP4",
    "Settings": {
      "ContainerSettings": {
        "Mp4Settings": {
          "FreeSpaceBox": "EXCLUDE",
          "CslgAtom": "INCLUDE",
          "MoovPlacement": "PROGRESSIVE_DOWNLOAD"
        },
        "Container": "MP4"
      },
      "AudioDescriptions": [
        {
          "LanguageCodeControl": "FOLLOW_INPUT",
          "AudioTypeControl": "FOLLOW_INPUT",
          "CodecSettings": {
            "AacSettings": {
              "RawFormat": "NONE",
              "CodecProfile": "LC",
              "AudioDescriptionBroadcasterMix": "NORMAL",
              "SampleRate": 48000,
              "Bitrate": 96000,
              "RateControlMode": "CBR",
              "Specification": "MPEG4",
              "CodingMode": "CODING_MODE_2_0"
            },
            "Codec": "AAC"
          }
        }
      ],
      "VideoDescription": {
        "RespondToAfd": "NONE",
        "TimecodeInsertion": "DISABLED",
        "Sharpness": 50,
        "ColorMetadata": "INSERT",
        "CodecSettings": {
          "H264Settings": {
            "FramerateControl": "INITIALIZE_FROM_SOURCE",
```



```

        "SpatialAdaptiveQuantization": "ENABLED",
        "Softness": 0,
        "Telecine": "NONE",
        "CodecLevel": "AUTO",
        "QualityTuningLevel": "SINGLE_PASS",
        "UnregisteredSeiTimecode": "DISABLED",
        "Slices": 1,
        "Syntax": "DEFAULT",
        "GopClosedCadence": 1,
        "AdaptiveQuantization": "HIGH",
        "EntropyEncoding": "CABAC",
        "InterlaceMode": "PROGRESSIVE",
        "ParControl": "INITIALIZE_FROM_SOURCE",
        "NumberBFramesBetweenReferenceFrames": 2,
        "GopSizeUnits": "FRAMES",
        "RepeatPps": "DISABLED",
        "CodecProfile": "MAIN",
        "FieldEncoding": "PAFF",
        "GopSize": 90.0,
        "SlowPal": "DISABLED",
        "SceneChangeDetect": "ENABLED",
        "GopBReference": "DISABLED",
        "RateControlMode": "CBR",
        "FramerateConversionAlgorithm": "DUPLICATE_DROP",
        "FlickerAdaptiveQuantization": "DISABLED",
        "DynamicSubGop": "STATIC",
        "MinIInterval": 0,
        "TemporalAdaptiveQuantization": "ENABLED",
        "Bitrate": 400000,
        "NumberReferenceFrames": 3
    },
    "Codec": "H_264"
},
"AfdSignaling": "NONE",
"AntiAlias": "ENABLED",
"ScalingBehavior": "DEFAULT",
"DropFrameTimecode": "ENABLED"
}
},
"Type": "CUSTOM",
"CreatedAt": 1568841521
}
}

```

有关更多信息，请参阅 [《AWS 元素用户指南》中的使用AWS 元素 MediaConvert MediaConvert 输出预设](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPreset](#)中的。

## get-queue

以下代码示例演示了如何使用 get-queue。

### AWS CLI

获取队列的详细信息

以下get-queue示例检索指定自定义队列的详细信息。

```
aws mediaconvert get-queue \  
  --name Customer1 \  
  --endpoint-url https://abcd1234.mediaconvert.us-west-2.amazonaws.com
```

输出：

```
{  
  "Queue": {  
    "LastUpdated": 1526428502,  
    "Type": "CUSTOM",  
    "SubmittedJobsCount": 0,  
    "Status": "ACTIVE",  
    "PricingPlan": "ON_DEMAND",  
    "CreatedAt": 1526428502,  
    "ProgressingJobsCount": 0,  
    "Arn": "arn:aws:mediaconvert:us-west-2:123456789012:queues/Customer1",  
    "Name": "Customer1"  
  }  
}
```

有关更多信息，请参阅 [《AWS 元素 MediaConvert 用户指南》中的使用AWS 元素 MediaConvert 队列](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetQueue](#)中的。

## list-job-templates

以下代码示例演示了如何使用 list-job-templates。

### AWS CLI

示例 1：列出您的自定义作业模板

以下list-job-templates示例列出了当前区域中的所有自定义作业模板。要列出系统作业模板，请参阅下一个示例。

```
aws mediaconvert list-job-templates \
  --endpoint-url https://abcd1234.mediaconvert.us-west-2.amazonaws.com
```

输出：

```
{
  "JobTemplates": [
    {
      "Description": "Create a DASH streaming ABR stack",
      "Arn": "arn:aws:mediaconvert:us-west-2:123456789012:jobTemplates/DASH
Streaming",
      "Name": "DASH Streaming",
      "LastUpdated": 1568653007,
      "Priority": 0,
      "Settings": {
        ...<truncatedforbrevity>...
      },
      "Type": "CUSTOM",
      "StatusUpdateInterval": "SECONDS_60",
      "CreatedAt": 1568653007
    },
    {
      "Description": "Create a high-res file",
      "Arn": "arn:aws:mediaconvert:us-west-2:123456789012:jobTemplates/File",
      "Name": "File",
      "LastUpdated": 1568653007,
      "Priority": 0,
      "Settings": {
        ...<truncatedforbrevity>...
      },
      "Type": "CUSTOM",
      "StatusUpdateInterval": "SECONDS_60",
```

```

        "CreatedAt": 1568653023
      }
    ]
  }

```

## 示例 2：列出 MediaConvert 系统作业模板

以下 `list-job-templates` 示例列出了所有系统作业模板。

```

aws mediaconvert list-job-templates \
  --endpoint-url https://abcd1234.mediaconvert.us-east-1.amazonaws.com \
  --list-by SYSTEM

```

输出：

```

{
  "JobTemplates": [
    {
      "CreatedAt": 1568321779,
      "Arn": "arn:aws:mediaconvert:us-east-1:123456789012:jobTemplates/System-
Generic_Mp4_Hev1_Avc_Aac_Sdr_Qvbr",
      "Name": "System-Generic_Mp4_Hev1_Avc_Aac_Sdr_Qvbr",
      "Description": "GENERIC, MP4, AVC + HEV1(HEVC,SDR), AAC, SDR, QVBR",
      "Category": "GENERIC",
      "Settings": {
        "AdAvailOffset": 0,
        "OutputGroups": [
          {
            "Outputs": [
              {
                "Extension": "mp4",
                "Preset": "System-
Generic_Hd_Mp4_Avc_Aac_16x9_Sdr_1280x720p_30Hz_5Mbps_Qvbr_Vq9",
                "NameModifier":
                "_Generic_Hd_Mp4_Avc_Aac_16x9_Sdr_1280x720p_30Hz_5000Kbps_Qvbr_Vq9"
              },
              {
                "Extension": "mp4",
                "Preset": "System-
Generic_Hd_Mp4_Avc_Aac_16x9_Sdr_1920x1080p_30Hz_10Mbps_Qvbr_Vq9",
                "NameModifier":
                "_Generic_Hd_Mp4_Avc_Aac_16x9_Sdr_1920x1080p_30Hz_10000Kbps_Qvbr_Vq9"
              }
            ]
          }
        ]
      }
    }
  ]
}

```

```

        {
            "Extension": "mp4",
            "Preset": "System-
Generic_Sd_Mp4_Avc_Aac_16x9_Sdr_640x360p_30Hz_0.8Mbps_Qvbr_Vq7",
            "NameModifier":
            "_Generic_Sd_Mp4_Avc_Aac_16x9_Sdr_640x360p_30Hz_800Kbps_Qvbr_Vq7"
        },
        {
            "Extension": "mp4",
            "Preset": "System-
Generic_Hd_Mp4_Hev1_Aac_16x9_Sdr_1280x720p_30Hz_4Mbps_Qvbr_Vq9",
            "NameModifier":
            "_Generic_Hd_Mp4_Hev1_Aac_16x9_Sdr_1280x720p_30Hz_4000Kbps_Qvbr_Vq9"
        },
        {
            "Extension": "mp4",
            "Preset": "System-
Generic_Hd_Mp4_Hev1_Aac_16x9_Sdr_1920x1080p_30Hz_8Mbps_Qvbr_Vq9",
            "NameModifier":
            "_Generic_Hd_Mp4_Hev1_Aac_16x9_Sdr_1920x1080p_30Hz_8000Kbps_Qvbr_Vq9"
        },
        {
            "Extension": "mp4",
            "Preset": "System-
Generic_Uhd_Mp4_Hev1_Aac_16x9_Sdr_3840x2160p_30Hz_12Mbps_Qvbr_Vq9",
            "NameModifier":
            "_Generic_Uhd_Mp4_Hev1_Aac_16x9_Sdr_3840x2160p_30Hz_12000Kbps_Qvbr_Vq9"
        }
    ],
    "OutputGroupSettings": {
        "FileGroupSettings": {

        },
        "Type": "FILE_GROUP_SETTINGS"
    },
    "Name": "File Group"
}
]
},
"Type": "SYSTEM",
"LastUpdated": 1568321779
},
...<truncatedforbrevity>...
]

```

```
}
```

有关更多信息，请参阅 [AWS Elemental MediaConvert 用户指南中的使用AWS 元素 MediaConvert 作业模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListJobTemplates](#)中的。

## list-jobs

以下代码示例演示了如何使用 `list-jobs`。

### AWS CLI

获取某个区域中所有作业的详细信息

以下示例请求您在指定区域所有作业的信息。

```
aws mediaconvert list-jobs \  
  --endpoint-url https://abcd1234.mediaconvert.region-name-1.amazonaws.com \  
  --region region-name-1
```

要获取账户特定的端点，请使用 `describe-endpoints`，或发送不带端点的命令。该服务会返回错误和您的端点。

有关更多信息，请参阅《[AWS 元素 MediaConvert 用户指南](#)》中的“[使用AWS 元素 MediaConvert 任务](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListJobs](#)中的。

## list-presets

以下代码示例演示了如何使用 `list-presets`。

### AWS CLI

示例 1：列出您的自定义输出预设

以下`list-presets`示例列出了您的自定义输出预设。要列出系统预设，请参阅下一个示例。

```
aws mediaconvert list-presets \  
  --endpoint-url https://abcd1234.mediaconvert.us-west-2.amazonaws.com
```

输出：

```
{
  "Presets": [
    {
      "Name": "SimpleMP4",
      "CreatedAt": 1568841521,
      "Settings": {
        .....
      },
      "Arn": "arn:aws:mediaconvert:us-east-1:003235472598:presets/SimpleMP4",
      "Type": "CUSTOM",
      "LastUpdated": 1568843141,
      "Description": "Creates basic MP4 file. No filtering or preprocessing."
    },
    {
      "Name": "SimpleTS",
      "CreatedAt": 1568843113,
      "Settings": {
        ... truncated for brevity ...
      },
      "Arn": "arn:aws:mediaconvert:us-east-1:003235472598:presets/SimpleTS",
      "Type": "CUSTOM",
      "LastUpdated": 1568843113,
      "Description": "Create a basic transport stream."
    }
  ]
}
```

示例 2：列出系统输出预设

以下list-presets示例列出了可用的 MediaConvert 系统预设。要列出您的自定义预设，请参阅前面的示例。

```
aws mediaconvert list-presets \
  --list-by SYSTEM \
  --endpoint-url https://abcd1234.mediaconvert.us-west-2.amazonaws.com
```

输出：

```
{
  "Presets": [
    {
```

```

    "Arn": "arn:aws:mediaconvert:us-west-2:123456789012:presets/System-
Avc_16x9_1080p_29_97fps_8500kbps",
    "Name": "System-Avc_16x9_1080p_29_97fps_8500kbps",
    "CreatedAt": 1568321789,
    "Description": "Wifi, 1920x1080, 16:9, 29.97fps, 8500kbps",
    "LastUpdated": 1568321789,
    "Type": "SYSTEM",
    "Category": "HLS",
    "Settings": {
    ...<output settings removed for brevity>...
    }
  },
  ...<list of presets shortened for brevity>...

  {
    "Arn": "arn:aws:mediaconvert:us-east-1:123456789012:presets/System-
Xdcam_HD_1080i_29_97fps_35mpbs",
    "Name": "System-Xdcam_HD_1080i_29_97fps_35mpbs",
    "CreatedAt": 1568321790,
    "Description": "XDCAM MPEG HD, 1920x1080i, 29.97fps, 35mbps",
    "LastUpdated": 1568321790,
    "Type": "SYSTEM",
    "Category": "MXF",
    "Settings": {
    ...<output settings removed for brevity>...
    }
  }
]
}

```

有关更多信息，请参阅 [《AWS 元素用户指南》中的使用AWS 元素 MediaConvert MediaConvert 输出预设](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPresets](#)中的。

## list-queues

以下代码示例演示了如何使用 list-queues。

### AWS CLI

列出您的队列



以下 `list-queues` 示例列出了您的所有 MediaConvert 队列。

```
aws mediaconvert list-queues \  
  --endpoint-url https://abcd1234.mediaconvert.us-west-2.amazonaws.com
```

输出：

```
{  
  "Queues": [  
    {  
      "PricingPlan": "ON_DEMAND",  
      "Type": "SYSTEM",  
      "Status": "ACTIVE",  
      "CreatedAt": 1503451595,  
      "Name": "Default",  
      "SubmittedJobsCount": 0,  
      "ProgressingJobsCount": 0,  
      "Arn": "arn:aws:mediaconvert:us-west-2:123456789012:queues/Default",  
      "LastUpdated": 1534549158  
    },  
    {  
      "PricingPlan": "ON_DEMAND",  
      "Type": "CUSTOM",  
      "Status": "ACTIVE",  
      "CreatedAt": 1537460025,  
      "Name": "Customer1",  
      "SubmittedJobsCount": 0,  
      "Description": "Jobs we run for our cusotmer.",  
      "ProgressingJobsCount": 0,  
      "Arn": "arn:aws:mediaconvert:us-west-2:123456789012:queues/Customer1",  
      "LastUpdated": 1537460025  
    },  
    {  
      "ProgressingJobsCount": 0,  
      "Status": "ACTIVE",  
      "Name": "transcode-library",  
      "SubmittedJobsCount": 0,  
      "LastUpdated": 1564066204,  
      "ReservationPlan": {  
        "Status": "ACTIVE",  
        "ReservedSlots": 1,  
        "PurchasedAt": 1564066203,  
        "Commitment": "ONE_YEAR",  
      }  
    }  
  ]  
}
```

```

        "ExpiresAt": 1595688603,
        "RenewalType": "EXPIRE"
    },
    "PricingPlan": "RESERVED",
    "Arn": "arn:aws:mediaconvert:us-west-2:123456789012:queues/transcode-
library",
    "Type": "CUSTOM",
    "CreatedAt": 1564066204
  }
]
}

```

有关更多信息，请参阅《[AWS 元素 MediaConvert 用户指南](#)》中的使用AWS 元素 MediaConvert 队列。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListQueues](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出 MediaConvert 队列、作业模板或输出预设上的标签

以下`list-tags-for-resource`示例列出了指定输出预设上的标签。

```

aws mediaconvert list-tags-for-resource \
  --arn arn:aws:mediaconvert:us-west-2:123456789012:presets/SimpleMP4 \
  --endpoint-url https://abcd1234.mediaconvert.us-west-2.amazonaws.com

```

输出：

```

{
  "ResourceTags": {
    "Tags": {
      "customer": "zippyVideo"
    },
    "Arn": "arn:aws:mediaconvert:us-west-2:123456789012:presets/SimpleMP4"
  }
}

```

有关更多信息，请参阅 Elemental 用户指南中的为 [AWS 元素 MediaConvert 队列、Job 模板和输出预设添加标签](#)。AWS MediaConvert

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## update-job-template

以下代码示例演示了如何使用 update-job-template。

### AWS CLI

#### 更改作业模板

以下 update-job-template 示例将指定自定义作业模板的 JSON 定义替换为提供的文件中的 JSON 定义。

```
aws mediaconvert update-job-template --name File1--endpoint-url https://
abcd1234.mediaconvert.us-west-2.amazonaws.com -- file: ~/ .json cli-input-json job-template-
update
```

job-template-update.json 的内容：

```
{
  "Description": "A simple job template that generates a single file output.",
  "Queue": "arn:aws:mediaconvert:us-east-1:012345678998:queues/Default",
  "Name": "SimpleFile",
  "Settings": {
    "OutputGroups": [
      {
        "Name": "File Group",
        "Outputs": [
          {
            "ContainerSettings": {
              "Container": "MP4",
              "Mp4Settings": {
                "CslgAtom": "INCLUDE",
                "FreeSpaceBox": "EXCLUDE",
                "MoovPlacement": "PROGRESSIVE_DOWNLOAD"
              }
            }
          }
        ],
        "VideoDescription": {
          "ScalingBehavior": "DEFAULT",
          "TimecodeInsertion": "DISABLED",
```

```
"AntiAlias": "ENABLED",
"Sharpness": 50,
"CodecSettings": {
  "Codec": "H_264",
  "H264Settings": {
    "InterlaceMode": "PROGRESSIVE",
    "NumberReferenceFrames": 3,
    "Syntax": "DEFAULT",
    "Softness": 0,
    "GopClosedCadence": 1,
    "GopSize": 90,
    "Slices": 1,
    "GopBReference": "DISABLED",
    "SlowPal": "DISABLED",
    "SpatialAdaptiveQuantization": "ENABLED",
    "TemporalAdaptiveQuantization": "ENABLED",
    "FlickerAdaptiveQuantization": "DISABLED",
    "EntropyEncoding": "CABAC",
    "Bitrate": 400000,
    "FramerateControl": "INITIALIZE_FROM_SOURCE",
    "RateControlMode": "CBR",
    "CodecProfile": "MAIN",
    "Telecine": "NONE",
    "MinIInterval": 0,
    "AdaptiveQuantization": "HIGH",
    "CodecLevel": "AUTO",
    "FieldEncoding": "PAFF",
    "SceneChangeDetect": "ENABLED",
    "QualityTuningLevel": "SINGLE_PASS",
    "FramerateConversionAlgorithm": "DUPLICATE_DROP",
    "UnregisteredSeiTimecode": "DISABLED",
    "GopSizeUnits": "FRAMES",
    "ParControl": "INITIALIZE_FROM_SOURCE",
    "NumberBFramesBetweenReferenceFrames": 2,
    "RepeatPps": "DISABLED",
    "DynamicSubGop": "STATIC"
  }
},
"AfdSignaling": "NONE",
"DropFrameTimecode": "ENABLED",
"RespondToAfd": "NONE",
"ColorMetadata": "INSERT"
},
"AudioDescriptions": [
```

```
    {
      "AudioTypeControl": "FOLLOW_INPUT",
      "CodecSettings": {
        "Codec": "AAC",
        "AacSettings": {
          "AudioDescriptionBroadcasterMix": "NORMAL",
          "Bitrate": 96000,
          "RateControlMode": "CBR",
          "CodecProfile": "LC",
          "CodingMode": "CODING_MODE_2_0",
          "RawFormat": "NONE",
          "SampleRate": 48000,
          "Specification": "MPEG4"
        }
      },
      "LanguageCodeControl": "FOLLOW_INPUT"
    }
  ]
},
"OutputGroupSettings": {
  "Type": "FILE_GROUP_SETTINGS",
  "FileGroupSettings": {}
}
],
"AdAvailOffset": 0
},
"StatusUpdateInterval": "SECONDS_60",
"Priority": 0
}
```

即使请求导致错误，系统也会返回您随请求发送的 JSON 负载。因此，返回的 JSON 不一定是作业模板的新定义。

由于 JSON 有效负载可能很长，因此您可能需要向上滚动才能看到任何错误消息。

有关更多信息，请参阅 [AWS Elemental MediaConvert 用户指南中的使用AWS 元素 MediaConvert 作业模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateJobTemplate](#)中的。

## update-preset

以下代码示例演示了如何使用 update-preset。

### AWS CLI

#### 更改预设

以下update-preset示例替换了指定预设的描述。

```
aws mediaconvert update-preset \  
--name Customer1 \  
--description "New description text." \  
--endpoint-url https://abcd1234.mediaconvert.us-west-2.amazonaws.com
```

此命令不生成任何输出。输出：

```
{  
  "Preset": {  
    "Arn": "arn:aws:mediaconvert:us-east-1:003235472598:presets/SimpleMP4",  
    "Settings": {  
      ...<output settings removed for brevity>...  
    },  
    "Type": "CUSTOM",  
    "LastUpdated": 1568938411,  
    "Description": "New description text.",  
    "Name": "SimpleMP4",  
    "CreatedAt": 1568938240  
  }  
}
```

有关更多信息，请参阅 [《AWS 元素用户指南》中的使用AWS 元素 MediaConvert MediaConvert 输出预设](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePreset](#)中的。

## update-queue

以下代码示例演示了如何使用 update-queue。

### AWS CLI

#### 更改队列

以下update-queue示例通过将指定队列的状态更改为，来暂停指定队列。PAUSED

```
aws mediaconvert update-queue \  
--name Customer1 \  
--status PAUSED \  
--endpoint-url https://abcd1234.mediaconvert.us-west-2.amazonaws.com
```

输出：

```
{  
  "Queue": {  
    "LastUpdated": 1568839845,  
    "Status": "PAUSED",  
    "ProgressingJobsCount": 0,  
    "CreatedAt": 1526428516,  
    "Arn": "arn:aws:mediaconvert:us-west-1:123456789012:queues/Customer1",  
    "Name": "Customer1",  
    "SubmittedJobsCount": 0,  
    "PricingPlan": "ON_DEMAND",  
    "Type": "CUSTOM"  
  }  
}
```

有关更多信息，请参阅 [《AWS 元素 MediaConvert 用户指南》中的使用AWS 元素 MediaConvert 队列](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateQueue](#)中的。

## MediaLive 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 MediaLive。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

## 操作

### create-channel

以下代码示例演示了如何使用 create-channel。

#### AWS CLI

##### 创建频道

以下 create-channel 示例通过传入包含您要指定的参数的 JSON 文件来创建频道。

此示例中的频道接收一个 HLS PULL 输入，该输入连接到包含视频、音频和嵌入式字幕的源。该频道创建一个以 Akamai 服务器为目标的 HLS 输出组。输出组包含两个输出：一个用于 H.265 视频和 AAC 音频，另一个用于 Web-VTT 字幕，仅提供英语版本。

此示例频道的 JSON 包含使用 HLS PULL 输入并生成以 Akamai 为目的地的 HLS 输出组的频道所需的最低参数。JSON 包含以下主要部分：

**InputAttachments**，它为音频指定一个来源，为字幕指定一个来源。它没有指定视频选择器，这意味着它会 MediaLive 提取它在源中找到的第一个视频。**Destinations**，其中包含此频道中单个输出组的两个 IP 地址 (URL)。这些地址需要密码。**EncoderSettings**，其中包含小节。**AudioDescriptions**，它指定该频道包含一个音频输出资源，该资源使用来自 **InputAttachments** 的音频并产生 AAC 格式的音频。**CaptionDescriptions**，它指定该频道包含一个字幕输出资源，该资源使用来自的字幕 **InputAttachments**，并以 Web-VTT 格式生成字幕。**VideoDescriptions**，它指定该频道包含一个具有指定分辨率的视频输出资源。**OutputGroups**，它指定了输出组。在此示例中，有一个名为的组 **Akamai**。连接是使用 HLS PUT 进行的。输出组包含两个输出。一个输出用于视频资源（已命名 **Video\_high**）和音频资源（已命名 **Audio\_EN**）。一个输出是字幕资源（已命名 **WebVTT\_EN**）。

在此示例中，某些参数不包含任何值或包含嵌套的空参数。**OutputSettings** 例如，**Video\_and\_audio** 输出包含几个以空参数 **m3u8Settings** 结尾的嵌套参数。必须包含此参数，但您可以省略其一个、多个或所有子项，这意味着子项将采用其默认值或为空。

适用于此示例频道但未在此文件中指定的所有参数要么采用默认值，要么设置为 null，要么采用生成的唯一值 **MediaLive**。

```
aws medialive create-channel \  
  --cli-input-json file://channel-in-hls-out-hls-akamai.json
```



channel-in-hls-out-hls-akamai.json 的内容：

```
{
  "Name": "News_West",
  "RoleArn": "arn:aws:iam::111122223333:role/MediaLiveAccessRole",
  "InputAttachments": [
    {
      "InputAttachmentName": "local_news",
      "InputId": "1234567",
      "InputSettings": {
        "AudioSelectors": [
          {
            "Name": "English-Audio",
            "SelectorSettings": {
              "AudioLanguageSelection": {
                "LanguageCode": "EN"
              }
            }
          }
        ],
        "CaptionSelectors": [
          {
            "LanguageCode": "ENE",
            "Name": "English_embedded"
          }
        ]
      }
    }
  ],
  "Destinations": [
    {
      "Id": "akamai-server-west",
      "Settings": [
        {
          "PasswordParam": "/medialive/examplecorp1",
          "Url": "http://203.0.113.55/news/news_west",
          "Username": "examplecorp"
        },
        {
          "PasswordParam": "/medialive/examplecorp2",
          "Url": "http://203.0.113.82/news/news_west",
          "Username": "examplecorp"
        }
      ]
    }
  ]
}
```

```
    }
  ],
  "EncoderSettings": {
    "AudioDescriptions": [
      {
        "AudioSelectorName": "English-Audio",
        "CodecSettings": {
          "AacSettings": {}
        },
        "Name": "Audio_EN"
      }
    ],
    "CaptionDescriptions": [
      {
        "CaptionSelectorName": "English_embedded",
        "DestinationSettings": {
          "WebvttDestinationSettings": {}
        },
        "Name": "WebVTT_EN"
      }
    ],
    "VideoDescriptions": [
      {
        "Height": 720,
        "Name": "Video_high",
        "Width": 1280
      }
    ],
    "OutputGroups": [
      {
        "Name": "Akamai",
        "OutputGroupSettings": {
          "HlsGroupSettings": {
            "Destination": {
              "DestinationRefId": "akamai-server-west"
            },
            "HlsCdnSettings": {
              "HlsBasicPutSettings": {}
            }
          }
        },
        "Outputs": [
          {
            "AudioDescriptionNames": [
```

```

        "Audio_EN"
    ],
    "OutputName": "Video_and_audio",
    "OutputSettings": {
        "HlsOutputSettings": {
            "HlsSettings": {
                "StandardHlsSettings": {
                    "M3u8Settings": {}
                }
            },
            "NameModifier": "_1"
        }
    },
    "VideoDescriptionName": "Video_high"
},
{
    "CaptionDescriptionNames": [
        "WebVTT_EN"
    ],
    "OutputName": "Captions-WebVTT",
    "OutputSettings": {
        "HlsOutputSettings": {
            "HlsSettings": {
                "StandardHlsSettings": {
                    "M3u8Settings": {}
                }
            },
            "NameModifier": "_2"
        }
    }
}
]
}
},
"TimecodeConfig": {
    "Source": "EMBEDDED"
}
}
}

```

输出：

输出重复了 JSON 文件的内容，再加上以下值。所有参数均按字母顺序排序。

ARN对于频道。ARN 的最后部分是唯一的频道 ID。EgressEndpoints在此示例通道中为空，因为它仅用于 PUSH 输入。当它应用时，它会显示 MediaLive 该内容被推送到的地址。OutputGroups, Outputs。它们显示了输出组和输出的所有参数，包括您未包含但与此通道相关的参数。参数可能为空（可能表示此频道配置中已禁用参数或功能），也可能显示将适用的默认值。LogLevel设置为默认值（已禁用）。Tags设置为默认值（空）。PipelinesRunningCount并State显示频道的当前状态。

有关更多信息，请参阅 AWS Elemental MediaLive 用户指南中的[从头开始创建频道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateChannel](#)中的。

## create-input

以下代码示例演示了如何使用 create-input。

### AWS CLI

#### 创建输入

以下create-input示例通过传入包含适用于此类HLS PULL输入的参数的 JSON 文件来创建输入。此示例输入的 JSON 为输入指定了两个来源（地址），以支持采集中的冗余。这些地址需要密码。

```
aws medialive create-input \  
  --cli-input-json file://input-hls-pull-news.json
```

input-hls-pull-news.json 的内容：

```
{  
  "Name": "local_news",  
  "RequestId": "cli000059",  
  "Sources": [  
    {  
      "Url": "https://203.0.113.13/newschannel/anytownusa.m3u8",  
      "Username": "examplecorp",  
      "PasswordParam": "/medialive/examplecorp1"  
    },  
    {  
      "Url": "https://198.51.100.54/fillervideos/oceanwaves.mp4",  
      "Username": "examplecorp",  
      "PasswordParam": "examplecorp2"  
    }  
  ]  
}
```

```
    }  
  ],  
  "Type": "URL_PULL"  
}
```

输出：

输出重复了 JSON 文件的内容，再加上以下值。所有参数均按字母顺序排序。

Arn用于输入。ARN 的最后一部分是唯一的输入 ID。Attached Channels，对于新创建的输入，它始终为空。Destinations，在本示例中为空，因为它仅与 PUSH 输入一起使用。Id对于输入，与 ARN 中的 ID 相同。MediaConnectFlows，在本示例中为空，因为它仅与类型的输入一起使用 MediaConnect。SecurityGroups，在本示例中为空，因为它仅与 PUSH 输入一起使用。State这个输入。Tags，为空（此参数的默认值）。

有关更多信息，请参阅 AWS Elemental MediaLive 用户指南中的[创建输入](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateInput](#)中的。

## MediaPackage 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 MediaPackage。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### create-channel

以下代码示例演示了如何使用 create-channel。

## AWS CLI

### 创建频道

以下`create-channel`命令创建一个以当前账号命名的`sportschannel`频道。

```
aws mediapackage create-channel --id sportschannel
```

输出：

```
{
  "Arn": "arn:aws:mediapackage:us-west-2:111222333:channels/6d345804ec3f46c9b454a91d4a80d0e0",
  "HlsIngest": {
    "IngestEndpoints": [
      {
        "Id": "6d345804ec3f46c9b454a91d4a80d0e0",
        "Password": "generatedwebdavpassword1",
        "Url": "https://f31c86aed53b815a.mediapackage.us-west-2.amazonaws.com/in/v2/6d345804ec3f46c9b454a91d4a80d0e0/6d345804ec3f46c9b454a91d4a80d0e0/channel",
        "Username": "generatedwebdavusername1"
      },
      {
        "Id": "2daa32878af24803b24183727211b8ff",
        "Password": "generatedwebdavpassword2",
        "Url": "https://6ebbe7e04c4b0afa.mediapackage.us-west-2.amazonaws.com/in/v2/6d345804ec3f46c9b454a91d4a80d0e0/2daa32878af24803b24183727211b8ff/channel",
        "Username": "generatedwebdavusername2"
      }
    ]
  },
  "Id": "sportschannel",
  "Tags": {
    "region": "west"
  }
}
```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[创建频道](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateChannel](#)中的。

## create-origin-endpoint

以下代码示例演示了如何使用 create-origin-endpoint。

### AWS CLI

#### 创建源终端节点

以下create-origin-endpoint命令使用 JSON 文件中提供的软件包设置和指定的端点设置创建名为cmafsports为的源端点。

```
aws mediapackage create-origin-endpoint \  
  --channel-id sportschannel \  
  --id cmafsports \  
  --cmaf-package file:///file/path/cmafpkg.json --description "cmaf output of sports" \  
  --id cmaf_sports \  
  --manifest-name sports_channel \  
  --startover-window-seconds 300 \  
  --tags region=west,media=sports \  
  --time-delay-seconds 10
```

输出：

```
{  
  "Arn": "arn:aws:mediapackage:us-west-2:111222333:origin_endpoints/1dc6718be36f4f34bb9cd86bc50925e6",  
  "ChannelId": "sportschannel",  
  "CmafPackage": {  
    "HlsManifests": [  
      {  
        "AdMarkers": "PASSTHROUGH",  
        "Id": "cmaf_sports_endpoint",  
        "IncludeIframeOnlyStream": true,  
        "ManifestName": "index",  
        "PlaylistType": "EVENT",  
        "PlaylistWindowSeconds": 300,  
        "ProgramDateTimeIntervalSeconds": 300,  
        "Url": "https://c4af3793bf76b33c.mediapackage.us-west-2.amazonaws.com/out/v1/1dc6718be36f4f34bb9cd86bc50925e6/cmaf_sports_endpoint/index.m3u8"  
      }  
    ],  
  },  
}
```

```
    "SegmentDurationSeconds": 2,
    "SegmentPrefix": "sportschannel"
  },
  "Description": "cmaf output of sports",
  "Id": "cmaf_sports",
  "ManifestName": "sports_channel",
  "StartoverWindowSeconds": 300,
  "Tags": {
    "region": "west",
    "media": "sports"
  },
  "TimeDelaySeconds": 10,
  "Url": "",
  "Whitelist": []
}
```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[创建端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateOriginEndpoint](#)中的。

## delete-channel

以下代码示例演示了如何使用 delete-channel。

### AWS CLI

#### 删除频道

以下 delete-channel 命令删除名为的频道 test。

```
aws mediapackage delete-channel \
  --id test
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[删除频道](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteChannel](#)中的。

## delete-origin-endpoint

以下代码示例演示了如何使用 delete-origin-endpoint。



## AWS CLI

### 删除源端点

以下delete-origin-endpoint命令删除名为的源端点tester2。

```
aws mediapackage delete-origin-endpoint \  
  --id tester2
```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[删除端点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteOriginEndpoint](#)中的。

## describe-channel

以下代码示例演示了如何使用 describe-channel。

## AWS CLI

### 描述频道

以下describe-channel命令显示名为的频道的所有详细信息test。

```
aws mediapackage describe-channel \  
  --id test
```

输出：

```
{  
  "Arn": "arn:aws:mediapackage:us-  
west-2:111222333:channels/584797f1740548c389a273585dd22a63",  
  "HlsIngest": {  
    "IngestEndpoints": [  
      {  
        "Id": "584797f1740548c389a273585dd22a63",  
        "Password": "webdavgeneratedpassword1",  
        "Url": "https://9be9c4405c474882.mediapackage.us-  
west-2.amazonaws.com/in/  
v2/584797f1740548c389a273585dd22a63/584797f1740548c389a273585dd22a63/channel",  
        "Username": "webdavgeneratedusername1"  
      },  
      {
```

```

        "Id": "7d187c8616fd455f88aaa5a9fcf74442",
        "Password": "webdavgeneratedpassword2",
        "Url": "https://7bf454c57220328d.mediapackage.us-
west-2.amazonaws.com/in/
v2/584797f1740548c389a273585dd22a63/7d187c8616fd455f88aaa5a9fcf74442/channel",
        "Username": "webdavgeneratedusername2"
    }
]
},
"Id": "test",
"Tags": {}
}

```

有关更多信息，请参阅 < <https://docs.aws.amazon.com/mediapackage/latest/ug/channels-view.html> > AWS Elemental MediaPackage 用户指南中的查看频道详情

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeChannel](#) 中的。

## describe-origin-endpoint

以下代码示例演示了如何使用 describe-origin-endpoint。

### AWS CLI

#### 描述源端点

以下 describe-origin-endpoint 命令显示名为的源端点的所有详细信息 cmaf\_sports。

```
aws mediapackage describe-origin-endpoint \
  --id cmaf_sports
```

输出：

```
{
  "Arn": "arn:aws:mediapackage:us-
west-2:111222333:origin_endpoints/1dc6718be36f4f34bb9cd86bc50925e6",
  "ChannelId": "sportschannel",
  "CmafPackage": {
    "HlsManifests": [
      {
        "AdMarkers": "NONE",
        "Id": "cmaf_sports_endpoint",

```

```
        "IncludeIframeOnlyStream": false,
        "PlaylistType": "EVENT",
        "PlaylistWindowSeconds": 60,
        "ProgramDateTimeIntervalSeconds": 0,
        "Url": "https://c4af3793bf76b33c.mediapackage.us-
west-2.amazonaws.com/out/v1/1dc6718be36f4f34bb9cd86bc50925e6/cmaf_sports_endpoint/
index.m3u8"
    }
  ],
  "SegmentDurationSeconds": 2,
  "SegmentPrefix": "sportschannel"
},
"Id": "cmaf_sports",
"ManifestName": "index",
"StartoverWindowSeconds": 0,
"Tags": {
  "region": "west",
  "media": "sports"
},
"TimeDelaySeconds": 0,
"Url": "",
"Whitelist": []
}
```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[查看单个端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeOriginEndpoint](#)中的。

## list-channels

以下代码示例演示了如何使用 list-channels。

### AWS CLI

列出所有通道

以下 list-channels 命令列出了在当前 AWS 账户上配置的所有频道。

```
aws mediapackage list-channels
```

输出：

```
{
```

```

    "Channels": [
      {
        "Arn": "arn:aws:mediapackage:us-
west-2:111222333:channels/584797f1740548c389a273585dd22a63",
        "HlsIngest": {
          "IngestEndpoints": [
            {
              "Id": "584797f1740548c389a273585dd22a63",
              "Password": "webdavgeneratedpassword1",
              "Url": "https://9be9c4405c474882.mediapackage.us-
west-2.amazonaws.com/in/
v2/584797f1740548c389a273585dd22a63/584797f1740548c389a273585dd22a63/channel",
              "Username": "webdavgeneratedusername1"
            },
            {
              "Id": "7d187c8616fd455f88aaa5a9fcf74442",
              "Password": "webdavgeneratedpassword2",
              "Url": "https://7bf454c57220328d.mediapackage.us-
west-2.amazonaws.com/in/
v2/584797f1740548c389a273585dd22a63/7d187c8616fd455f88aaa5a9fcf74442/channel",
              "Username": "webdavgeneratedusername2"
            }
          ]
        },
        "Id": "test",
        "Tags": {}
      }
    ]
  }
}

```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[查看频道详情](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListChannels](#)中的。

## list-origin-endpoints

以下代码示例演示了如何使用 list-origin-endpoints。

### AWS CLI

列出通道上的所有源端点

以下 list-origin-endpoints 命令列出了名为 test 的通道中配置的所有源端点。

```
aws mediapackage list-origin-endpoints \  
  --channel-id test
```

输出：

```
{  
  "OriginEndpoints": [  
    {  
      "Arn": "arn:aws:mediapackage:us-  
west-2:111222333:origin_endpoints/247cff871f2845d3805129be22f2c0a2",  
      "ChannelId": "test",  
      "DashPackage": {  
        "ManifestLayout": "FULL",  
        "ManifestWindowSeconds": 60,  
        "MinBufferTimeSeconds": 30,  
        "MinUpdatePeriodSeconds": 15,  
        "PeriodTriggers": [],  
        "Profile": "NONE",  
        "SegmentDurationSeconds": 2,  
        "SegmentTemplateFormat": "NUMBER_WITH_TIMELINE",  
        "StreamSelection": {  
          "MaxVideoBitsPerSecond": 2147483647,  
          "MinVideoBitsPerSecond": 0,  
          "StreamOrder": "ORIGINAL"  
        },  
        "SuggestedPresentationDelaySeconds": 25  
      },  
      "Id": "tester2",  
      "ManifestName": "index",  
      "StartoverWindowSeconds": 0,  
      "Tags": {},  
      "TimeDelaySeconds": 0,  
      "Url": "https://8343f7014c0ea438.mediapackage.us-west-2.amazonaws.com/  
out/v1/247cff871f2845d3805129be22f2c0a2/index.mpd",  
      "Whitelist": []  
    },  
    {  
      "Arn": "arn:aws:mediapackage:us-  
west-2:111222333:origin_endpoints/869e237f851549e9bcf10e3bc2830839",  
      "ChannelId": "test",  
      "HlsPackage": {  
        "AdMarkers": "NONE",  
        "IncludeIframeOnlyStream": false,  
      }  
    }  
  ]  
}
```

```

        "PlaylistType": "EVENT",
        "PlaylistWindowSeconds": 60,
        "ProgramDateTimeIntervalSeconds": 0,
        "SegmentDurationSeconds": 6,
        "StreamSelection": {
            "MaxVideoBitsPerSecond": 2147483647,
            "MinVideoBitsPerSecond": 0,
            "StreamOrder": "ORIGINAL"
        },
        "UseAudioRenditionGroup": false
    },
    "Id": "tester",
    "ManifestName": "index",
    "StartoverWindowSeconds": 0,
    "Tags": {},
    "TimeDelaySeconds": 0,
    "Url": "https://8343f7014c0ea438.mediapackage.us-west-2.amazonaws.com/
out/v1/869e237f851549e9bcf10e3bc2830839/index.m3u8",
    "Whitelist": []
}
]
}

```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[查看与频道关联的所有端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListOriginEndpoints](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出分配给资源的标签

以下 `list-tags-for-resource` 命令列出了分配给指定资源的标签。

```

aws mediapackage list-tags-for-resource \
  --resource-arn arn:aws:mediapackage:us-
west-2:111222333:channels/6d345804ec3f46c9b454a91d4a80d0e0

```

输出：

```
{
  "Tags": {
    "region": "west"
  }
}
```

有关更多信息，请参阅《元素用户指南》中的“[在 AWS Elemental MediaPackage 中为 AWS 资源添加标签](#)”。MediaPackage

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## rotate-ingest-endpoint-credentials

以下代码示例演示了如何使用 rotate-ingest-endpoint-credentials。

### AWS CLI

#### 轮换采集凭证

以下 rotate-ingest-endpoint-credentials 命令轮换指定采集端点的 WebDAV 用户名和密码。

```
aws mediapackage rotate-ingest-endpoint-credentials \
  --id test \
  --ingest-endpoint-id 584797f1740548c389a273585dd22a63
```

输出：

```
{
  "Arn": "arn:aws:mediapackage:us-west-2:111222333:channels/584797f1740548c389a273585dd22a63",
  "HlsIngest": {
    "IngestEndpoints": [
      {
        "Id": "584797f1740548c389a273585dd22a63",
        "Password": "webdavregeneratedpassword1",
        "Url": "https://9be9c4405c474882.mediapackage.us-west-2.amazonaws.com/in/v2/584797f1740548c389a273585dd22a63/584797f1740548c389a273585dd22a63/channel",
        "Username": "webdavregeneratedusername1"
      },
      {
```

```
        "Id": "7d187c8616fd455f88aaa5a9fcf74442",
        "Password": "webdavgeneratedpassword2",
        "Url": "https://7bf454c57220328d.mediapackage.us-
west-2.amazonaws.com/in/
v2/584797f1740548c389a273585dd22a63/7d187c8616fd455f88aaa5a9fcf74442/channel",
        "Username": "webdavgeneratedusername2"
    }
]
},
"Id": "test",
"Tags": {}
}
```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[轮换输入 URL 上的凭证](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RotateIngestEndpointCredentials](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

要将标签添加到资源中

以下 tag-resource 命令将密 region=west 键和值对添加到指定资源。

```
aws mediapackage tag-resource \
  --resource-arn arn:aws:mediapackage:us-
west-2:111222333:channels/6d345804ec3f46c9b454a91d4a80d0e0 \
  --tags region=west
```

此命令不生成任何输出。

有关更多信息，请参阅《元素用户指南》中的“[在 AWS Elemental MediaPackage 中为 AWS 资源添加标签](#)”。MediaPackage

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。



## AWS CLI

### 从资源中移除标签

以下 `untag-resource` 命令 `region` 从指定频道中删除带有密钥的标签。

```
aws mediapackage untag-resource \  
  --resource-arn arn:aws:mediapackage:us-  
west-2:111222333:channels/6d345804ec3f46c9b454a91d4a80d0e0 \  
  --tag-keys region
```

有关更多信息，请参阅《元素用户指南》中的“[在 AWS Elemental MediaPackage 中为 AWS 资源添加标签](#)”。MediaPackage

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-channel

以下代码示例演示了如何使用 `update-channel`。

## AWS CLI

### 更新频道

以下 `update-channel` 命令更新名为的频道 `sportschannel` 以包含描述 `24x7 sports`。

```
aws mediapackage update-channel \  
  --id sportschannel \  
  --description "24x7 sports"
```

输出：

```
{  
  "Arn": "arn:aws:mediapackage:us-  
west-2:111222333:channels/6d345804ec3f46c9b454a91d4a80d0e0",  
  "Description": "24x7 sports",  
  "HlsIngest": {  
    "IngestEndpoints": [  
      {  
        "Id": "6d345804ec3f46c9b454a91d4a80d0e0",  
        "Password": "generatedwebdavpassword1",
```

```

        "Url": "https://f31c86aed53b815a.mediapackage.us-
west-2.amazonaws.com/in/
v2/6d345804ec3f46c9b454a91d4a80d0e0/6d345804ec3f46c9b454a91d4a80d0e0/channel",
        "Username": "generatedwebdavusername1"
    },
    {
        "Id": "2daa32878af24803b24183727211b8ff",
        "Password": "generatedwebdavpassword2",
        "Url": "https://6ebbe7e04c4b0afa.mediapackage.us-
west-2.amazonaws.com/in/
v2/6d345804ec3f46c9b454a91d4a80d0e0/2daa32878af24803b24183727211b8ff/channel",
        "Username": "generatedwebdavusername2"
    }
]
},
"Id": "sportschannel",
"Tags": {}
}

```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[编辑频道](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateChannel](#)中的。

## update-origin-endpoint

以下代码示例演示了如何使用 update-origin-endpoint。

### AWS CLI

#### 更新源端点

以下 update-origin-endpoint 命令更新名为的源端点 cmaf\_sports。它将延迟时间更改为 0 秒。

```

aws mediapackage update-origin-endpoint \
  --id cmaf_sports \
  --time-delay-seconds 0

```

输出：

```

{
  "Arn": "arn:aws:mediapackage:us-
west-2:111222333:origin_endpoints/1dc6718be36f4f34bb9cd86bc50925e6",

```

```
"ChannelId": "sportschannel",
"CmafPackage": {
  "HlsManifests": [
    {
      "AdMarkers": "NONE",
      "Id": "cmaf_sports_endpoint",
      "IncludeIframeOnlyStream": false,
      "PlaylistType": "EVENT",
      "PlaylistWindowSeconds": 60,
      "ProgramDateTimeIntervalSeconds": 0,
      "Url": "https://c4af3793bf76b33c.mediapackage.us-
west-2.amazonaws.com/out/v1/1dc6718be36f4f34bb9cd86bc50925e6/cmaf_sports_endpoint/
index.m3u8"
    }
  ],
  "SegmentDurationSeconds": 2,
  "SegmentPrefix": "sportschannel"
},
"Id": "cmaf_sports",
"ManifestName": "index",
"StartoverWindowSeconds": 0,
"Tags": {
  "region": "west",
  "media": "sports"
},
"TimeDelaySeconds": 0,
"Url": "",
"Whitelist": []
}
```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[编辑端点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateOriginEndpoint](#)中的。

## MediaPackage 使用的 VOD 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 MediaPackage VOD 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### create-asset

以下代码示例演示了如何使用 create-asset。

#### AWS CLI

##### 创建资产

以下create-asset示例创建了在当前 AWS 账户Chicken\_Asset中名为的资产。资产会将文件摄取30sec\_chicken.smil到。MediaPackage

```
aws mediapackage-vod create-asset \  
  --id chicken_asset \  
  --packaging-group-id hls_chicken_gp \  
  --source-role-arn arn:aws:iam::111122223333:role/EMP_Vod \  
  --source-arn arn:aws:s3::111122223333:video-bucket/A/30sec_chicken.smil
```

##### 输出：

```
{  
  "Arn": "arn:aws:mediapackage-vod:us-west-2:111122223333:assets/chicken_asset",  
  "Id": "chicken_asset",  
  "PackagingGroupId": "hls_chicken_gp",  
  "SourceArn": "arn:aws:s3::111122223333:video-bucket/A/30sec_chicken.smil",  
  "SourceRoleArn": "arn:aws:iam::111122223333:role/EMP_Vod",  
  "EgressEndpoints": [  
    {  
      "PackagingConfigurationId": "New_config_1",  
      "Url": "https://c75ea2668ab49d02bca7ae10ef31c59e.egress.mediapackage-  
vod.us-west-2.amazonaws.com/out/  
v1/6644b55df1744261ab3732a8e5cdaf07/904b06a58c7645e08d57d40d064216ac/  
f5b2e633ff4942228095d164c10074f3/index.m3u8"  
    },  
  ],  
}
```

```
    {
      "PackagingConfigurationId":"new_hls",
      "Url":" https://c75ea2668ab49d02bca7ae10ef31c59e.egress.mediapackage-
vod.us-west-2.amazonaws.com/out/v1/6644b55df1744261ab3732a8e5cdaf07/
fe8f1f00a80e424cb4f8da4095835e9e/7370ec57432343af816332356d2bd5c6/string.m3u8"
    }
  ]
}
```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[收录资产](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateAsset](#) 中的。

## create-packaging-configuration

以下代码示例演示了如何使用 create-packaging-configuration。

### AWS CLI

#### 创建打包配置

以下 create-packaging-configuration 示例在名为的打包组 new\_hls 中创建了一个名为的打包配置 hls\_chicken。此示例使用磁盘上名为的文件 hls\_pc.json 来提供详细信息。

```
aws mediapackage-vod create-packaging-configuration \
  --id new_hls \
  --packaging-group-id hls_chicken \
  --hls-package file://hls_pc.json
```

hls\_pc.json 的内容：

```
{
  "HlsManifests":[
    {
      "AdMarkers":"NONE",
      "IncludeIframeOnlyStream":false,
      "ManifestName":"string",
      "ProgramDateTimeIntervalSeconds":60,
      "RepeatExtXKey":true,
      "StreamSelection":{"
        "MaxVideoBitsPerSecond":1000,
        "MinVideoBitsPerSecond":0,
```

```

        "StreamOrder":"ORIGINAL"
      }
    }
  ],
  "SegmentDurationSeconds":6,
  "UseAudioRenditionGroup":false
}

```

输出：

```

{
  "Arn":"arn:aws:mediapackage-vod:us-west-2:111122223333:packaging-configurations/new_hls",
  "Id":"new_hls",
  "PackagingGroupId":"hls_chicken",
  "HlsManifests":{
    "SegmentDurationSeconds":6,
    "UseAudioRenditionGroup":false,
    "HlsMarkers":[
      {
        "AdMarkers":"NONE",
        "IncludeIframeOnlyStream":false,
        "ManifestName":"string",
        "ProgramDateTimeIntervalSeconds":60,
        "RepeatExtXKey":true,
        "StreamSelection":{
          "MaxVideoBitsPerSecond":1000,
          "MinVideoBitsPerSecond":0,
          "StreamOrder":"ORIGINAL"
        }
      }
    ]
  }
}

```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[创建打包配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreatePackagingConfiguration](#)中的。

## create-packaging-group

以下代码示例演示了如何使用 create-packaging-group。

## AWS CLI

### 创建打包组

以下create-packaging-group示例列出了当前 AWS 账户中配置的所有打包组。

```
aws mediapackage-vod create-packaging-group \  
  --id hls_chicken
```

输出：

```
{  
  "Arn": "arn:aws:mediapackage-vod:us-west-2:111122223333:packaging-groups/  
hls_chicken",  
  "Id": "hls_chicken"  
}
```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[创建打包组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePackagingGroup](#)中的。

## delete-asset

以下代码示例演示了如何使用 delete-asset。

### AWS CLI

#### 删除资产

以下delete-asset示例删除名为的资产30sec\_chicken。

```
aws mediapackage-vod delete-asset \  
  --id 30sec_chicken
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[删除资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAsset](#)中的。

## delete-packaging-configuration

以下代码示例演示了如何使用 delete-packaging-configuration。

## AWS CLI

### 删除打包配置

以下delete-packaging-configuration示例删除名为的打包配置CMAF。

```
aws mediapackage-vod delete-packaging-configuration \  
  --id CMAF
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[删除打包配置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePackagingConfiguration](#)中的。

## delete-packaging-group

以下代码示例演示了如何使用 delete-packaging-group。

### AWS CLI

#### 删除包装组

以下delete-packaging-group示例删除名为的打包组Dash\_widevine。

```
aws mediapackage-vod delete-packaging-group \  
  --id Dash_widevine
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[删除包装组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePackagingGroup](#)中的。

## describe-asset

以下代码示例演示了如何使用 describe-asset。

### AWS CLI

#### 描述资产



以下describe-asset示例显示名为的资产的所有详细信息30sec\_chicken。

```
aws mediapackage-vod describe-asset \  
  --id 30sec_chicken
```

输出：

```
{  
  "Arn": "arn:aws:mediapackage-vod:us-west-2:111122223333:assets/30sec_chicken",  
  "Id": "30sec_chicken",  
  "PackagingGroupId": "Packaging_group_1",  
  "SourceArn": "arn:aws:s3::111122223333:video-bucket/A/30sec_chicken.smil",  
  "SourceRoleArn": "arn:aws:iam::111122223333:role/EMP_Vod",  
  "EgressEndpoints": [  
    {  
      "PackagingConfigurationId": "DASH",  
      "Url": "https://a5f46a44118ba3e3724ef39ef532e701.egress.mediapackage-  
vod.us-west-2.amazonaws.com/out/v1/  
aad7962c569946119c2d5a691be5663c/66c25aff456d463aae0855172b3beb27/4ddfda6da17c4c279a1b8401cb  
index.mpd"  
    },  
    {  
      "PackagingConfigurationId": "HLS",  
      "Url": "https://a5f46a44118ba3e3724ef39ef532e701.egress.mediapackage-  
vod.us-west-2.amazonaws.com/out/v1/  
aad7962c569946119c2d5a691be5663c/6e5bf286a3414254a2bf0d22ae148d7e/06b5875b4d004c3cbdc4da2dc4  
index.m3u8"  
    },  
    {  
      "PackagingConfigurationId": "CMAF",  
      "Url": "https://a5f46a44118ba3e3724ef39ef532e701.egress.mediapackage-  
vod.us-west-2.amazonaws.com/out/v1/  
aad7962c569946119c2d5a691be5663c/628fb5d8d89e4702958b020af27fde0e/05eb062214064238ad6330a443  
index.m3u8"  
    }  
  ]  
}
```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[查看资产详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAsset](#)中的。

## describe-packaging-configuration

以下代码示例演示了如何使用 describe-packaging-configuration。

### AWS CLI

#### 描述打包配置

以下describe-packaging-configuration示例显示名为的打包配置的所有详细信息DASH。

```
aws mediapackage-vod describe-packaging-configuration \  
  --id DASH
```

输出：

```
{  
  "Arn": "arn:aws:mediapackage-vod:us-west-2:111122223333:packaging-configurations/  
DASH",  
  "Id": "DASH",  
  "PackagingGroupId": "Packaging_group_1",  
  "DashPackage": [  
    {  
      "SegmentDurationSeconds": "2"  
    },  
    {  
      "DashManifests": {  
        "ManifestName": "index",  
        "MinBufferTimeSeconds": "30",  
        "Profile": "NONE"  
      }  
    }  
  ]  
}
```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[查看打包配置详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePackagingConfiguration](#)中的。

## describe-packaging-group

以下代码示例演示了如何使用 describe-packaging-group。

## AWS CLI

### 描述包装组

以下describe-packaging-group示例显示名为的打包组的所有详细信息Packaging\_group\_1。

```
aws mediapackage-vod describe-packaging-group \  
  --id Packaging_group_1
```

输出：

```
{  
  "Arn": "arn:aws:mediapackage-vod:us-west-2:111122223333:packaging-groups/  
Packaging_group_1",  
  "Id": "Packaging_group_1"  
}
```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[查看包装组详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePackagingGroup](#)中的。

## list-assets

以下代码示例演示了如何使用 list-assets。

### AWS CLI

列出所有资产

以下list-assets示例列出了当前 AWS 账户中配置的所有资产。

```
aws mediapackage-vod list-assets
```

输出：

```
{  
  "Assets": [  
    "Arn": "arn:aws:mediapackage-vod:us-  
west-2:111122223333:assets/30sec_chicken",  
    "Id": "30sec_chicken",  
    "PackagingGroupId": "Packaging_group_1",
```

```
    "SourceArn": "arn:aws:s3::111122223333:video-bucket/A/30sec_chicken.smil",
    "SourceRoleArn": "arn:aws:iam::111122223333:role/EMP_Vod"
  ]
}
```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[查看资产详细信息](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAssets](#) 中的。

## list-packaging-configurations

以下代码示例演示了如何使用 list-packaging-configurations。

### AWS CLI

列出所有打包配置

以下 list-packaging-configurations 示例列出了在名为的打包组上配置的所有打包配置 Packaging\_group\_1。

```
aws mediapackage-vod list-packaging-configurations \
  --packaging-group-id Packaging_group_1
```

输出：

```
{
  "PackagingConfigurations": [
    {
      "Arn": "arn:aws:mediapackage-vod:us-west-2:111122223333:packaging-
configurations/CMAF",
      "Id": "CMAF",
      "PackagingGroupId": "Packaging_group_1",
      "CmafPackage": [
        {
          "SegmentDurationSeconds": "2"
        },
        {
          "HlsManifests": {
            "AdMarkers": "NONE",
            "RepeatExtXKey": "False",
            "ManifestName": "index",
            "ProgramDateTimeIntervalSeconds": "0",
            "IncludeIframeOnlyStream": "False"
          }
        }
      ]
    }
  ]
}
```

```

    }
  }
]
},
{
  "Arn":"arn:aws:mediapackage-vod:us-west-2:111122223333:packaging-
configurations/DASH",
  "Id":"DASH",
  "PackagingGroupId":"Packaging_group_1",
  "DashPackage":[
    {
      "SegmentDurationSeconds":"2"
    },
    {
      "DashManifests":{
        "ManifestName":"index",
        "MinBufferTimeSeconds":"30",
        "Profile":"NONE"
      }
    }
  ]
},
{
  "Arn":"arn:aws:mediapackage-vod:us-west-2:111122223333:packaging-
configurations/HLS",
  "Id":"HLS",
  "PackagingGroupId":"Packaging_group_1",
  "HlsPackage":[
    {
      "SegmentDurationSeconds":"6",
      "UseAudioRenditionGroup":"False"
    },
    {
      "HlsManifests":{
        "AdMarkers":"NONE",
        "RepeatExtXKey":"False",
        "ManifestName":"index",
        "ProgramDateTimeIntervalSeconds":"0",
        "IncludeIframeOnlyStream":"False"
      }
    }
  ]
},
{

```

```

    "Arn": "arn:aws:mediapackage-vod:us-west-2:111122223333:packaging-
configurations/New_config_0_copy",
    "Id": "New_config_0_copy",
    "PackagingGroupId": "Packaging_group_1",
    "HlsPackage": [
      {
        "SegmentDurationSeconds": "6",
        "UseAudioRenditionGroup": "False"
      },
      {
        "Encryption": {
          "EncryptionMethod": "AWS_128",
          "SpekeKeyProvider": {
            "RoleArn": "arn:aws:iam:111122223333::role/SPEKERole",
            "Url": "https://lfgubdvs97.execute-api.us-
west-2.amazonaws.com/EkeStage/copyProtection/",
            "SystemIds": [
              "81376844-f976-481e-a84e-cc25d39b0b33"
            ]
          }
        }
      },
      {
        "HlsManifests": {
          "AdMarkers": "NONE",
          "RepeatExtXKey": "False",
          "ManifestName": "index",
          "ProgramDateTimeIntervalSeconds": "0",
          "IncludeIframeOnlyStream": "False"
        }
      }
    ]
  }
]
}

```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[查看打包配置详细信息](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListPackagingConfigurations](#)中的。

## list-packaging-groups

以下代码示例演示了如何使用 list-packaging-groups。

## AWS CLI

### 列出所有包装组

以下`list-packaging-groups`示例列出了当前 AWS 账户中配置的所有打包组。

```
aws mediapackage-vod list-packaging-groups
```

输出：

```
{
  "PackagingGroups": [
    {
      "Arn": "arn:aws:mediapackage-vod:us-west-2:111122223333:packaging-
groups/Dash_widevine",
      "Id": "Dash_widevine"
    },
    {
      "Arn": "arn:aws:mediapackage-vod:us-west-2:111122223333:packaging-
groups/Encrypted_HLS",
      "Id": "Encrypted_HLS"
    },
    {
      "Arn": "arn:aws:mediapackage-vod:us-west-2:111122223333:packaging-
groups/Packaging_group_1",
      "Id": "Packaging_group_1"
    }
  ]
}
```

有关更多信息，请参阅 AWS Elemental MediaPackage 用户指南中的[查看包装组详细信息](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListPackagingGroups](#)中的。

## MediaStore 使用数据平面示例 AWS CLI

以下代码示例向您展示了如何使用 with D MediaStore ata Plane 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## delete-object

以下代码示例演示了如何使用 delete-object。

AWS CLI

删除对象

以下delete-object示例删除了指定的对象。

```
aws mediastore-data delete-object \  
  --endpoint=https://aaabbbcccddee.data.mediastore.us-west-2.amazonaws.com \  
  --path=/folder_name/README.md
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[删除对象](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteObject](#)中的。

## describe-object

以下代码示例演示了如何使用 describe-object。

AWS CLI

查看对象的标题

以下describe-object示例显示了指定路径上某个对象的标题。

```
aws mediastore-data describe-object \  
  --path=/folder_name/README.md
```





以下`get-object`示例下载对象的指定部分。

```
aws mediastore-data get-object \  
  --endpoint https://aaabbbcccddee.data.mediastore.us-west-2.amazonaws.com \  
  --path events/baseball/setup.jpg setup.jpg \  
  --range "bytes=0-100"
```

输出：

```
{  
  "StatusCode": 206,  
  "LastModified": "Fri, 19 Jul 2019 21:50:31 GMT",  
  "ContentType": "image/jpeg",  
  "ContentRange": "bytes 0-100/3860266",  
  "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3",  
  "ContentLength": "101"  
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[下载对象](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetObject](#)中的。

## list-items

以下代码示例演示了如何使用 `list-items`。

### AWS CLI

示例 1：查看存储在容器中的项目（对象和文件夹）列表

以下`list-items`示例显示了存储在指定容器中的项目（对象和文件夹）的列表。

```
aws mediastore-data list-items \  
  --endpoint https://aaabbbcccddee.data.mediastore.us-west-2.amazonaws.com
```

输出：

```
{  
  "Items": [  
    {
```

```
    "Type": "OBJECT",
    "ContentLength": 3784,
    "Name": "setup.jpg",
    "ETag":
"2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f55555555555555da6d3",
    "ContentType": "image/jpeg",
    "LastModified": 1563571859.379
  },
  {
    "Type": "FOLDER",
    "Name": "events"
  }
]
}
```

示例 2：查看存储在文件夹中的项目（对象和文件夹）列表

以下 `list-items` 示例显示了存储在指定文件夹中的项目（对象和文件夹）的列表。

```
aws mediastore-data list-items \
  --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
  --path events/baseball
```

输出：

```
{
  "Items": [
    {
      "ETag":
"2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f55555555555555da6d3",
      "ContentType": "image/jpeg",
      "Type": "OBJECT",
      "ContentLength": 3860266,
      "LastModified": 1563573031.872,
      "Name": "setup.jpg"
    }
  ]
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[查看对象列表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListItems](#) 中的。

## put-object

以下代码示例演示了如何使用 put-object。

### AWS CLI

#### 示例 1：将对象上传到容器

以下put-object示例将对象上传到指定容器。

```
aws mediastore-data put-object \  
  --endpoint https://aaabbbccdddee.data.mediastore.us-west-2.amazonaws.com \  
  --body README.md \  
  --path README.md \  
  --cache-control "max-age=6, public" \  
  --content-type binary/octet-stream
```

输出：

```
{  
  "ContentSHA256":  
    "f29bc64a9d3732b4b9035125fdb3285f5b6455778edca72414671e0ca3b2e0de",  
  "StorageClass": "TEMPORAL",  
  "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9e00000004dd89ff7f555555555555da6d3"  
}
```

#### 示例 2：将对象上传到容器内的文件夹

以下put-object示例将对象上传到容器内的指定文件夹。

```
aws mediastore-data put-object \  
  --endpoint https://aaabbbccdddee.data.mediastore.us-west-2.amazonaws.com \  
  --body README.md \  
  --path /september-events/README.md \  
  --cache-control "max-age=6, public" \  
  --content-type binary/octet-stream
```

输出：

```
{  
  "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9e00000004dd89ff7f555555555555da6d3",  
  "ContentSHA256":  
    "f29bc64a9d3732b4b9035125fdb3285f5b6455778edca72414671e0ca3b2e0de",
```

```
"StorageClass": "TEMPORAL"  
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的[上传对象](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutObject](#) 中的。

## MediaTailor 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 MediaTailor。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **delete-playback-configuration**

以下代码示例演示了如何使用 `delete-playback-configuration`。

AWS CLI

删除配置

以下内容 `delete-playback-configuration` 删除名为的配置 `campaign_short`。

```
aws mediatailor delete-playback-configuration \  
  --name campaign_short
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 基本 MediaTailor 用户指南》中的[删除配置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePlaybackConfiguration](#)中的。

## get-playback-configuration

以下代码示例演示了如何使用 get-playback-configuration。

### AWS CLI

#### 描述配置

下面get-playback-configuration显示了名为的配置的所有详细信息west\_campaign。

```
aws mediatailor get-playback-configuration \  
  --name west_campaign
```

输出：

```
{  
  "AdDecisionServerUrl": "http://your.ads.url",  
  "CdnConfiguration": {},  
  "DashConfiguration": {  
    "ManifestEndpointPrefix":  
      "https://170c14299689462897d0cc45fc2000bb.mediatailor.us-west-2.amazonaws.com/v1/  
dash/1cbfeaaecb69778e0c167d0505a2bc57da2b1754/west_campaign/",  
    "MpdLocation": "EMT_DEFAULT",  
    "OriginManifestType": "MULTI_PERIOD"  
  },  
  "HlsConfiguration": {  
    "ManifestEndpointPrefix":  
      "https://170c14299689462897d0cc45fc2000bb.mediatailor.us-west-2.amazonaws.com/v1/  
master/1cbfeaaecb69778e0c167d0505a2bc57da2b1754/west_campaign/"  
  },  
  "Name": "west_campaign",  
  "PlaybackConfigurationArn": "arn:aws:mediatailor:us-  
west-2:123456789012:playbackConfiguration/west_campaign",  
  "PlaybackEndpointPrefix":  
    "https://170c14299689462897d0cc45fc2000bb.mediatailor.us-west-2.amazonaws.com",  
  "SessionInitializationEndpointPrefix":  
    "https://170c14299689462897d0cc45fc2000bb.mediatailor.us-west-2.amazonaws.com/v1/  
session/1cbfeaaecb69778e0c167d0505a2bc57da2b1754/west_campaign/",  
  "Tags": {},  
  "VideoContentSourceUrl": "https://8343f7014c0ea438.mediapackage.us-  
west-2.amazonaws.com/out/v1/683f0f2ff7cd43a48902e6dcd5e16dcf/index.m3u8"
```

```
}
```

有关更多信息，请参阅 AWS Elemental MediaTailor 用户指南中的[查看配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetPlaybackConfiguration](#)中的。

## list-playback-configurations

以下代码示例演示了如何使用 list-playback-configurations。

### AWS CLI

列出所有配置

下图 list-playback-configurations 显示了当前 AWS 账户的配置的所有详细信息。

```
aws mediatailor list-playback-configurations
```

输出：

```
{
  "Items": [
    {
      "AdDecisionServerUrl": "http://your.ads.url",
      "CdnConfiguration": {},
      "DashConfiguration": {
        "ManifestEndpointPrefix":
"https://170c14299689462897d0cc45fc2000bb.mediatailor.us-west-2.amazonaws.com/v1/dash/1cbfeaaecb69778e0c167d0505a2bc57da2b1754/west_campaign/",
        "MpdLocation": "EMT_DEFAULT",
        "OriginManifestType": "MULTI_PERIOD"
      },
      "HlsConfiguration": {
        "ManifestEndpointPrefix":
"https://170c14299689462897d0cc45fc2000bb.mediatailor.us-west-2.amazonaws.com/v1/master/1cbfeaaecb69778e0c167d0505a2bc57da2b1754/west_campaign/"
      },
      "Name": "west_campaign",
      "PlaybackConfigurationArn": "arn:aws:mediatailor:us-west-2:123456789012:playbackConfiguration/west_campaign",
      "PlaybackEndpointPrefix":
"https://170c14299689462897d0cc45fc2000bb.mediatailor.us-west-2.amazonaws.com",
    }
  ]
}
```

```

    "SessionInitializationEndpointPrefix":
    "https://170c14299689462897d0cc45fc2000bb.mediatailor.us-west-2.amazonaws.com/v1/
    session/1cbfeaaecb69778e0c167d0505a2bc57da2b1754/west_campaign/",
    "Tags": {},
    "VideoContentSourceUrl": "https://8343f7014c0ea438.mediapackage.us-
    west-2.amazonaws.com/out/v1/683f0f2ff7cd43a48902e6dcd5e16dcf/index.m3u8"
  },
  {
    "AdDecisionServerUrl": "http://your.ads.url",
    "CdnConfiguration": {},
    "DashConfiguration": {
      "ManifestEndpointPrefix":
      "https://73511f91d6a24ca2b93f3cf1d7cedd67.mediatailor.us-west-2.amazonaws.com/v1/
      dash/1cbfeaaecb69778e0c167d0505a2bc57da2b1754/sports_campaign/",
      "MpdLocation": "DISABLED",
      "OriginManifestType": "MULTI_PERIOD"
    },
    "HlsConfiguration": {
      "ManifestEndpointPrefix":
      "https://73511f91d6a24ca2b93f3cf1d7cedd67.mediatailor.us-west-2.amazonaws.com/v1/
      master/1cbfeaaecb69778e0c167d0505a2bc57da2b1754/sports_campaign/"
    },
    "Name": "sports_campaign",
    "PlaybackConfigurationArn": "arn:aws:mediatailor:us-
    west-2:123456789012:playbackConfiguration/sports_campaign",
    "PlaybackEndpointPrefix":
    "https://73511f91d6a24ca2b93f3cf1d7cedd67.mediatailor.us-west-2.amazonaws.com",
    "SessionInitializationEndpointPrefix":
    "https://73511f91d6a24ca2b93f3cf1d7cedd67.mediatailor.us-west-2.amazonaws.com/v1/
    session/1cbfeaaecb69778e0c167d0505a2bc57da2b1754/sports_campaign/",
    "SlateAdUrl": "http://s3.bucket/slate_ad.mp4",
    "Tags": {},
    "VideoContentSourceUrl": "https://c4af3793bf76b33c.mediapackage.us-
    west-2.amazonaws.com/out/v1/1dc6718be36f4f34bb9cd86bc50925e6/sports_endpoint/
    index.m3u8"
  }
]
}

```

有关更多信息，请参阅 < <https://docs.aws.amazon.com/mediatailor/latest/ug/configurations-view.html> > AWS Elemental MediaTailor 用户指南中的查看配置。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListPlaybackConfigurations](#) 中的。



## put-playback-configuration

以下代码示例演示了如何使用 `put-playback-configuration`。

### AWS CLI

#### 创建配置

以下内容 `put-playback-configuration` 创建了一个名为的配置 `campaign_short`。

```
aws mediatailor put-playback-configuration \  
  --name campaign_short \  
  --ad-decision-server-url http://your.ads.url \  
  --video-content-source-url http://video.bucket/index.m3u8
```

#### 输出：

```
{  
  "AdDecisionServerUrl": "http://your.ads.url",  
  "CdnConfiguration": {},  
  "DashConfiguration": {  
    "ManifestEndpointPrefix":  
      "https://13484114d38f4383bc0d6a7cb879bd00.mediatailor.us-west-2.amazonaws.com/v1/  
dash/1cbfeaaecb69778e0c167d0505a2bc57da2b1754/campaign_short/",  
    "MpdLocation": "EMT_DEFAULT",  
    "OriginManifestType": "MULTI_PERIOD"  
  },  
  "HlsConfiguration": {  
    "ManifestEndpointPrefix":  
      "https://13484114d38f4383bc0d6a7cb879bd00.mediatailor.us-west-2.amazonaws.com/v1/  
master/1cbfeaaecb69778e0c167d0505a2bc57da2b1754/campaign_short/"  
  },  
  "Name": "campaign_short",  
  "PlaybackConfigurationArn": "arn:aws:mediatailor:us-  
west-2:123456789012:playbackConfiguration/campaign_short",  
  "PlaybackEndpointPrefix":  
    "https://13484114d38f4383bc0d6a7cb879bd00.mediatailor.us-west-2.amazonaws.com",  
  "SessionInitializationEndpointPrefix":  
    "https://13484114d38f4383bc0d6a7cb879bd00.mediatailor.us-west-2.amazonaws.com/v1/  
session/1cbfeaaecb69778e0c167d0505a2bc57da2b1754/campaign_short/",  
  "Tags": {},  
  "VideoContentSourceUrl": "http://video.bucket/index.m3u8"  
}
```

有关更多信息，请参阅 AWS Elemental MediaTailor 用户指南中的[创建配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[PutPlaybackConfiguration](#)中的。

## 使用的 MemoryDB 示例 AWS CLI

以下代码示例向您展示了如何使用 with MemoryDB 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### copy-snapshot

以下代码示例演示了如何使用 copy-snapshot。

#### AWS CLI

##### 复制快照

以下 copy-snapshot 示例创建了快照的副本。

```
aws memorydb copy-snapshot \  
  --source-snapshot-name my-cluster-snapshot \  
  --target-snapshot-name my-cluster-snapshot-copy
```

##### 输出

```
{  
  "Snapshot": {  
    "Name": "my-cluster-snapshot-copy",
```

```

    "Status": "creating",
    "Source": "manual",
    "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:snapshot/my-cluster-
snapshot-copy",
    "ClusterConfiguration": {
      "Name": "my-cluster",
      "Description": " ",
      "NodeType": "db.r6g.large",
      "EngineVersion": "6.2",
      "MaintenanceWindow": "wed:03:00-wed:04:00",
      "Port": 6379,
      "ParameterGroupName": "default.memorydb-redis6",
      "SubnetGroupName": "my-sg",
      "VpcId": "vpc-xx2574fc",
      "SnapshotRetentionLimit": 0,
      "SnapshotWindow": "04:30-05:30",
      "NumShards": 2
    }
  }
}

```

有关更多信息，请参阅《MemoryDB 用户指南》中的[复制快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CopySnapshot](#)中的。

## create-acl

以下代码示例演示了如何使用 create-acl。

### AWS CLI

#### 创建 ACL

以下create-acl示例创建了一个新的访问控制列表。

```

aws memorydb create-acl \
  --acl-name "new-acl-1" \
  --user-names "my-user"

```

输出：

```
{
```

```
"ACL": {
  "Name": "new-acl-1",
  "Status": "creating",
  "UserNames": [
    "my-user"
  ],
  "MinimumEngineVersion": "6.2",
  "Clusters": [],
  "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:acl/new-acl-1"
}
```

有关更多信息，请参阅《MemoryDB [用户指南](#)》中的[使用访问控制列表对用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAcl](#)中的。

## create-cluster

以下代码示例演示了如何使用 create-cluster。

### AWS CLI

#### 创建集群

以下create-cluster示例创建了一个新集群。

```
aws memorydb create-cluster \
  --cluster-name my-new-cluster \
  --node-type db.r6g.large \
  --acl-name my-acl \
  --subnet-group my-sg
```

输出：

```
{
  "Cluster": {
    "Name": "my-new-cluster",
    "Status": "creating",
    "NumberOfShards": 1,
    "AvailabilityMode": "MultiAZ",
    "ClusterEndpoint": {
      "Port": 6379
    }
  }
}
```

```
    },
    "NodeType": "db.r6g.large",
    "EngineVersion": "6.2",
    "EnginePatchVersion": "6.2.6",
    "ParameterGroupName": "default.memorydb-redis6",
    "ParameterGroupStatus": "in-sync",
    "SubnetGroupName": "my-sg",
    "TLSEnabled": true,
    "ARN": "arn:aws:memorydb:us-east-1:49165xxxxxx:cluster/my-new-cluster",
    "SnapshotRetentionLimit": 0,
    "MaintenanceWindow": "sat:10:00-sat:11:00",
    "SnapshotWindow": "07:30-08:30",
    "ACLName": "my-acl",
    "AutoMinorVersionUpgrade": true
  }
}
```

有关更多信息，请参阅 MemoryDB 用户指南中的[管理集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateCluster](#) 中的。

## create-parameter-group

以下代码示例演示了如何使用 create-parameter-group。

### AWS CLI

#### 创建参数组

以下 create-parameter-group 示例创建了一个参数组。

```
aws memorydb create-parameter-group \
  --parameter-group-name myRedis6x \
  --family memorydb_redis6 \
  --description "my-parameter-group"
```

输出：

```
{
  "ParameterGroup": {
    "Name": "myredis6x",
    "Family": "memorydb_redis6",
```

```
    "Description": "my-parameter-group",
    "ARN": "arn:aws:memorydb:us-east-1:49165xxxxxx:parametergroup/myredis6x"
  }
}
```

有关更多信息，请参阅《MemoryDB 用户指南》中的[创建参数组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateParameterGroup](#)中的。

## create-snapshot

以下代码示例演示了如何使用 create-snapshot。

### AWS CLI

#### 创建快照

以下create-snapshot示例创建快照。

```
aws memorydb create-snapshot \
  --cluster-name my-cluster \
  --snapshot-name my-cluster-snapshot
```

输出：

```
{
  "Snapshot": {
    "Name": "my-cluster-snapshot1",
    "Status": "creating",
    "Source": "manual",
    "ARN": "arn:aws:memorydb:us-east-1:49165xxxxxx:snapshot/my-cluster-snapshot",
    "ClusterConfiguration": {
      "Name": "my-cluster",
      "Description": "",
      "NodeType": "db.r6g.large",
      "EngineVersion": "6.2",
      "MaintenanceWindow": "wed:03:00-wed:04:00",
      "Port": 6379,
      "ParameterGroupName": "default.memorydb-redis6",
      "SubnetGroupName": "my-sg",
      "VpcId": "vpc-862xxxxc",
    }
  }
}
```

```

        "SnapshotRetentionLimit": 0,
        "SnapshotWindow": "04:30-05:30",
        "NumShards": 2
    }
}
}

```

有关更多信息，请参阅《MemoryDB 用户指南》中的[制作手动快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSnapshot](#)中的。

## create-subnet-group

以下代码示例演示了如何使用 create-subnet-group。

### AWS CLI

#### 创建子网组

以下create-subnet-group示例创建了一个子网组。

```

aws memorydb create-subnet-group \
  --subnet-group-name mysubnetgroup \
  --description "my subnet group" \
  --subnet-ids subnet-5623xxxx

```

#### 输出：

```

{
  "SubnetGroup": {
    "Name": "mysubnetgroup",
    "Description": "my subnet group",
    "VpcId": "vpc-86257xxx",
    "Subnets": [
      {
        "Identifier": "subnet-5623xxxx",
        "AvailabilityZone": {
          "Name": "us-east-1a"
        }
      }
    ],
    "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:subnetgroup/mysubnetgroup"
  }
}

```

```
}  
}
```

有关更多信息，请参阅《MemoryDB 用户指南》中的[创建子网组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateSubnetGroup](#) 中的。

## create-user

以下代码示例演示了如何使用 create-user。

### AWS CLI

#### 创建用户

以下 create-user 示例创建了一个新用户。

```
aws memorydb create-user \  
  --user-name user-name-1 \  
  --access-string "~objects:* ~items:* ~public:*" \  
  --authentication-mode \  
    Passwords="enterapasswordhere",Type=password
```

输出：

```
{  
  "User": {  
    "Name": "user-name-1",  
    "Status": "active",  
    "AccessString": "off ~objects:* ~items:* ~public:* resetchannels -@all",  
    "ACLNames": [],  
    "MinimumEngineVersion": "6.2",  
    "Authentication": {  
      "Type": "password",  
      "PasswordCount": 1  
    },  
    "ARN": "arn:aws:memorydb:us-west-2:491658xxxxxx:user/user-name-1"  
  }  
}
```

有关更多信息，请参阅《MemoryDB [用户指南](#)》中的[使用访问控制列表对用户进行身份验证](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUser](#)中的。

## delete-acl

以下代码示例演示了如何使用 delete-acl。

### AWS CLI

#### 删除 ACL

以下delete-acl示例删除访问控制列表。

```
aws memorydb delete-acl \  
  --acl-name "new-acl-1"
```

输出：

```
{  
  "ACL": {  
    "Name": "new-acl-1",  
    "Status": "deleting",  
    "UserNames": [  
      "pat"  
    ],  
    "MinimumEngineVersion": "6.2",  
    "Clusters": [],  
    "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:acl/new-acl-1"  
  }  
}
```

有关更多信息，请参阅《MemoryDB [用户指南](#)》中的[使用访问控制列表对用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAcl](#)中的。

## delete-cluster

以下代码示例演示了如何使用 delete-cluster。

### AWS CLI

#### 删除集群

以下delete-cluster示例删除集群。

```
aws memorydb delete-cluster \  
  --cluster-name my-new-cluster
```

输出：

```
{  
  "Cluster": {  
    "Name": "my-new-cluster",  
    "Status": "deleting",  
    "NumberOfShards": 1,  
    "ClusterEndpoint": {  
      "Address": "clustercfg.my-new-cluster.xxxxx.memorydb.us-  
east-1.amazonaws.com",  
      "Port": 6379  
    },  
    "NodeType": "db.r6g.large",  
    "EngineVersion": "6.2",  
    "EnginePatchVersion": "6.2.6",  
    "ParameterGroupName": "default.memorydb-redis6",  
    "ParameterGroupStatus": "in-sync",  
    "SubnetGroupName": "my-sg",  
    "TLSEnabled": true,  
    "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:cluster/my-new-cluster",  
    "SnapshotRetentionLimit": 0,  
    "MaintenanceWindow": "sat:10:00-sat:11:00",  
    "SnapshotWindow": "07:30-08:30",  
    "AutoMinorVersionUpgrade": true  
  }  
}
```

有关更多信息，请参阅《MemoryDB 用户指南》中的[删除集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCluster](#)中的。

## delete-parameter-group

以下代码示例演示了如何使用 delete-parameter-group。

### AWS CLI

#### 删除参数组

以下delete-parameter-group示例删除参数组。

```
aws memorydb delete-parameter-group \  
  --parameter-group-name myRedis6x
```

输出：

```
{  
  "ParameterGroup": {  
    "Name": "myredis6x",  
    "Family": "memorydb_redis6",  
    "Description": "my-parameter-group",  
    "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:parametergroup/myredis6x"  
  }  
}
```

有关更多信息，请参阅《MemoryDB 用户指南》中的[删除参数组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteParameterGroup](#)中的。

## delete-snapshot

以下代码示例演示了如何使用 delete-snapshot。

### AWS CLI

#### 删除快照

以下delete-snapshot示例删除快照。

```
aws memorydb delete-snapshot \  
  --snapshot-name my-cluster-snapshot
```

输出：

```
{  
  "Snapshot": {  
    "Name": "my-cluster-snapshot",  
    "Status": "deleting",  
    "Source": "manual",
```

```

    "ARN": "arn:aws:memorydb:us-east-1:49165xxxxxx:snapshot/my-cluster-
snapshot",
    "ClusterConfiguration": {
      "Name": "my-cluster",
      "Description": "",
      "NodeType": "db.r6g.large",
      "EngineVersion": "6.2",
      "MaintenanceWindow": "wed:03:00-wed:04:00",
      "Port": 6379,
      "ParameterGroupName": "default.memorydb-redis6",
      "SubnetGroupName": "my-sg",
      "VpcId": "vpc-862xxxxc",
      "SnapshotRetentionLimit": 0,
      "SnapshotWindow": "04:30-05:30",
      "NumShards": 2
    }
  }
}

```

有关更多信息，请参阅《MemoryDB 用户指南》中的[删除快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSnapshot](#)中的。

## delete-subnet-group

以下代码示例演示了如何使用 delete-subnet-group。

### AWS CLI

#### 删除子网组

以下delete-subnet-group示例删除子网。

```
aws memorydb delete-subnet-group \
  --subnet-group-name mysubnetgroup
```

输出：

```
{
  "SubnetGroup": {
    "Name": "mysubnetgroup",
    "Description": "my subnet group",

```

```

    "VpcId": "vpc-86xxxx4fc",
    "Subnets": [
      {
        "Identifier": "subnet-56xxx61b",
        "AvailabilityZone": {
          "Name": "us-east-1a"
        }
      }
    ],
    "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:subnetgroup/mysubnetgroup"
  }
}

```

有关更多信息，请参阅《MemoryDB 用户指南》中的[删除子网组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSubnetGroup](#)中的。

## delete-user

以下代码示例演示了如何使用 delete-user。

### AWS CLI

#### 删除用户

以下delete-user示例删除用户。

```
aws memorydb delete-user \
  --user-name my-user
```

#### 输出：

```

{
  "User": {
    "Name": "my-user",
    "Status": "deleting",
    "AccessString": "on ~app:* resetchannels -@all +@read",
    "ACLNames": [
      "my-acl"
    ],
    "MinimumEngineVersion": "6.2",
    "Authentication": {

```

```
        "Type": "password",
        "PasswordCount": 1
    },
    "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:user/my-user"
}
}
```

有关更多信息，请参阅《MemoryDB [用户指南](#)》中的[使用访问控制列表对用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUser](#)中的。

## describe-acls

以下代码示例演示了如何使用 describe-acls。

### AWS CLI

返回 ACL 列表

以下 describe-acls 返回 ACL 列表。

```
aws memorydb describe-acls
```

输出：

```
{
  "ACLs": [
    {
      "Name": "open-access",
      "Status": "active",
      "UserNames": [
        "default"
      ],
      "MinimumEngineVersion": "6.2",
      "Clusters": [],
      "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:acl/open-access"
    },
    {
      "Name": "my-acl",
      "Status": "active",
      "UserNames": [],
      "MinimumEngineVersion": "6.2",
```

```
    "Clusters": [
      "my-cluster"
    ],
    "ARN": "arn:aws:memorydb:us-east-1:49165xxxxxxx:acl/my-acl"
  }
]
```

有关更多信息，请参阅《MemoryDB [用户指南](#)》中的[使用访问控制列表对用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAcls](#)中的。

## describe-clusters

以下代码示例演示了如何使用 describe-clusters。

### AWS CLI

返回集群列表

以下 describe-clusters 返回集群列表。

```
aws memorydb describe-clusters
```

输出：

```
{
  "Clusters": [
    {
      "Name": "my-cluster",
      "Status": "available",
      "NumberOfShards": 2,
      "ClusterEndpoint": {
        "Address": "clustercfg.my-cluster.1lru6f.memorydb.us-
east-1.amazonaws.com",
        "Port": 6379
      },
      "NodeType": "db.r6g.large",
      "EngineVersion": "6.2",
      "EnginePatchVersion": "6.2.6",
      "ParameterGroupName": "default.memorydb-redis6",
      "ParameterGroupStatus": "in-sync",
    }
  ]
}
```

```
    "SecurityGroups": [
      {
        "SecurityGroupId": "sg-0a1434xxxxxc9fae",
        "Status": "active"
      }
    ],
    "SubnetGroupName": "pat-sg",
    "TLSEnabled": true,
    "ARN": "arn:aws:memorydb:us-east-1:49165xxxxxx:cluster/my-cluster",
    "SnapshotRetentionLimit": 0,
    "MaintenanceWindow": "wed:03:00-wed:04:00",
    "SnapshotWindow": "04:30-05:30",
    "ACLName": "my-acl",
    "AutoMinorVersionUpgrade": true
  }
]
```

有关更多信息，请参阅《MemoryDB 用户指南》中的[管理集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeClusters](#) 中的。

## describe-engine-versions

以下代码示例演示了如何使用 describe-engine-versions。

### AWS CLI

返回引擎版本列表

以下 describe-engine-versions 返回引擎版本列表。

```
aws memorydb describe-engine-versions
```

输出：

```
{
  "EngineVersions": [
    {
      "EngineVersion": "6.2",
      "EnginePatchVersion": "6.2.6",
      "ParameterGroupFamily": "memorydb_redis6"
    }
  ]
}
```



```
]
}
```

有关更多信息，请参阅《MemoryDB 用户指南》中的[引擎版本和升级](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEngineVersions](#)中的。

## describe-events

以下代码示例演示了如何使用 describe-events。

### AWS CLI

返回事件列表

以下 describe-events 返回事件列表。

```
aws memorydb describe-events
```

输出：

```
{
  "Events": [
    {
      "SourceName": "my-cluster",
      "SourceType": "cluster",
      "Message": "Increase replica count started for replication group my-cluster on 2022-07-22T14:09:01.440Z",
      "Date": "2022-07-22T07:09:01.443000-07:00"
    },
    {
      "SourceName": "my-user",
      "SourceType": "user",
      "Message": "Create user my-user operation completed.",
      "Date": "2022-07-22T07:00:02.975000-07:00"
    }
  ]
}
```

有关更多信息，请参阅《MemoryDB 用户指南》中的[监控事件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEvents](#)中的。

## describe-parameter-groups

以下代码示例演示了如何使用 describe-parameter-groups。

### AWS CLI

返回参数组列表

以下 describe-parameter-groups 返回参数组列表。

```
aws memorydb describe-parameter-groups
```

输出：

```
{
  "ParameterGroups": [
    {
      "Name": "default.memorydb-redis6",
      "Family": "memorydb_redis6",
      "Description": "Default parameter group for memorydb_redis6",
      "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:parametergroup/default.memorydb-redis6"
    }
  ]
}
```

有关更多信息，请参阅《MemoryDB 用户指南》中的[使用参数组配置引擎参数](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeParameterGroups](#) 中的。

## describe-parameters

以下代码示例演示了如何使用 describe-parameters。

### AWS CLI

返回参数列表

以下描述参数返回参数列表。

```
aws memorydb describe-parameters
```

输出：

```
{
  "Parameters": [
    {
      "Name": "acllog-max-len",
      "Value": "128",
      "Description": "The maximum length of the ACL Log",
      "DataType": "integer",
      "AllowedValues": "1-10000",
      "MinimumEngineVersion": "6.2.4"
    },
    {
      "Name": "activedefrag",
      "Value": "no",
      "Description": "Enabled active memory defragmentation",
      "DataType": "string",
      "AllowedValues": "yes,no",
      "MinimumEngineVersion": "6.2.4"
    },
    {
      "Name": "active-defrag-cycle-max",
      "Value": "75",
      "Description": "Maximal effort for defrag in CPU percentage",
      "DataType": "integer",
      "AllowedValues": "1-75",
      "MinimumEngineVersion": "6.2.4"
    },
    {
      "Name": "active-defrag-cycle-min",
      "Value": "5",
      "Description": "Minimal effort for defrag in CPU percentage",
      "DataType": "integer",
      "AllowedValues": "1-75",
      "MinimumEngineVersion": "6.2.4"
    },
    {
      "Name": "active-defrag-ignore-bytes",
      "Value": "104857600",
      "Description": "Minimum amount of fragmentation waste to start active defrag",
      "DataType": "integer",
      "AllowedValues": "1048576-",
      "MinimumEngineVersion": "6.2.4"
    }
  ]
}
```

```
    },
    {
      "Name": "active-defrag-max-scan-fields",
      "Value": "1000",
      "Description": "Maximum number of set/hash/zset/list fields that will be
processed from the main dictionary scan",
      "DataType": "integer",
      "AllowedValues": "1-1000000",
      "MinimumEngineVersion": "6.2.4"
    },
    {
      "Name": "active-defrag-threshold-lower",
      "Value": "10",
      "Description": "Minimum percentage of fragmentation to start active
defrag",
      "DataType": "integer",
      "AllowedValues": "1-100",
      "MinimumEngineVersion": "6.2.4"
    },
    {
      "Name": "active-defrag-threshold-upper",
      "Value": "100",
      "Description": "Maximum percentage of fragmentation at which we use
maximum effort",
      "DataType": "integer",
      "AllowedValues": "1-100",
      "MinimumEngineVersion": "6.2.4"
    },
    {
      "Name": "active-expire-effort",
      "Value": "1",
      "Description": "The amount of effort that redis uses to expire items in
the active expiration job",
      "DataType": "integer",
      "AllowedValues": "1-10",
      "MinimumEngineVersion": "6.2.4"
    },
    {
      "Name": "activeresharding",
      "Value": "yes",
      "Description": "Apply rehashing or not",
      "DataType": "string",
      "AllowedValues": "yes,no",
      "MinimumEngineVersion": "6.2.4"
    }
  ],
  {
    "Name": "activeresharding",
    "Value": "yes",
    "Description": "Apply rehashing or not",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "MinimumEngineVersion": "6.2.4"
  }
],
{
  "Name": "activeresharding",
  "Value": "yes",
  "Description": "Apply rehashing or not",
  "DataType": "string",
  "AllowedValues": "yes,no",
  "MinimumEngineVersion": "6.2.4"
}
```

```
  },
  {
    "Name": "client-output-buffer-limit-normal-hard-limit",
    "Value": "0",
    "Description": "Normal client output buffer hard limit in bytes",
    "DataType": "integer",
    "AllowedValues": "0-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "client-output-buffer-limit-normal-soft-limit",
    "Value": "0",
    "Description": "Normal client output buffer soft limit in bytes",
    "DataType": "integer",
    "AllowedValues": "0-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "client-output-buffer-limit-normal-soft-seconds",
    "Value": "0",
    "Description": "Normal client output buffer soft limit in seconds",
    "DataType": "integer",
    "AllowedValues": "0-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "client-output-buffer-limit-pubsub-hard-limit",
    "Value": "33554432",
    "Description": "Pubsub client output buffer hard limit in bytes",
    "DataType": "integer",
    "AllowedValues": "0-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "client-output-buffer-limit-pubsub-soft-limit",
    "Value": "8388608",
    "Description": "Pubsub client output buffer soft limit in bytes",
    "DataType": "integer",
    "AllowedValues": "0-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "client-output-buffer-limit-pubsub-soft-seconds",
    "Value": "60",
```

```
    "Description": "Pubsub client output buffer soft limit in seconds",
    "DataType": "integer",
    "AllowedValues": "0-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "hash-max-ziplist-entries",
    "Value": "512",
    "Description": "The maximum number of hash entries in order for the
dataset to be compressed",
    "DataType": "integer",
    "AllowedValues": "0-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "hash-max-ziplist-value",
    "Value": "64",
    "Description": "The threshold of biggest hash entries in order for the
dataset to be compressed",
    "DataType": "integer",
    "AllowedValues": "0-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "hll-sparse-max-bytes",
    "Value": "3000",
    "Description": "HyperLogLog sparse representation bytes limit",
    "DataType": "integer",
    "AllowedValues": "1-16000",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "lazyfree-lazy-eviction",
    "Value": "no",
    "Description": "Perform an asynchronous delete on evictions",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "lazyfree-lazy-expire",
    "Value": "no",
    "Description": "Perform an asynchronous delete on expired keys",
    "DataType": "string",
```

```
    "AllowedValues": "yes,no",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "lazyfree-lazy-server-del",
    "Value": "no",
    "Description": "Perform an asynchronous delete on key updates",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "lazyfree-lazy-user-del",
    "Value": "no",
    "Description": "Specifies whether the default behavior of DEL command
acts the same as UNLINK",
    "DataType": "string",
    "AllowedValues": "yes,no",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "lfu-decay-time",
    "Value": "1",
    "Description": "The amount of time in minutes to decrement the key
counter for LFU eviction policy",
    "DataType": "integer",
    "AllowedValues": "0-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "lfu-log-factor",
    "Value": "10",
    "Description": "The log factor for incrementing key counter for LFU
eviction policy",
    "DataType": "integer",
    "AllowedValues": "1-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "list-compress-depth",
    "Value": "0",
    "Description": "Number of quicklist ziplist nodes from each side of
the list to exclude from compression. The head and tail of the list are always
uncompressed for fast push/pop operations",
```

```

        "DataType": "integer",
        "AllowedValues": "0-",
        "MinimumEngineVersion": "6.2.4"
    },
    {
        "Name": "maxmemory-policy",
        "Value": "noeviction",
        "Description": "Max memory policy",
        "DataType": "string",
        "AllowedValues": "volatile-lru,allkeys-lru,volatile-lfu,allkeys-
lfu,volatile-random,allkeys-random,volatile-ttl,noeviction",
        "MinimumEngineVersion": "6.2.4"
    },
    {
        "Name": "maxmemory-samples",
        "Value": "3",
        "Description": "Max memory samples",
        "DataType": "integer",
        "AllowedValues": "1-",
        "MinimumEngineVersion": "6.2.4"
    },
    {
        "Name": "notify-keyspace-events",
        "Description": "The keyspace events for Redis to notify Pub/Sub clients
about. By default all notifications are disabled",
        "DataType": "string",
        "MinimumEngineVersion": "6.2.4"
    },
    {
        "Name": "set-max-intset-entries",
        "Value": "512",
        "Description": "The limit in the size of the set in order for the
dataset to be compressed",
        "DataType": "integer",
        "AllowedValues": "0-",
        "MinimumEngineVersion": "6.2.4"
    },
    {
        "Name": "slowlog-log-slower-than",
        "Value": "10000",
        "Description": "The execution time, in microseconds, to exceed in order
for the command to get logged. Note that a negative number disables the slow log,
while a value of zero forces the logging of every command",
        "DataType": "integer",

```



```
    "AllowedValues": "-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "slowlog-max-len",
    "Value": "128",
    "Description": "The length of the slow log. There is no limit to this
length. Just be aware that it will consume memory. You can reclaim memory used by
the slow log with SLOWLOG RESET.",
    "DataType": "integer",
    "AllowedValues": "0-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "stream-node-max-bytes",
    "Value": "4096",
    "Description": "The maximum size of a single node in a stream in bytes",
    "DataType": "integer",
    "AllowedValues": "0-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "stream-node-max-entries",
    "Value": "100",
    "Description": "The maximum number of items a single node in a stream
can contain",
    "DataType": "integer",
    "AllowedValues": "0-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "tcp-keepalive",
    "Value": "300",
    "Description": "If non-zero, send ACKs every given number of seconds",
    "DataType": "integer",
    "AllowedValues": "0-",
    "MinimumEngineVersion": "6.2.4"
  },
  {
    "Name": "timeout",
    "Value": "0",
    "Description": "Close connection if client is idle for a given number of
seconds, or never if 0",
    "DataType": "integer",
```

```

        "AllowedValues": "0,20-",
        "MinimumEngineVersion": "6.2.4"
    },
    {
        "Name": "tracking-table-max-keys",
        "Value": "1000000",
        "Description": "The maximum number of keys allowed for the tracking
table for client side caching",
        "DataType": "integer",
        "AllowedValues": "1-1000000000",
        "MinimumEngineVersion": "6.2.4"
    },
    {
        "Name": "zset-max-ziplist-entries",
        "Value": "128",
        "Description": "The maximum number of sorted set entries in order for
the dataset to be compressed",
        "DataType": "integer",
        "AllowedValues": "0-",
        "MinimumEngineVersion": "6.2.4"
    },
    {
        "Name": "zset-max-ziplist-value",
        "Value": "64",
        "Description": "The threshold of biggest sorted set entries in order for
the dataset to be compressed",
        "DataType": "integer",
        "AllowedValues": "0-",
        "MinimumEngineVersion": "6.2.4"
    }
}
]
}

```

有关更多信息，请参阅《MemoryDB 用户指南》中的[使用参数组配置引擎参数](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeParameters](#)中的。

## describe-snapshots

以下代码示例演示了如何使用 describe-snapshots。

AWS CLI

返回快照列表

以下描述快照返回快照列表。

```
aws memorydb describe-snapshots
```

输出：

```
{
  "Snapshots": [
    {
      "Name": "my-cluster-snapshot",
      "Status": "available",
      "Source": "manual",
      "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx2:snapshot/my-cluster-snapshot",
      "ClusterConfiguration": {
        "Name": "my-cluster",
        "Description": " ",
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "Port": 6379,
        "ParameterGroupName": "default.memorydb-redis6",
        "SubnetGroupName": "my-sg",
        "VpcId": "vpc-862574fc",
        "SnapshotRetentionLimit": 0,
        "SnapshotWindow": "04:30-05:30",
        "NumShards": 2
      }
    }
  ]
}
```

有关更多信息，请参阅 M emoryDB 用户指南中的[快照和恢复](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSnapshots](#)中的。

## describe-subnet-groups

以下代码示例演示了如何使用 describe-subnet-groups。

AWS CLI

返回子网组列表

以下 `describe-subnet-groups` 返回子网组列表。

```
aws memorydb describe-subnet-groups
```

输出

```
{
  "SubnetGroups": [
    {
      "Name": "my-sg",
      "Description": "pat-sg",
      "VpcId": "vpc-86xxx4fc",
      "Subnets": [
        {
          "Identifier": "subnet-faxx84a6",
          "AvailabilityZone": {
            "Name": "us-east-1b"
          }
        },
        {
          "Identifier": "subnet-56xxf61b",
          "AvailabilityZone": {
            "Name": "us-east-1a"
          }
        }
      ],
      "ARN": "arn:aws:memorydb:us-east-1:49165xxxxxx:subnetgroup/my-sg"
    }
  ]
}
```

有关更多信息，请参阅 MemoryDB 用户指南中的[子网和子网组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeSubnetGroups](#)中的。

## describe-users

以下代码示例演示了如何使用 `describe-users`。

### AWS CLI

返回用户列表

以下 `describe-users` 返回用户列表。

```
aws memorydb describe-users
```

输出

```
{
  "Users": [
    {
      "Name": "default",
      "Status": "active",
      "AccessString": "on ~* &* +@all",
      "ACLNames": [
        "open-access"
      ],
      "MinimumEngineVersion": "6.0",
      "Authentication": {
        "Type": "no-password"
      },
      "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:user/default"
    },
    {
      "Name": "my-user",
      "Status": "active",
      "AccessString": "off ~objects:* ~items:* ~public:* resetchannels -@all",
      "ACLNames": [],
      "MinimumEngineVersion": "6.2",
      "Authentication": {
        "Type": "password",
        "PasswordCount": 2
      },
      "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:user/my-user"
    }
  ]
}
```

有关更多信息，请参阅《MemoryDB [用户指南](#)》中的[使用访问控制列表对用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeUsers](#)中的。

## failover-shard

以下代码示例演示了如何使用 `failover-shard`。

## AWS CLI

### 对分片进行故障切换

以下故障转移分片在分片上进行故障转移。

```
aws memorydb failover-shard \  
  --cluster-name my-cluster --shard-name 0001
```

输出：

```
{  
  "Cluster": {  
    "Name": "my-cluster",  
    "Status": "available",  
    "NumberOfShards": 2,  
    "ClusterEndpoint": {  
      "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-  
east-1.amazonaws.com",  
      "Port": 6379  
    },  
    "NodeType": "db.r6g.large",  
    "EngineVersion": "6.2",  
    "EnginePatchVersion": "6.2.6",  
    "ParameterGroupName": "default.memorydb-redis6",  
    "ParameterGroupStatus": "in-sync",  
    "SecurityGroups": [  
      {  
        "SecurityGroupId": "sg-0a143xxxx45c9fae",  
        "Status": "active"  
      }  
    ],  
    "SubnetGroupName": "my-sg",  
    "TLSEnabled": true,  
    "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:cluster/my-cluster",  
    "SnapshotRetentionLimit": 0,  
    "MaintenanceWindow": "wed:03:00-wed:04:00",  
    "SnapshotWindow": "04:30-05:30",  
    "AutoMinorVersionUpgrade": true  
  }  
}
```

有关更多信息，请参阅 Memory DB 用户指南中的[使用多可用区最大限度地减少停机时间](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[FailoverShard](#)中的。

## list-allowed-node-type-updates

以下代码示例演示了如何使用 list-allowed-node-type-updates。

### AWS CLI

返回允许的节点类型更新的列表

以下 list-allowed-node-type-updates 返回可用节点类型更新的列表。

```
aws memorydb list-allowed-node-type-updates
```

输出：

```
{
  "Cluster": {
    "Name": "my-cluster",
    "Status": "available",
    "NumberOfShards": 2,
    "ClusterEndpoint": {
      "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
      "Port": 6379
    },
    "NodeType": "db.r6g.large",
    "EngineVersion": "6.2",
    "EnginePatchVersion": "6.2.6",
    "ParameterGroupName": "default.memorydb-redis6",
    "ParameterGroupStatus": "in-sync",
    "SecurityGroups": [
      {
        "SecurityGroupId": "sg-0a143xxxx45c9fae",
        "Status": "active"
      }
    ],
    "SubnetGroupName": "my-sg",
    "TLSEnabled": true,
    "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:cluster/my-cluster",
    "SnapshotRetentionLimit": 0,
    "MaintenanceWindow": "wed:03:00-wed:04:00",
    "SnapshotWindow": "04:30-05:30",
  }
}
```

```
    "AutoMinorVersionUpgrade": true
  }
}
```

有关更多信息，请参阅《MemoryDB 用户指南》中的[缩放](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAllowedNodeTypeUpdates](#)中的。

## list-tags

以下代码示例演示了如何使用 list-tags。

### AWS CLI

返回标签列表

以下列表标签返回标签列表。

```
aws memorydb list-tags \
  --resource-arn arn:aws:memorydb:us-east-1:491658xxxxxx:cluster/my-cluster
```

输出：

```
{
  "TagList": [
    {
      "Key": "mytag",
      "Value": "myvalue"
    }
  ]
}
```

有关更多信息，请参阅《MemoryDB 用户指南》中的[标记资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTags](#)中的。

## reset-parameter-group

以下代码示例演示了如何使用 reset-parameter-group。

### AWS CLI

重置参数组



以下 `reset-parameter-group` 重置参数组。

```
aws memorydb reset-parameter-group \  
  --parameter-group-name my-parameter-group \  
  --all-parameters
```

输出：

```
{  
  "ParameterGroup": {  
    "Name": "my-parameter-group",  
    "Family": "memorydb_redis6",  
    "Description": "my parameter group",  
    "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:parametergroup/my-parameter-group"  
  }  
}
```

有关更多信息，请参阅《MemoryDB 用户指南》中的[使用参数组配置引擎参数](#)。

• 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ResetParameterGroup](#) 中的。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

AWS CLI

为资源添加标签

以下 `tag-resource` 为资源添加标签。

```
aws memorydb tag-resource \  
  --resource-arn arn:aws:memorydb:us-east-1:491658xxxxxx:cluster/my-cluster \  
  --tags Key="mykey",Value="myvalue"
```

输出：

```
{  
  "TagList": [  
    {
```

```
        "Key": "mytag",
        "Value": "myvalue"
    },
    {
        "Key": "mykey",
        "Value": "myvalue"
    }
]
}
```

有关更多信息，请参阅《MemoryDB 用户指南》中的[标记资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

#### 更新 ACL

以下 update-acl 通过添加用户来更新 ACL。

```
aws memorydb untag-resource \
  --resource-arn arn:aws:memorydb:us-east-1:491658xxxxx:cluster/my-cluster \
  --tag-keys mykey
```

输出：

```
{
  "TagList": [
    {
      "Key": "mytag",
      "Value": "myvalue"
    }
  ]
}
```

有关更多信息，请参阅《MemoryDB 用户指南》中的[标记资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-cluster

以下代码示例演示了如何使用 update-cluster。

### AWS CLI

#### 更新集群

以下 update-cluster 将集群的参数组更新为 my-parameter-group

```
aws memorydb update-cluster \  
  --cluster-name my-cluster \  
  --parameter-group-name my-parameter-group
```

输出：

```
{  
  "Cluster": {  
    "Name": "my-cluster",  
    "Status": "available",  
    "NumberOfShards": 2,  
    "AvailabilityMode": "MultiAZ",  
    "ClusterEndpoint": {  
      "Address": "clustercfg.my-cluster.llru6f.memorydb.us-  
east-1.amazonaws.com",  
      "Port": 6379  
    },  
    "NodeType": "db.r6g.large",  
    "EngineVersion": "6.2",  
    "EnginePatchVersion": "6.2.6",  
    "ParameterGroupName": "my-parameter-group",  
    "ParameterGroupStatus": "in-sync",  
    "SecurityGroups": [  
      {  
        "SecurityGroupId": "sg-0a143xxxxxc9fae",  
        "Status": "active"  
      }  
    ],  
    "SubnetGroupName": "pat-sg",  
    "TLSEnabled": true,  
    "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:cluster/my-cluster",  
    "SnapshotRetentionLimit": 0,  
    "MaintenanceWindow": "wed:03:00-wed:04:00",  
  }  
}
```

```
    "SnapshotWindow": "04:30-05:30",
    "ACLName": "my-acl",
    "AutoMinorVersionUpgrade": true
  }
}
```

有关更多信息，请参阅 *MemoryDB 用户指南* 中的 [修改集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateCluster](#) 中的。

## update-parameter-group

以下代码示例演示了如何使用 `update-parameter-group`。

### AWS CLI

#### 更新参数组

以下 `update-parameter-group` 更新参数组。

```
aws memorydb update-parameter-group \
  --parameter-group-name my-parameter-group \
  --parameter-name-values "ParameterName=activedefrag, ParameterValue=no"
```

输出：

```
{
  "ParameterGroup": {
    "Name": "my-parameter-group",
    "Family": "memorydb_redis6",
    "Description": "my parameter group",
    "ARN": "arn:aws:memorydb:us-east-1:49165xxxxxx:parametergroup/my-parameter-
group"
  }
}
```

有关更多信息，请参阅《*MemoryDB 用户指南*》中的 [修改参数组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateParameterGroup](#) 中的。

## update-subnet-group

以下代码示例演示了如何使用 `update-subnet-group`。

## AWS CLI

### 更新子网组

以下 `update-subnet-group` 更新子网组的子网 ID。

```
aws memorydb update-subnet-group \  
  --subnet-group-name my-sg \  
  --subnet-ids subnet-01f29d458f3xxxxxx
```

输出：

```
{  
  "SubnetGroup": {  
    "Name": "my-sg-1",  
    "Description": "my-sg",  
    "VpcId": "vpc-09d2cfc01xxxxxxxx",  
    "Subnets": [  
      {  
        "Identifier": "subnet-01f29d458f3xxxxxx",  
        "AvailabilityZone": {  
          "Name": "us-east-1a"  
        }  
      }  
    ],  
    "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:subnetgroup/my-sg"  
  }  
}
```

有关更多信息，请参阅 *MemoryDB 用户指南* 中的 [子网和子网组](#)。

- 有关 API 的详细信息，请参阅 *AWS CLI 命令参考* [UpdateSubnetGroup](#) 中的。

## update-user

以下代码示例演示了如何使用 `update-user`。

## AWS CLI

### 更新用户

以下内容 `update-user` 修改了用户的访问字符串。

```
aws memorydb update-user \  
  --user-name my-user \  
  --access-string "off ~objects:* ~items:* ~public:* resetchannels -@all"
```

输出：

```
{  
  "User": {  
    "Name": "my-user",  
    "Status": "modifying",  
    "AccessString": "off ~objects:* ~items:* ~public:* resetchannels -@all",  
    "ACLNames": [  
      "myt-acl"  
    ],  
    "MinimumEngineVersion": "6.2",  
    "Authentication": {  
      "Type": "password",  
      "PasswordCount": 2  
    },  
    "ARN": "arn:aws:memorydb:us-east-1:491658xxxxxx:user/my-user"  
  }  
}
```

有关更多信息，请参阅《MemoryDB [用户指南](#)》中的[使用访问控制列表对用户进行身份验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUser](#)中的。

## 使用 Amazon MSK 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon MSK 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### create-cluster

以下代码示例演示了如何使用 `create-cluster`。

#### AWS CLI

##### 创建 Amazon MSK 集群

以下 `create-cluster` 示例创建了一个名为 `MessagingCluster` 的 MSK 集群，其中包含三个代理节点。名为 `brokernodegroupinfo.json` 的 JSON 文件指定了您希望 Amazon MSK 在其上分配代理节点的三个子网。此示例未指定监控级别，因此集群将获得该 `DEFAULT` 级别。

```
aws kafka create-cluster \  
  --cluster-name "MessagingCluster" \  
  --broker-node-group-info file://brokernodegroupinfo.json \  
  --kafka-version "2.2.1" \  
  --number-of-broker-nodes 3
```

`brokernodegroupinfo.json` 的内容：

```
{  
  "InstanceType": "kafka.m5.xlarge",  
  "BrokerAZDistribution": "DEFAULT",  
  "ClientSubnets": [  
    "subnet-0123456789111abcd",  
    "subnet-0123456789222abcd",  
    "subnet-0123456789333abcd"  
  ]  
}
```

输出：

```
{  
  "ClusterArn": "arn:aws:kafka:us-west-2:123456789012:cluster/MessagingCluster/  
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE-2",  
  "ClusterName": "MessagingCluster",  
  "State": "CREATING"
```

```
}
```

有关更多信息，请参阅[在 Apache Kafka Managed Streaming for Apache 的亚马逊托管流中创建亚马逊 MSK 集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCluster](#)中的。

## create-configuration

以下代码示例演示了如何使用 create-configuration。

### AWS CLI

#### 创建自定义 Amazon MSK 配置

以下create-configuration示例使用在输入文件中指定的服务器属性创建自定义 MSK 配置。

```
aws kafka create-configuration \  
  --name "CustomConfiguration" \  
  --description "Topic autocreation enabled; Apache ZooKeeper timeout 2000 ms; Log  
rolling 604800000 ms." \  
  --kafka-versions "2.2.1" \  
  --server-properties file://configuration.txt
```

configuration.txt 的内容：

```
auto.create.topics.enable = true  
zookeeper.connection.timeout.ms = 2000  
log.roll.ms = 604800000
```

此命令不生成任何输出。输出：

```
{  
  "Arn": "arn:aws:kafka:us-west-2:123456789012:configuration/CustomConfiguration/  
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE-2",  
  "CreationTime": "2019-10-09T15:26:05.548Z",  
  "LatestRevision":  
    {  
      "CreationTime": "2019-10-09T15:26:05.548Z",  
      "Description": "Topic autocreation enabled; Apache ZooKeeper timeout  
2000 ms; Log rolling 604800000 ms.",  
      "Revision": 1
```



```
    },  
    "Name": "CustomConfiguration"  
  }  
}
```

有关更多信息，请参阅《适用于 Apache [Managed Streaming 的 Apache Kafka 开发者指南](#)》中的 [亚马逊 MSK 配置操作](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateConfiguration](#) 中的。

## describe-cluster

以下代码示例演示了如何使用 describe-cluster。

### AWS CLI

#### 描述集群

以下 describe-cluster 示例描述了一个 Amazon MSK 集群。

```
aws kafka describe-cluster \  
  --cluster-arn arn:aws:kafka:us-east-1:123456789012:cluster/demo-  
cluster-1/6357e0b2-0e6a-4b86-a0b4-70df934c2e31-5
```

输出：

```
{  
  "ClusterInfo": {  
    "BrokerNodeGroupInfo": {  
      "BrokerAZDistribution": "DEFAULT",  
      "ClientSubnets": [  
        "subnet-cbfff283",  
        "subnet-6746046b"  
      ],  
      "InstanceType": "kafka.m5.large",  
      "SecurityGroups": [  
        "sg-f839b688"  
      ],  
      "StorageInfo": {  
        "EbsStorageInfo": {  
          "VolumeSize": 100  
        }  
      }  
    }  
  }  
}
```

```

    },
    "ClusterArn": "arn:aws:kafka:us-east-1:123456789012:cluster/demo-
cluster-1/6357e0b2-0e6a-4b86-a0b4-70df934c2e31-5",
    "ClusterName": "demo-cluster-1",
    "CreationTime": "2020-07-09T02:31:36.223000+00:00",
    "CurrentBrokerSoftwareInfo": {
      "KafkaVersion": "2.2.1"
    },
    "CurrentVersion": "K3AEGXETSR30VB",
    "EncryptionInfo": {
      "EncryptionAtRest": {
        "DataVolumeKMSKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a7ca56d5-0768-4b64-a670-339a9fbef81c"
      },
      "EncryptionInTransit": {
        "ClientBroker": "TLS_PLAINTEXT",
        "InCluster": true
      }
    },
    "EnhancedMonitoring": "DEFAULT",
    "OpenMonitoring": {
      "Prometheus": {
        "JmxExporter": {
          "EnabledInBroker": false
        },
        "NodeExporter": {
          "EnabledInBroker": false
        }
      }
    },
    "NumberOfBrokerNodes": 2,
    "State": "ACTIVE",
    "Tags": {},
    "ZookeeperConnectString": "z-2.demo-cluster-1.xuy0sb.c5.kafka.us-
east-1.amazonaws.com:2181,z-1.demo-cluster-1.xuy0sb.c5.kafka.us-
east-1.amazonaws.com:2181,z-3.demo-cluster-1.xuy0sb.c5.kafka.us-
east-1.amazonaws.com:2181"
  }
}

```

有关更多信息，请参阅《适用于 Apache Managed Streaming 的亚马逊托管流 Kafka 开发者指南》中的“列出[亚马逊 MSK 集群](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeCluster](#)中的。

## get-bootstrap-brokers

以下代码示例演示了如何使用 `get-bootstrap-brokers`。

### AWS CLI

获取引导经纪人

以下`get-bootstrap-brokers`示例检索 Amazon MSK 集群的引导代理信息。

```
aws kafka get-bootstrap-brokers \  
  --cluster-arn arn:aws:kafka:us-east-1:123456789012:cluster/demo-  
cluster-1/6357e0b2-0e6a-4b86-a0b4-70df934c2e31-5
```

输出：

```
{  
  "BootstrapBrokerString": "b-1.demo-cluster-1.xuy0sb.c5.kafka.us-  
east-1.amazonaws.com:9092,b-2.demo-cluster-1.xuy0sb.c5.kafka.us-  
east-1.amazonaws.com:9092",  
  "BootstrapBrokerStringTls": "b-1.demo-cluster-1.xuy0sb.c5.kafka.us-  
east-1.amazonaws.com:9094,b-2.demo-cluster-1.xuy0sb.c5.kafka.us-  
east-1.amazonaws.com:9094"  
}
```

有关更多信息，请参阅 [《适用于 Apache Kafka 的亚马逊托管流媒体 Kafka 开发者指南》](#) 中的“获取 Bootstrap 代理”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetBootstrapBrokers](#) 中的。

## list-clusters

以下代码示例演示了如何使用 `list-clusters`。

### AWS CLI

列出可用集群

以下`list-clusters`示例列出了您 AWS 账户中的 Amazon MSK 集群。

```
aws kafka list-clusters
```

输出：

```
{
  "ClusterInfoList": [
    {
      "BrokerNodeGroupInfo": {
        "BrokerAZDistribution": "DEFAULT",
        "ClientSubnets": [
          "subnet-cbfff283",
          "subnet-6746046b"
        ],
        "InstanceType": "kafka.m5.large",
        "SecurityGroups": [
          "sg-f839b688"
        ],
        "StorageInfo": {
          "EbsStorageInfo": {
            "VolumeSize": 100
          }
        }
      },
      "ClusterArn": "arn:aws:kafka:us-east-1:123456789012:cluster/demo-cluster-1/6357e0b2-0e6a-4b86-a0b4-70df934c2e31-5",
      "ClusterName": "demo-cluster-1",
      "CreationTime": "2020-07-09T02:31:36.223000+00:00",
      "CurrentBrokerSoftwareInfo": {
        "KafkaVersion": "2.2.1"
      },
      "CurrentVersion": "K3AEGXETSR30VB",
      "EncryptionInfo": {
        "EncryptionAtRest": {
          "DataVolumeKMSKeyId": "arn:aws:kms:us-east-1:123456789012:key/a7ca56d5-0768-4b64-a670-339a9fbef81c"
        },
        "EncryptionInTransit": {
          "ClientBroker": "TLS_PLAINTEXT",
          "InCluster": true
        }
      },
      "EnhancedMonitoring": "DEFAULT",
      "OpenMonitoring": {
        "Prometheus": {
          "JmxExporter": {
            "EnabledInBroker": false
          }
        }
      }
    }
  ]
}
```

```

        },
        "NodeExporter": {
            "EnabledInBroker": false
        }
    },
    "NumberOfBrokerNodes": 2,
    "State": "ACTIVE",
    "Tags": {},
    "ZookeeperConnectionString": "z-2.demo-cluster-1.xuy0sb.c5.kafka.us-
east-1.amazonaws.com:2181,z-1.demo-cluster-1.xuy0sb.c5.kafka.us-
east-1.amazonaws.com:2181,z-3.demo-cluster-1.xuy0sb.c5.kafka.us-
east-1.amazonaws.com:2181"
    }
]
}

```

有关更多信息，请参阅《适用于 Apache Managed Streaming 的亚马逊托管流 Kafka 开发者指南》中的“列出[亚马逊 MSK 集群](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListClusters](#) 中的。

## update-broker-storage

以下代码示例演示了如何使用 update-broker-storage。

### AWS CLI

#### 更新经纪商的 EBS 存储

以下 update-broker-storage 示例更新了集群中所有代理的 EBS 存储量。Amazon MSK 将每个代理的目标存储量设置为示例中指定的存储量。您可以通过描述集群或列出所有集群来获取集群的当前版本。

```

aws kafka update-broker-storage \
  --cluster-arn "arn:aws:kafka:us-west-2:123456789012:cluster/MessagingCluster/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE-2" \
  --current-version "K21V3IB1VIZYYH" \
  --target-broker-efs-volume-info "KafkaBrokerNodeId=ALL,VolumeSizeGB=1100"

```

输出返回此 update-broker-storage 操作的 ARN。要确定此操作是否已完成，请使用以此 ARN 作为输入的 describe-cluster-operation 命令。

```
{
  "ClusterArn": "arn:aws:kafka:us-west-2:123456789012:cluster/MessagingCluster/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE-2",
  "ClusterOperationArn": "arn:aws:kafka:us-west-2:123456789012:cluster-
operation/V123450123/a1b2c3d4-1234-abcd-cdef-22222EXAMPLE-2/a1b2c3d4-abcd-1234-
bcde-33333EXAMPLE"
}
```

有关更多信息，请参阅 [《适用于 Apache Kafka 的亚马逊托管流媒体 Kafka 开发者指南》](#) 中的 [更新经纪商的 EBS 存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateBrokerStorage](#) 中的。

## update-cluster-configuration

以下代码示例演示了如何使用 update-cluster-configuration。

### AWS CLI

#### 更新 Amazon MSK 集群的配置

以下 update-cluster-configuration 示例更新了指定的现有 MSK 集群的配置。它使用自定义 MSK 配置。

```
aws kafka update-cluster-configuration \
  --cluster-arn "arn:aws:kafka:us-west-2:123456789012:cluster/MessagingCluster/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE-2" \
  --configuration-info file://configuration-info.json \
  --current-version "K21V3IB1VIZYYH"
```

configuration-info.json 的内容：

```
{
  "Arn": "arn:aws:kafka:us-west-2:123456789012:configuration/CustomConfiguration/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE-2",
  "Revision": 1
}
```

输出返回此 update-cluster-configuration 操作的 ARN。要确定此操作是否已完成，请使用以此 ARN 作为输入的 describe-cluster-operation 命令。

```
{
  "ClusterArn": "arn:aws:kafka:us-west-2:123456789012:cluster/MessagingCluster/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE-2",
  "ClusterOperationArn": "arn:aws:kafka:us-west-2:123456789012:cluster-
operation/V123450123/a1b2c3d4-1234-abcd-cdef-22222EXAMPLE-2/a1b2c3d4-abcd-1234-
bcde-33333EXAMPLE"
}
```

有关更多信息，请参阅 [《适用于 Apache Kafka 的亚马逊托管流媒体 Kafka 开发者指南》](#) 中的“[更新亚马逊 MSK 集群的配置](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateClusterConfiguration](#) 中的。

## 使用网络管理器示例 AWS CLI

以下代码示例向您展示了如何通过 AWS Command Line Interface 与网络管理器一起使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-customer-gateway**

以下代码示例演示了如何使用 associate-customer-gateway。

AWS CLI

关联客户网关

以下 associate-customer-gateway 示例将指定全球网络 cgw-11223344556677889 中的客户网关与设备相关联 device-07f6fd08867abc123。

```
aws networkmanager associate-customer-gateway \  
  --customer-gateway-arn arn:aws:ec2:us-west-2:123456789012:customer-gateway/  
cgw-11223344556677889 \  
  --global-network-id global-network-01231231231231231 \  
  --device-id device-07f6fd08867abc123 \  
  --region us-west-2
```

输出：

```
{  
  "CustomerGatewayAssociation": {  
    "CustomerGatewayArn": "arn:aws:ec2:us-west-2:123456789012:customer-gateway/  
cgw-11223344556677889",  
    "GlobalNetworkId": "global-network-01231231231231231",  
    "DeviceId": "device-07f6fd08867abc123",  
    "State": "PENDING"  
  }  
}
```

有关更多信息，请参阅 [T ransit Gateway 网络管理器指南中的客户网关关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateCustomerGateway](#) 中的。

## associate-link

以下代码示例演示了如何使用 `associate-link`。

### AWS CLI

#### 关联链接

以下 `associate-link` 示例将链接 `link-11112222aaaabbbb1` 与设备关联起来 `device-07f6fd08867abc123`。链路和设备位于指定的全局网络中。

```
aws networkmanager associate-link \  
  --global-network-id global-network-01231231231231231 \  
  --device-id device-07f6fd08867abc123 \  
  --link-id link-11112222aaaabbbb1 \  
  --region us-west-2
```

输出：



```
{
  "LinkAssociation": {
    "GlobalNetworkId": "global-network-01231231231231231",
    "DeviceId": "device-07f6fd08867abc123",
    "LinkId": "link-11112222aaaabbbb1",
    "LinkAssociationState": "PENDING"
  }
}
```

有关更多信息，请参阅 Transit Gateway 网络管理器指南中的设备和链路[关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateLink](#) 中的。

## create-core-network

以下代码示例演示了如何使用 create-core-network。

### AWS CLI

#### 创建核心网络

以下 create-core-network 示例使用可选描述和标签在 AWS Cloud WAN 全球网络中创建核心网络。

```
aws networkmanager create-core-network \
  --global-network-id global-network-0d59060f16a73bc41 \
  --description "Main headquarters location" \
  --tags Key=Name,Value="New York City office"
```

#### 输出：

```
{
  "CoreNetwork": {
    "GlobalNetworkId": "global-network-0d59060f16a73bc41",
    "CoreNetworkId": "core-network-0fab62fe438d94db6",
    "CoreNetworkArn": "arn:aws:networkmanager::987654321012:core-network/core-network-0fab62fe438d94db6",
    "Description": "Main headquarters location",
    "CreatedAt": "2022-01-10T19:53:59+00:00",
    "State": "AVAILABLE",
    "Tags": [
```

```
    {
      "Key": "Name",
      "Value": "New York City office"
    }
  ]
}
```

有关更多信息，请参阅 AWS Cloud WAN 用户指南中的[核心网络](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateCoreNetwork](#)中的。

## create-device

以下代码示例演示了如何使用 create-device。

### AWS CLI

#### 创建设备

以下 create-device 示例在指定的全局网络中创建设备。设备详细信息包括描述、类型、供应商、型号和序列号。

```
aws networkmanager create-device
  --global-network-id global-network-01231231231231231 \
  --description "New York office device" \
  --type "office device" \
  --vendor "anycompany" \
  --model "abcabc" \
  --serial-number "1234" \
  --region us-west-2
```

输出：

```
{
  "Device": {
    "DeviceId": "device-07f6fd08867abc123",
    "DeviceArn": "arn:aws:networkmanager::123456789012:device/global-
network-01231231231231231/device-07f6fd08867abc123",
    "GlobalNetworkId": "global-network-01231231231231231",
    "Description": "New York office device",
    "Type": "office device",
```

```
    "Vendor": "anycompany",
    "Model": "abcabc",
    "SerialNumber": "1234",
    "CreatedAt": 1575554005.0,
    "State": "PENDING"
  }
}
```

有关更多信息，请参阅 [T ransit Gateway 网络管理器指南中的使用设备](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDevice](#)中的。

## create-global-network

以下代码示例演示了如何使用 create-global-network。

### AWS CLI

#### 创建全球网络

以下create-global-network示例创建了一个新的全球网络。创建时的初始状态为PENDING。

```
aws networkmanager create-global-network
```

输出：

```
{
  "GlobalNetwork": {
    "GlobalNetworkId": "global-network-00a77fc0f722dae74",
    "GlobalNetworkArn": "arn:aws:networkmanager::987654321012:global-network/global-network-00a77fc0f722dae74",
    "CreatedAt": "2022-03-14T20:31:56+00:00",
    "State": "PENDING"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateGlobalNetwork](#)中的。

## create-link

以下代码示例演示了如何使用 create-link。

## AWS CLI

### 创建链接

以下`create-link`示例在指定的全球网络中创建链接。该链接包括有关链路类型、带宽和提供商的描述和详细信息。站点 ID 表示与该链接关联的站点。

```
aws networkmanager create-link \  
  --global-network-id global-network-01231231231231231 \  
  --description "VPN Link" \  
  --type "broadband" \  
  --bandwidth UploadSpeed=10,DownloadSpeed=20 \  
  --provider "AnyCompany" \  
  --site-id site-444555aaaabbb11223 \  
  --region us-west-2
```

输出：

```
{  
  "Link": {  
    "LinkId": "link-11112222aaaabbbb1",  
    "LinkArn": "arn:aws:networkmanager::123456789012:link/global-  
network-01231231231231231/link-11112222aaaabbbb1",  
    "GlobalNetworkId": "global-network-01231231231231231",  
    "SiteId": "site-444555aaaabbb11223",  
    "Description": "VPN Link",  
    "Type": "broadband",  
    "Bandwidth": {  
      "UploadSpeed": 10,  
      "DownloadSpeed": 20  
    },  
    "Provider": "AnyCompany",  
    "CreatedAt": 1575555811.0,  
    "State": "PENDING"  
  }  
}
```

有关更多信息，请参阅 [T ransit Gateway 网络管理器指南中的使用链接](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLink](#)中的。

## create-site

以下代码示例演示了如何使用 create-site。

### AWS CLI

#### 创建网站

以下create-site示例在指定的全球网络中创建一个站点。网站详细信息包括描述和位置信息。

```
aws networkmanager create-site \  
  --global-network-id global-network-01231231231231231 \  
  --description "New York head office" \  
  --location Latitude=40.7128,Longitude=-74.0060 \  
  --region us-west-2
```

输出：

```
{  
  "Site": {  
    "SiteId": "site-444555aaabbb11223",  
    "SiteArn": "arn:aws:networkmanager::123456789012:site/global-  
network-01231231231231231/site-444555aaabbb11223",  
    "GlobalNetworkId": "global-network-01231231231231231",  
    "Description": "New York head office",  
    "Location": {  
      "Latitude": "40.7128",  
      "Longitude": "-74.0060"  
    },  
    "CreatedAt": 1575554300.0,  
    "State": "PENDING"  
  }  
}
```

有关更多信息，请参阅 [T ransit Gateway 网络管理器指南中的使用站点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSite](#)中的。

## create-vpc-attachment

以下代码示例演示了如何使用 create-vpc-attachment。

## AWS CLI

### 创建 VPC 附件

以下`create-vpc-attachment`示例在核心网络中创建支持 IPv6 的 VPC 连接。

```
aws networkmanager create-vpc-attachment \  
  --core-network-id core-network-0fab62fe438d94db6 \  
  --vpc-arn arn:aws:ec2:us-east-1:987654321012:vpc/vpc-09f37f69e2786eeb8 \  
  --subnet-arns arn:aws:ec2:us-east-1:987654321012:subnet/subnet-04ca4e010857e7bb7 \  
  --Ipv6Support=true
```

输出：

```
{  
  "VpcAttachment": {  
    "Attachment": {  
      "CoreNetworkId": "core-network-0fab62fe438d94db6",  
      "AttachmentId": "attachment-05e1da6eba87a06e6",  
      "OwnerAccountId": "987654321012",  
      "AttachmentType": "VPC",  
      "State": "CREATING",  
      "EdgeLocation": "us-east-1",  
      "ResourceArn": "arn:aws:ec2:us-east-1:987654321012:vpc/  
vpc-09f37f69e2786eeb8",  
      "Tags": [],  
      "CreatedAt": "2022-03-10T20:59:14+00:00",  
      "UpdatedAt": "2022-03-10T20:59:14+00:00"  
    },  
    "SubnetArns": [  
      "arn:aws:ec2:us-east-1:987654321012:subnet/subnet-04ca4e010857e7bb7"  
    ],  
    "Options": {  
      "Ipv6Support": true  
    }  
  }  
}
```

有关更多信息，请参阅 Cloud WAN 用户指南中的[创建附件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateVpcAttachment](#)中的。

## delete-attachment

以下代码示例演示了如何使用 delete-attachment。

### AWS CLI

#### 删除附件

以下delete-attachment示例删除了 Connect 附件。

```
aws networkmanager delete-attachment \  
  --attachment-id attachment-01feddaeae26ab68c
```

输出：

```
{  
  "Attachment": {  
    "CoreNetworkId": "core-network-0f4b0a9d5ee7761d1",  
    "AttachmentId": "attachment-01feddaeae26ab68c",  
    "OwnerAccountId": "987654321012",  
    "AttachmentType": "CONNECT",  
    "State": "DELETING",  
    "EdgeLocation": "us-east-1",  
    "ResourceArn": "arn:aws:networkmanager::987654321012:attachment/  
attachment-02c3964448fedf5aa",  
    "CreatedAt": "2022-03-15T19:18:41+00:00",  
    "UpdatedAt": "2022-03-15T19:28:59+00:00"  
  }  
}
```

有关更多信息，请参阅 Cloud WAN 用户指南中的[删除附件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAttachment](#)中的。

## delete-bucket-analytics-configuration

以下代码示例演示了如何使用 delete-bucket-analytics-configuration。

### AWS CLI

#### 删除存储桶的分析配置

以下 `delete-bucket-analytics-configuration` 示例移除指定存储桶和 ID 的分析配置。

```
aws s3api delete-bucket-analytics-configuration \  
  --bucket my-bucket \  
  --id 1
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketAnalyticsConfiguration](#)中的。

## **delete-bucket-metrics-configuration**

以下代码示例演示了如何使用 `delete-bucket-metrics-configuration`。

### AWS CLI

删除存储桶的指标配置

以下 `delete-bucket-metrics-configuration` 示例移除指定存储桶和 ID 的指标配置。

```
aws s3api delete-bucket-metrics-configuration \  
  --bucket my-bucket \  
  --id 123
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketMetricsConfiguration](#)中的。

## **delete-core-network**

以下代码示例演示了如何使用 `delete-core-network`。

### AWS CLI

删除核心网络

以下`delete-core-network`示例从 Cloud WAN 全球网络中删除核心网络。

```
aws networkmanager delete-core-network \  
  --core-network-id core-network-0fab62fe438d94db6
```



输出：

```
{
  "CoreNetwork": {
    "GlobalNetworkId": "global-network-0d59060f16a73bc41",
    "CoreNetworkId": "core-network-0fab62fe438d94db6",
    "Description": "Main headquarters location",
    "CreatedAt": "2021-12-09T18:31:11+00:00",
    "State": "DELETING",
    "Segments": [
      {
        "Name": "dev",
        "EdgeLocations": [
          "us-east-1"
        ],
        "SharedSegments": []
      }
    ],
    "Edges": [
      {
        "EdgeLocation": "us-east-1",
        "Asn": 64512,
        "InsideCidrBlocks": []
      }
    ]
  }
}
```

有关更多信息，请参阅 Cloud WAN 用户指南中的[核心网络](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCoreNetwork](#)中的。

## delete-device

以下代码示例演示了如何使用 delete-device。

### AWS CLI

#### 删除设备

以下delete-device示例从指定的全局网络中删除指定设备。

```
aws networkmanager delete-device \
```

```
--global-network-id global-network-01231231231231231 \  
--device-id device-07f6fd08867abc123 \  
--region us-west-2
```

输出：

```
{  
  "Device": {  
    "DeviceId": "device-07f6fd08867abc123",  
    "DeviceArn": "arn:aws:networkmanager::123456789012:device/global-  
network-01231231231231231/device-07f6fd08867abc123",  
    "GlobalNetworkId": "global-network-01231231231231231",  
    "Description": "New York office device",  
    "Type": "office device",  
    "Vendor": "anycompany",  
    "Model": "abcabc",  
    "SerialNumber": "1234",  
    "SiteId": "site-444555aaabbb11223",  
    "CreatedAt": 1575554005.0,  
    "State": "DELETING"  
  }  
}
```

有关更多信息，请参阅 [T ransit Gateway 网络管理器指南中的使用设备](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteDevice](#) 中的。

## delete-global-network

以下代码示例演示了如何使用 delete-global-network。

### AWS CLI

#### 删除全球网络

以下 delete-global-network 示例删除了一个全球网络。

```
aws networkmanager delete-global-network \  
--global-network-id global-network-052bedddccb193b6b
```

输出：

```
{
  "GlobalNetwork": {
    "GlobalNetworkId": "global-network-052bedddccb193b6b",
    "GlobalNetworkArn": "arn:aws:networkmanager::987654321012:global-network/global-network-052bedddccb193b6b",
    "CreatedAt": "2021-12-09T18:19:12+00:00",
    "State": "DELETING"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteGlobalNetwork](#)中的。

## delete-link

以下代码示例演示了如何使用 delete-link。

### AWS CLI

#### 删除链接

以下delete-link示例从指定的全球网络中删除指定的链接。

```
aws networkmanager delete-link \
  --global-network-id global-network-01231231231231231 \
  --link-id link-11112222aaaabbbb1 \
  --region us-west-2
```

输出：

```
{
  "Link": {
    "LinkId": "link-11112222aaaabbbb1",
    "LinkArn": "arn:aws:networkmanager::123456789012:link/global-network-01231231231231231/link-11112222aaaabbbb1",
    "GlobalNetworkId": "global-network-01231231231231231",
    "SiteId": "site-444555aaaabbb11223",
    "Description": "VPN Link",
    "Type": "broadband",
    "Bandwidth": {
      "UploadSpeed": 20,
      "DownloadSpeed": 20
    }
  }
}
```

```
    },  
    "Provider": "AnyCompany",  
    "CreatedAt": 1575555811.0,  
    "State": "DELETING"  
  }  
}
```

有关更多信息，请参阅 [T ransit Gateway 网络管理器指南中的使用链接](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 DeleteLink](#) 中的。

## delete-public-access-block

以下代码示例演示了如何使用 delete-public-access-block。

### AWS CLI

删除存储桶的屏蔽公共访问权限配置

以下 delete-public-access-block 示例移除指定存储桶上的屏蔽公共访问权限配置。

```
aws s3api delete-public-access-block \  
  --bucket my-bucket
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 DeletePublicAccessBlock](#) 中的。

## delete-site

以下代码示例演示了如何使用 delete-site。

### AWS CLI

删除站点

以下 delete-site 示例删除指定全球网络中的指定站点 (site-444555aaabbb11223)。

```
aws networkmanager delete-site \  
  --global-network-id global-network-01231231231231231 \  
  --site-id site-444555aaabbb11223 \  
  \
```

```
--region us-west-2
```

输出：

```
{
  "Site": {
    "SiteId": "site-444555aaabbb11223",
    "SiteArn": "arn:aws:networkmanager::123456789012:site/global-
network-01231231231231231/site-444555aaabbb11223",
    "GlobalNetworkId": "global-network-01231231231231231",
    "Description": "New York head office",
    "Location": {
      "Latitude": "40.7128",
      "Longitude": "-74.0060"
    },
    "CreatedAt": 1575554300.0,
    "State": "DELETING"
  }
}
```

有关更多信息，请参阅 [T transit Gateway 网络管理器指南中的使用站点](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 DeleteSite](#) 中的。

## deregister-transit-gateway

以下代码示例演示了如何使用 deregister-transit-gateway。

### AWS CLI

从全球网络中注销公交网关

以下deregister-transit-gateway示例从指定的全球网络取消注册指定的传输网关。

```
aws networkmanager deregister-transit-gateway \
  --global-network-id global-network-01231231231231231 \
  --transit-gateway-arn arn:aws:ec2:us-west-2:123456789012:transit-gateway/
tgw-123abc05e04123abc \
  --region us-west-2
```

输出：

```
{
  "TransitGatewayRegistration": {
    "GlobalNetworkId": "global-network-01231231231231231",
    "TransitGatewayArn": "arn:aws:ec2:us-west-2:123456789012:transit-gateway/tgw-123abc05e04123abc",
    "State": {
      "Code": "DELETING"
    }
  }
}
```

有关更多信息，请参阅 [Transit Gateway 网络管理器指南中的 Transit Gateway 注册](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeregisterTransitGateway](#) 中的。

## describe-global-networks

以下代码示例演示了如何使用 describe-global-networks。

### AWS CLI

描述您的全球网络

以下 describe-global-networks 示例描述了您账户中的所有全球网络。

```
aws networkmanager describe-global-networks \
  --region us-west-2
```

输出：

```
{
  "GlobalNetworks": [
    {
      "GlobalNetworkId": "global-network-01231231231231231",
      "GlobalNetworkArn": "arn:aws:networkmanager::123456789012:global-network/global-network-01231231231231231",
      "Description": "Company 1 global network",
      "CreatedAt": 1575553525.0,
      "State": "AVAILABLE"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeGlobalNetworks](#)中的。

## disassociate-customer-gateway

以下代码示例演示了如何使用 disassociate-customer-gateway。

### AWS CLI

取消与客户网关的关联

以下disassociate-customer-gateway示例取消指定客户网关 (cgw-11223344556677889) 与指定全球网络的关联。

```
aws networkmanager disassociate-customer-gateway \  
  --global-network-id global-network-01231231231231231 \  
  --customer-gateway-arn arn:aws:ec2:us-west-2:123456789012:customer-gateway/  
cgw-11223344556677889 \  
  --region us-west-2
```

输出：

```
{  
  "CustomerGatewayAssociation": {  
    "CustomerGatewayArn": "arn:aws:ec2:us-west-2:123456789012:customer-gateway/  
cgw-11223344556677889",  
    "GlobalNetworkId": "global-network-01231231231231231",  
    "DeviceId": "device-07f6fd08867abc123",  
    "State": "DELETING"  
  }  
}
```

有关更多信息，请参阅 Transit Gateway 网络管理器指南中的客户网关[关联](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateCustomerGateway](#)中的。

## disassociate-link

以下代码示例演示了如何使用 disassociate-link。

### AWS CLI

取消关联链接

以下disassociate-link示例取消指定链接与指定全局网络device-07f6fd08867abc123中设备的关联。

```
aws networkmanager disassociate-link \  
  --global-network-id global-network-01231231231231231 \  
  --device-id device-07f6fd08867abc123 \  
  --link-id link-11112222aaaabbbb1 \  
  --region us-west-2
```

输出：

```
{  
  "LinkAssociation": {  
    "GlobalNetworkId": "global-network-01231231231231231",  
    "DeviceId": "device-07f6fd08867abc123",  
    "LinkId": "link-11112222aaaabbbb1",  
    "LinkAssociationState": "DELETING"  
  }  
}
```

有关更多信息，请参阅 Transit Gateway 网络管理器指南中的设备和链路[关联](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateLink](#)中的。

## get-bucket-analytics-configuration

以下代码示例演示了如何使用 get-bucket-analytics-configuration。

AWS CLI

检索具有特定 ID 的存储桶的分析配置

以下 get-bucket-analytics-configuration 示例显示了指定存储桶和 ID 的分析配置。

```
aws s3api get-bucket-analytics-configuration \  
  --bucket my-bucket \  
  --id 1
```

输出：

```
{  
  "AnalyticsConfiguration": {
```



```
    "StorageClassAnalysis": {},
    "Id": "1"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketAnalyticsConfiguration](#)中的。

## get-bucket-metrics-configuration

以下代码示例演示了如何使用 get-bucket-metrics-configuration。

### AWS CLI

检索具有特定 ID 的存储桶的指标配置

以下 get-bucket-metrics-configuration 示例显示了指定存储桶和 ID 的指标配置。

```
aws s3api get-bucket-metrics-configuration \
  --bucket my-bucket \
  --id 123
```

输出：

```
{
  "MetricsConfiguration": {
    "Filter": {
      "Prefix": "logs"
    },
    "Id": "123"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketMetricsConfiguration](#)中的。

## get-customer-gateway-associations

以下代码示例演示了如何使用 get-customer-gateway-associations。

### AWS CLI

获取您的客户网关关联

以下`get-customer-gateway-associations`示例获取指定全球网络的客户网关关联。

```
aws networkmanager get-customer-gateway-associations \  
  --global-network-id global-network-01231231231231231 \  
  --region us-west-2
```

输出：

```
{  
  "CustomerGatewayAssociations": [  
    {  
      "CustomerGatewayArn": "arn:aws:ec2:us-west-2:123456789012:customer-  
gateway/cgw-11223344556677889",  
      "GlobalNetworkId": "global-network-01231231231231231",  
      "DeviceId": "device-07f6fd08867abc123",  
      "State": "AVAILABLE"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCustomerGatewayAssociations](#)中的。

## get-devices

以下代码示例演示了如何使用 `get-devices`。

### AWS CLI

要获取您的设备

以下`get-devices`示例获取指定全球网络中的设备。

```
aws networkmanager get-devices \  
  --global-network-id global-network-01231231231231231 \  
  --region us-west-2
```

输出：

```
{  
  "Devices": [  

```

```
{
  "DeviceId": "device-07f6fd08867abc123",
  "DeviceArn": "arn:aws:networkmanager::123456789012:device/global-
network-01231231231231231231/device-07f6fd08867abc123",
  "GlobalNetworkId": "global-network-01231231231231231",
  "Description": "NY office device",
  "Type": "office device",
  "Vendor": "anycompany",
  "Model": "abcabc",
  "SerialNumber": "1234",
  "CreatedAt": 1575554005.0,
  "State": "AVAILABLE"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDevices](#)中的。

## get-link-associations

以下代码示例演示了如何使用 get-link-associations。

### AWS CLI

获取您的链接关联

以下get-link-associations示例获取指定全球网络中的链接关联。

```
aws networkmanager get-link-associations \
  --global-network-id global-network-01231231231231231 \
  --region us-west-2
```

输出：

```
{
  "LinkAssociations": [
    {
      "GlobalNetworkId": "global-network-01231231231231231",
      "DeviceId": "device-07f6fd08867abc123",
      "LinkId": "link-11112222aaaabbbb1",
      "LinkAssociationState": "AVAILABLE"
    }
  ]
}
```

```
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLinkAssociations](#)中的。

## get-links

以下代码示例演示了如何使用 get-links。

### AWS CLI

获取您的链接

以下get-links示例获取指定全球网络中的链接。

```
aws networkmanager get-links \
  --global-network-id global-network-01231231231231231 \
  --region us-west-2
```

输出：

```
{
  "Links": [
    {
      "LinkId": "link-11112222aaaabbbb1",
      "LinkArn": "arn:aws:networkmanager::123456789012:link/global-
network-01231231231231231/link-11112222aaaabbbb1",
      "GlobalNetworkId": "global-network-01231231231231231",
      "SiteId": "site-444555aaaabbb11223",
      "Description": "VPN Link",
      "Type": "broadband",
      "Bandwidth": {
        "UploadSpeed": 10,
        "DownloadSpeed": 20
      },
      "Provider": "AnyCompany",
      "CreatedAt": 1575555811.0,
      "State": "AVAILABLE"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLinks](#)中的。

## get-object-retention

以下代码示例演示了如何使用 `get-object-retention`。

### AWS CLI

检索对象的对象保留配置

以下 `get-object-retention` 示例检索指定对象的对象保留配置。

```
aws s3api get-object-retention \  
  --bucket my-bucket-with-object-lock \  
  --key doc1.rtf
```

输出：

```
{  
  "Retention": {  
    "Mode": "GOVERNANCE",  
    "RetainUntilDate": "2025-01-01T00:00:00.000Z"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetObjectRetention](#)中的。

## get-public-access-block

以下代码示例演示了如何使用 `get-public-access-block`。

### AWS CLI

设置或修改存储桶的屏蔽公共访问权限配置

以下 `get-public-access-block` 示例显示了指定存储桶的屏蔽公共访问权限配置。

```
aws s3api get-public-access-block --bucket my-bucket
```

输出：

```
{
  "PublicAccessBlockConfiguration": {
    "IgnorePublicAcls": true,
    "BlockPublicPolicy": true,
    "BlockPublicAcls": true,
    "RestrictPublicBuckets": true
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPublicAccessBlock](#)中的。

## get-sites

以下代码示例演示了如何使用 get-sites。

### AWS CLI

获取您的网站

以下get-sites示例获取指定全球网络中的站点。

```
aws networkmanager get-sites \
  --global-network-id global-network-01231231231231231 \
  --region us-west-2
```

输出：

```
{
  "Sites": [
    {
      "SiteId": "site-444555aaabbb11223",
      "SiteArn": "arn:aws:networkmanager::123456789012:site/global-
network-01231231231231231/site-444555aaabbb11223",
      "GlobalNetworkId": "global-network-01231231231231231",
      "Description": "NY head office",
      "Location": {
        "Latitude": "40.7128",
        "Longitude": "-74.0060"
      },
      "CreatedAt": 1575554528.0,
      "State": "AVAILABLE"
    }
  ]
}
```

```
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSites](#)中的。

## get-transit-gateway-registrations

以下代码示例演示了如何使用 `get-transit-gateway-registrations`。

### AWS CLI

获取您的公网网关登记

以下`get-transit-gateway-registrations`示例获取注册到指定全球网络的传输网关。

```
aws networkmanager get-transit-gateway-registrations \  
  --global-network-id global-network-01231231231231231 \  
  --region us-west-2
```

输出：

```
{  
  "TransitGatewayRegistrations": [  
    {  
      "GlobalNetworkId": "global-network-01231231231231231",  
      "TransitGatewayArn": "arn:aws:ec2:us-west-2:123456789012:transit-  
gateway/tgw-123abc05e04123abc",  
      "State": {  
        "Code": "AVAILABLE"  
      }  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTransitGatewayRegistrations](#)中的。

## get-vpc-attachment

以下代码示例演示了如何使用 `get-vpc-attachment`。

## AWS CLI

### 获取 VPC 附件

以下 `get-vpc-attachment` 示例返回有关 VPC 连接的信息。

```
aws networkmanager get-vpc-attachment \  
  --attachment-id attachment-03b7ea450134787da
```

输出：

```
{  
  "VpcAttachment": {  
    "Attachment": {  
      "CoreNetworkId": "core-network-0522de1b226a5d7b3",  
      "AttachmentId": "attachment-03b7ea450134787da",  
      "OwnerAccountId": "987654321012",  
      "AttachmentType": "VPC",  
      "State": "CREATING",  
      "EdgeLocation": "us-east-1",  
      "ResourceArn": "arn:aws:ec2:us-east-1:987654321012:vpc/vpc-a7c4bbda",  
      "Tags": [  
        {  
          "Key": "Name",  
          "Value": "DevVPC"  
        }  
      ],  
      "CreatedAt": "2022-03-11T17:48:58+00:00",  
      "UpdatedAt": "2022-03-11T17:48:58+00:00"  
    },  
    "SubnetArns": [  
      "arn:aws:ec2:us-east-1:987654321012:subnet/subnet-202cde6c",  
      "arn:aws:ec2:us-east-1:987654321012:subnet/subnet-e5022dba",  
      "arn:aws:ec2:us-east-1:987654321012:subnet/subnet-2387ae02",  
      "arn:aws:ec2:us-east-1:987654321012:subnet/subnet-cda9dfc"  
    ],  
    "Options": {  
      "Ipv6Support": false  
    }  
  }  
}
```

有关更多信息，请参阅 Cloud WAN 用户指南中的[附件](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVpcAttachment](#)中的。

## list-bucket-analytics-configurations

以下代码示例演示了如何使用 `list-bucket-analytics-configurations`。

### AWS CLI

检索存储桶的分析配置列表

下面的 `list-bucket-analytics-configurations` 检索指定存储桶的分析配置列表。

```
aws s3api list-bucket-analytics-configurations \  
  --bucket my-bucket
```

输出：

```
{  
  "AnalyticsConfigurationList": [  
    {  
      "StorageClassAnalysis": {},  
      "Id": "1"  
    }  
  ],  
  "IsTruncated": false  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBucketAnalyticsConfigurations](#)中的。

## list-bucket-metrics-configurations

以下代码示例演示了如何使用 `list-bucket-metrics-configurations`。

### AWS CLI

检索存储桶的指标配置列表

以下`list-bucket-metrics-configurations`示例检索指定存储桶的指标配置列表。

```
aws s3api list-bucket-metrics-configurations \  
  --bucket my-bucket
```

输出：

```
{
  "IsTruncated": false,
  "MetricsConfigurationList": [
    {
      "Filter": {
        "Prefix": "logs"
      },
      "Id": "123"
    },
    {
      "Filter": {
        "Prefix": "tmp"
      },
      "Id": "234"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBucketMetricsConfigurations](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 list-tags-for-resource。

AWS CLI

列出资源的标签

以下list-tags-for-resource示例列出了指定设备资源 (device-07f6fd08867abc123) 的标签。

```
aws networkmanager list-tags-for-resource \
  --resource-arn arn:aws:networkmanager::123456789012:device/global-
network-01231231231231231231/device-07f6fd08867abc123 \
  --region us-west-2
```

输出：

```
{
  "TagList": [
```

```
    {
      "Key": "Network",
      "Value": "Northeast"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## put-bucket-metrics-configuration

以下代码示例演示了如何使用 `put-bucket-metrics-configuration`。

### AWS CLI

为存储桶设置指标配置

以下`put-bucket-metrics-configuration`示例为指定存储桶设置 ID 为 123 的指标配置。

```
aws s3api put-bucket-metrics-configuration \
  --bucket my-bucket \
  --id 123 \
  --metrics-configuration '{"Id": "123", "Filter": {"Prefix": "logs"}}'
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketMetricsConfiguration](#)中的。

## put-object-retention

以下代码示例演示了如何使用 `put-object-retention`。

### AWS CLI

为对象设置对象保留配置

以下 `put-object-retention` 示例为指定对象设置直到 2025 年 1 月 1 日的对象保留配置。

```
aws s3api put-object-retention \
  --bucket my-bucket-with-object-lock \
  --key doc1.rtf \
  --retention '{"Mode": "GOVERNANCE", "RetainUntilDate": "2025-01-01T00:00:00" }'
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutObjectRetention](#)中的。

## put-public-access-block

以下代码示例演示了如何使用 `put-public-access-block`。

### AWS CLI

为存储桶设置阻止公有访问配置

以下`put-public-access-block`示例为指定存储桶设置限制性封禁公共访问配置。

```
aws s3api put-public-access-block \  
  --bucket my-bucket \  
  --public-access-block-configuration  
  "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutPublicAccessBlock](#)中的。

## register-transit-gateway

以下代码示例演示了如何使用 `register-transit-gateway`。

### AWS CLI

在全球网络中注册传输网关

以下`register-transit-gateway`示例在指定的全球网络`tgw-123abc05e04123abc`中注册传输网关。

```
aws networkmanager register-transit-gateway \  
  --global-network-id global-network-01231231231231231 \  
  --transit-gateway-arn arn:aws:ec2:us-west-2:123456789012:transit-gateway/  
tgw-123abc05e04123abc \  
  --region us-west-2
```

输出：

```
{
  "TransitGatewayRegistration": {
    "GlobalNetworkId": "global-network-01231231231231231",
    "TransitGatewayArn": "arn:aws:ec2:us-west-2:123456789012:transit-gateway/tgw-123abc05e04123abc",
    "State": {
      "Code": "PENDING"
    }
  }
}
```

有关更多信息，请参阅 [Transit Gateway 网络管理器指南中的 Transit Gateway 注册](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [Register Transit Gateway](#) 中的。

## reject-attachment

以下代码示例演示了如何使用 reject-attachment。

### AWS CLI

#### 拒绝附件

以下 reject-attachment 示例拒绝了 VPC 连接请求。

```
aws networkmanager reject-attachment \
  --attachment-id attachment-03b7ea450134787da
```

输出：

```
{
  "Attachment": {
    "CoreNetworkId": "core-network-0522de1b226a5d7b3",
    "AttachmentId": "attachment-03b7ea450134787da",
    "OwnerAccountId": "987654321012",
    "AttachmentType": "VPC",
    "State": "AVAILABLE",
    "EdgeLocation": "us-east-1",
    "ResourceArn": "arn:aws:ec2:us-east-1:987654321012:vpc/vpc-a7c4bbda",
    "CreatedAt": "2022-03-11T17:48:58+00:00",
    "UpdatedAt": "2022-03-11T17:51:25+00:00"
  }
}
```

```
}
```

有关更多信息，请参阅 Cloud WAN 用户指南中的[附件接受](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RejectAttachment](#)中的。

## start-route-analysis

以下代码示例演示了如何使用 start-route-analysis。

### AWS CLI

#### 开始路径分析

以下 start-route-analysis 示例启动源和目标之间的分析，包括可选的 include-return-path。

```
aws networkmanager start-route-analysis \  
  --global-network-id global-network-00aa0aaa0b0aaa000 \  
  --source TransitGatewayAttachmentArn=arn:aws:ec2:us-east-1:503089527312:transit-  
gateway-attachment/tgw-attach-0d4a2d491bf68c093,IpAddress=10.0.0.0 \  
  --destination TransitGatewayAttachmentArn=arn:aws:ec2:us-  
west-1:503089527312:transit-gateway-attachment/tgw-  
attach-002577f30bb181742,IpAddress=11.0.0.0 \  
  --include-return-path
```

输出：

```
{  
  "RouteAnalysis": {  
    "GlobalNetworkId": "global-network-00aa0aaa0b0aaa000"  
    "OwnerId": "1111222233333",  
    "RouteAnalysisId": "a1873de1-273c-470c-1a2bc2345678",  
    "StartTimestamp": 1695760154.0,  
    "Status": "RUNNING",  
    "Source": {  
      "TransitGatewayAttachmentArn": "arn:aws:ec2:us-  
east-1:111122223333:transit-gateway-attachment/tgw-attach-1234567890abcdef0",  
      "TransitGatewayArn": "arn:aws:ec2:us-east-1:111122223333:transit-  
gateway/tgw-abcdef01234567890",  
      "IpAddress": "10.0.0.0"  
    },  
    "Destination": {
```

```
    "TransitGatewayAttachmentArn": "arn:aws:ec2:us-
west-1:555555555555:transit-gateway-attachment/tgw-attach-021345abcdef6789",
    "TransitGatewayArn": "arn:aws:ec2:us-west-1:111122223333:transit-
gateway/tgw-09876543210fedcba0",
    "IpAddress": "11.0.0.0"
  },
  "IncludeReturnPath": true,
  "UseMiddleboxes": false
}
}
```

有关更多信息，请参阅《AWS 全球公网网关用户指南》中的[路由分析器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartRouteAnalysis](#) 中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

对资源应用标签

以下 tag-resource 示例将标签应用 Network=Northeast 于设备 device-07f6fd08867abc123。

```
aws networkmanager tag-resource \
  --resource-arn arn:aws:networkmanager::123456789012:device/global-
network-01231231231231231231/device-07f6fd08867abc123 \
  --tags Key=Network,Value=Northeast \
  --region us-west-2
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

要从资源中删除标签

以下`untag-resource`示例Network从设备上删除带有密钥的标签`device-07f6fd08867abc123`。

```
aws networkmanager untag-resource \  
  --resource-arn arn:aws:networkmanager::123456789012:device/global-  
network-01231231231231231231/device-07f6fd08867abc123 ]  
  --tag-keys Network \  
  --region us-west-2
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-device

以下代码示例演示了如何使用 `update-device`。

### AWS CLI

#### 更新设备

以下`update-device`示例`device-07f6fd08867abc123`通过为设备指定站点 ID 来更新设备。

```
aws networkmanager update-device \  
  --global-network-id global-network-01231231231231231 \  
  --device-id device-07f6fd08867abc123 \  
  --site-id site-444555aaabbb11223 \  
  --region us-west-2
```

输出：

```
{  
  "Device": {  
    "DeviceId": "device-07f6fd08867abc123",  
    "DeviceArn": "arn:aws:networkmanager::123456789012:device/global-  
network-01231231231231231231/device-07f6fd08867abc123",  
    "GlobalNetworkId": "global-network-01231231231231231",  
    "Description": "NY office device",  
    "Type": "Office device",  
    "Vendor": "anycompany",  
    "Model": "abcabc",
```



```
    "SerialNumber": "1234",
    "SiteId": "site-444555aaabbb11223",
    "CreatedAt": 1575554005.0,
    "State": "UPDATING"
  }
}
```

有关更多信息，请参阅 [T ransit Gateway 网络管理器指南中的使用设备](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 UpdateDevice](#) 中的。

## update-global-network

以下代码示例演示了如何使用 `update-global-network`。

### AWS CLI

#### 更新全球网络

以下 `update-global-network` 示例更新了全球网络的描述 `global-network-01231231231231231`。

```
aws networkmanager update-global-network \
  --global-network-id global-network-01231231231231231 \
  --description "Head offices" \
  --region us-west-2
```

输出：

```
{
  "GlobalNetwork": {
    "GlobalNetworkId": "global-network-01231231231231231",
    "GlobalNetworkArn": "arn:aws:networkmanager::123456789012:global-network/global-network-01231231231231231",
    "Description": "Head offices",
    "CreatedAt": 1575553525.0,
    "State": "UPDATING"
  }
}
```

有关更多信息，请参阅 [T ransit Gateway 网络管理器指南中的全球网络](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateGlobalNetwork](#)中的。

## update-link

以下代码示例演示了如何使用 update-link。

### AWS CLI

#### 更新链接

以下update-link示例更新了链路的带宽信息link-11112222aaaabbbb1。

```
aws networkmanager update-link \  
  --global-network-id global-network-01231231231231231 \  
  --link-id link-11112222aaaabbbb1 \  
  --bandwidth UploadSpeed=20,DownloadSpeed=20 \  
  --region us-west-2
```

输出：

```
{  
  "Link": {  
    "LinkId": "link-11112222aaaabbbb1",  
    "LinkArn": "arn:aws:networkmanager::123456789012:link/global-  
network-01231231231231231/link-11112222aaaabbbb1",  
    "GlobalNetworkId": "global-network-01231231231231231",  
    "SiteId": "site-444555aaaabbb11223",  
    "Description": "VPN Link",  
    "Type": "broadband",  
    "Bandwidth": {  
      "UploadSpeed": 20,  
      "DownloadSpeed": 20  
    },  
    "Provider": "AnyCompany",  
    "CreatedAt": 1575555811.0,  
    "State": "UPDATING"  
  }  
}
```

有关更多信息，请参阅 [T ransit Gateway 网络管理器指南中的使用链接](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLink](#)中的。

## update-site

以下代码示例演示了如何使用 update-site。

### AWS CLI

#### 更新网站

以下update-site示例更新了指定全球网络site-444555aaabbb11223中站点的描述。

```
aws networkmanager update-site \  
  --global-network-id global-network-01231231231231231 \  
  --site-id site-444555aaabbb11223 \  
  --description "New York Office site" \  
  --region us-west-2
```

输出：

```
{  
  "Site": {  
    "SiteId": "site-444555aaabbb11223",  
    "SiteArn": "arn:aws:networkmanager::123456789012:site/global-  
network-01231231231231231/site-444555aaabbb11223",  
    "GlobalNetworkId": "global-network-01231231231231231",  
    "Description": "New York Office site",  
    "Location": {  
      "Latitude": "40.7128",  
      "Longitude": "-74.0060"  
    },  
    "CreatedAt": 1575554528.0,  
    "State": "UPDATING"  
  }  
}
```

有关更多信息，请参阅 [T ransit Gateway 网络管理器指南中的使用站点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateSite](#)中的。

## 使用灵活的工作室示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Nimble Studio 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## get-eula

以下代码示例演示了如何使用 get-eula。

AWS CLI

获取有关您的工作室的信息

以下get-eula示例列出了有关 EULA 的信息。

```
aws nimble get-eula \  
  --eula-id "EULAid"
```

输出：

```
{  
  "eula": {  
    "content": "https://www.mozilla.org/en-US/MPL/2.0/",  
    "createdAt": "2021-04-20T16:45:23+00:00",  
    "eulaId": "gJZLygd-Srq_5NNbSfiaLg",  
    "name": "Mozilla-FireFox",  
    "updatedAt": "2021-04-20T16:45:23+00:00"  
  }  
}
```

有关更多信息，请参阅 Amazon Nimble Studio 用户指南中的接受最终用户[许可协议](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetEula](#)中的。

## get-launch-profile-details

以下代码示例演示了如何使用 `get-launch-profile-details`。

### AWS CLI

列出可用的小部件

以下 `get-launch-profile-details` 示例列出了有关启动配置文件的详细信息。

```
aws nimble get-launch-profile-details \  
  --studio-id "StudioID" \  
  --launch-profile-id "LaunchProfileID"
```

输出：

```
{  
  "launchProfile": {  
    "arn": "arn:aws:nimble:us-west-2:123456789012:launch-profile/  
yeG7lDwNQEiwNTRT7DrV7Q",  
    "createdAt": "2022-01-27T21:18:59+00:00",  
    "createdBy": "AROA3002NEHCCYRNDDIFT:i-EXAMPLE11111",  
    "description": "The Launch Profile for the Render workers created by  
StudioBuilder.",  
    "ec2SubnetIds": [  
      "subnet-EXAMPLE11111"  
    ],  
    "launchProfileId": "yeG7lDwNQEiwNTRT7DrV7Q",  
    "launchProfileProtocolVersions": [  
      "2021-03-31"  
    ],  
    "name": "RenderWorker-Default",  
    "state": "READY",  
    "statusCode": "LAUNCH_PROFILE_CREATED",  
    "statusMessage": "Launch Profile has been created",  
    "streamConfiguration": {  
      "clipboardMode": "ENABLED",  
      "ec2InstanceTypes": [  
        "g4dn.4xlarge",  
        "g4dn.8xlarge"  
      ],  
      "maxSessionLengthInMinutes": 690,  
      "maxStoppedSessionLengthInMinutes": 0,  
      "streamingImageIds": [  

```

```
        "Cw_jXnp1QcSSXhE2hkNRoQ",
        "YGXAqgoWTnCNSV8VP20sHQ"
    ]
},
"studioComponentIds": [
    "_hR_-RaAReS0jAnLakbX7Q",
    "vQ5w_TbIRayPkAZgcbYRA",
    "ZQuMxN99Qfa_Js6ma9TwdA",
    "45Kj0SPPRzK20yvpCuQ6qw"
],
"tags": {
    "resourceArn": "arn:aws:nimble:us-west-2:123456789012:launch-profile/
yeG71DwNQEiwNTRT7DrV7Q"
},
"updatedAt": "2022-01-27T21:19:13+00:00",
"updatedBy": "AR0A3002NEHCCYRNDDIFT:i-00b98256b04d9e989",
"validationResults": [
    {
        "state": "VALIDATION_SUCCESS",
        "statusCode": "VALIDATION_SUCCESS",
        "statusMessage": "The validation succeeded.",
        "type": "VALIDATE_ACTIVE_DIRECTORY_STUDIO_COMPONENT"
    },
    {
        "state": "VALIDATION_SUCCESS",
        "statusCode": "VALIDATION_SUCCESS",
        "statusMessage": "The validation succeeded.",
        "type": "VALIDATE_SUBNET_ASSOCIATION"
    },
    {
        "state": "VALIDATION_SUCCESS",
        "statusCode": "VALIDATION_SUCCESS",
        "statusMessage": "The validation succeeded.",
        "type": "VALIDATE_NETWORK_ACL_ASSOCIATION"
    },
    {
        "state": "VALIDATION_SUCCESS",
        "statusCode": "VALIDATION_SUCCESS",
        "statusMessage": "The validation succeeded.",
        "type": "VALIDATE_SECURITY_GROUP_ASSOCIATION"
    }
]
},
"streamingImages": [
```

```
{
  "arn": "arn:aws:nimble:us-west-2:123456789012:streaming-image/
Cw_jXnp1QcSSXhE2hkNRoQ",
  "description": "Base windows image for NimbleStudio",
  "ec2ImageId": "ami-EXAMPLE11111",
  "eulaIds": [
    "gJZLygd-Srq_5NNbSfiaLg",
    "ggK2eIw6RQyt8PIee0lD3g",
    "a-D9Wc0VQCKUfxAinCDxaw",
    "RvoNmVXiSrS4LhLTb6ybkw",
    "wtp85BcSTa2NZeNRnMKdjw",
    "Rl-J0fM5S12hyIiwWIV6hw"
  ],
  "name": "NimbleStudioWindowsStreamImage",
  "owner": "amazon",
  "platform": "WINDOWS",
  "state": "READY",
  "streamingImageId": "Cw_jXnp1QcSSXhE2hkNRoQ",
  "tags": {
    "resourceArn": "arn:aws:nimble:us-west-2:123456789012:streaming-
image/Cw_jXnp1QcSSXhE2hkNRoQ"
  }
},
{
  "arn": "arn:aws:nimble:us-west-2:123456789012:streaming-image/
YGXAqgoWTnCNSV8VP20sHQ",
  "description": "Base linux image for NimbleStudio",
  "ec2ImageId": "ami-EXAMPLE11111",
  "eulaIds": [
    "gJZLygd-Srq_5NNbSfiaLg",
    "ggK2eIw6RQyt8PIee0lD3g",
    "a-D9Wc0VQCKUfxAinCDxaw",
    "RvoNmVXiSrS4LhLTb6ybkw",
    "wtp85BcSTa2NZeNRnMKdjw",
    "Rl-J0fM5S12hyIiwWIV6hw"
  ],
  "name": "NimbleStudioLinuxStreamImage",
  "owner": "amazon",
  "platform": "LINUX",
  "state": "READY",
  "streamingImageId": "YGXAqgoWTnCNSV8VP20sHQ",
  "tags": {
    "resourceArn": "arn:aws:nimble:us-west-2:123456789012:streaming-
image/YGXAqgoWTnCNSV8VP20sHQ"
  }
}
```

```
    }
  }
],
"studioComponentSummaries": [
  {
    "description": "FSx for Windows",
    "name": "FSxWindows",
    "studioComponentId": "ZQuMxN99Qfa_Js6ma9TwdA",
    "subtype": "AMAZON_FSX_FOR_WINDOWS",
    "type": "SHARED_FILE_SYSTEM"
  },
  {
    "description": "Instance configuration studio component.",
    "name": "InstanceConfiguration",
    "studioComponentId": "vQ5w_TbIRayPkAZgcbYRA",
    "subtype": "CUSTOM",
    "type": "CUSTOM"
  },
  {
    "name": "ActiveDirectory",
    "studioComponentId": "_hR_-RaAReS0jAnLakbX7Q",
    "subtype": "AWS_MANAGED_MICROSOFT_AD",
    "type": "ACTIVE_DIRECTORY"
  },
  {
    "description": "Render farm running Deadline",
    "name": "RenderFarm",
    "studioComponentId": "45Kj0SPPRzK20yvpCuQ6qw",
    "subtype": "CUSTOM",
    "type": "COMPUTE_FARM"
  }
]
}
```

有关更多信息，请参阅 Amazon Nimble Studio 用户指南中的[创建启动配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetLaunchProfileDetails](#)中的。

## get-launch-profile

以下代码示例演示了如何使用 get-launch-profile。



## AWS CLI

### 列出可用的小部件

以下`get-launch-profile`示例列出了有关启动配置文件的信息。

```
aws nimble get-launch-profile \  
  --studio-id "StudioID" \  
  --launch-profile-id "LaunchProfileID"
```

输出：

```
{  
  "launchProfile": {  
    "arn": "arn:aws:nimble:us-west-2:123456789012:launch-profile/  
yeG71DwNQEiwNTRT7DrV7Q",  
    "createdAt": "2022-01-27T21:18:59+00:00",  
    "createdBy": "AROA3002NEHCCYRNDDIFT:i-EXAMPLE11111",  
    "description": "The Launch Profile for the Render workers created by  
StudioBuilder.",  
    "ec2SubnetIds": [  
      "subnet-EXAMPLE11111"  
    ],  
    "launchProfileId": "yeG71DwNQEiwNTRT7DrV7Q",  
    "launchProfileProtocolVersions": [  
      "2021-03-31"  
    ],  
    "name": "RenderWorker-Default",  
    "state": "READY",  
    "statusCode": "LAUNCH_PROFILE_CREATED",  
    "statusMessage": "Launch Profile has been created",  
    "streamConfiguration": {  
      "clipboardMode": "ENABLED",  
      "ec2InstanceTypes": [  
        "g4dn.4xlarge",  
        "g4dn.8xlarge"  
      ],  
      "maxSessionLengthInMinutes": 690,  
      "maxStoppedSessionLengthInMinutes": 0,  
      "streamingImageIds": [  
        "Cw_jXnp1QcSSXhE2hkNRoQ",  
        "YGXAqgoWTnCNSV8VP20sHQ"  
      ]  
    }  
  }  
}
```

```
  },
  "studioComponentIds": [
    "_hR_-RaAReS0jAnLakbX7Q",
    "vQ5w_TbIRayPkAZgcbyYRA",
    "ZQuMxN99Qfa_Js6ma9TwdA",
    "45Kj0SPPRzK20yvpCuQ6qw"
  ],
  "tags": {},
  "updatedAt": "2022-01-27T21:19:13+00:00",
  "updatedBy": "AR0A3002NEHCCYRNDDIFT:i-00b98256b04d9e989",
  "validationResults": [
    {
      "state": "VALIDATION_SUCCESS",
      "statusCode": "VALIDATION_SUCCESS",
      "statusMessage": "The validation succeeded.",
      "type": "VALIDATE_ACTIVE_DIRECTORY_STUDIO_COMPONENT"
    },
    {
      "state": "VALIDATION_SUCCESS",
      "statusCode": "VALIDATION_SUCCESS",
      "statusMessage": "The validation succeeded.",
      "type": "VALIDATE_SUBNET_ASSOCIATION"
    },
    {
      "state": "VALIDATION_SUCCESS",
      "statusCode": "VALIDATION_SUCCESS",
      "statusMessage": "The validation succeeded.",
      "type": "VALIDATE_NETWORK_ACL_ASSOCIATION"
    },
    {
      "state": "VALIDATION_SUCCESS",
      "statusCode": "VALIDATION_SUCCESS",
      "statusMessage": "The validation succeeded.",
      "type": "VALIDATE_SECURITY_GROUP_ASSOCIATION"
    }
  ]
}
```

有关更多信息，请参阅 Amazon Nimble Studio 用户指南中的[创建启动配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetLaunchProfile](#)中的。

## get-studio

以下代码示例演示了如何使用 get-studio。

### AWS CLI

获取有关您的工作室的信息

以下get-studio示例列出了您 AWS 账户中的工作室。

```
aws nimble get-studio \  
  --studio-id "StudioID"
```

输出：

```
{  
  "studio": {  
    "adminRoleArn": "arn:aws:iam::123456789012:role/studio-admin-role",  
    "arn": "arn:aws:nimble:us-west-2:123456789012:studio/stid-EXAMPLE11111",  
    "createdAt": "2022-01-27T20:29:35+00:00",  
    "displayName": "studio-name",  
    "homeRegion": "us-west-2",  
    "ssoClientId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "state": "READY",  
    "statusCode": "STUDIO_CREATED",  
    "statusMessage": "The studio has been created successfully ",  
    "studioEncryptionConfiguration": {  
      "keyType": "AWS_OWNED_KEY"  
    },  
    "studioId": "us-west-2:stid-EXAMPLE11111",  
    "studioName": "studio-name",  
    "studioUrl": "https://studio-name.nimblestudio.us-west-2.amazonaws.com",  
    "tags": {},  
    "updatedAt": "2022-01-27T20:29:37+00:00",  
    "userRoleArn": "arn:aws:iam::123456789012:role/studio-user-role"  
  }  
}
```

有关更多信息，请参阅[什么是 Amazon Nimble Studio](#)？在 Amazon Nimble Studio 用户指南中。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetStudio](#)中的。

## list-eula-acceptances

以下代码示例演示了如何使用 `list-eula-acceptances`。

### AWS CLI

列出可用的小部件

以下 `list-eula-acceptances` 示例列出了您 AWS 账户中接受的 EULA。

```
aws nimble list-eula-acceptances \  
  --studio-id "StudioID"
```

输出：

```
{  
  "eulaAcceptances": [  
    {  
      "acceptedAt": "2022-01-28T17:44:35+00:00",  
      "acceptedBy": "92677b4b19-e9fd012a-94ad-4f16-9866-c69a63ab6486",  
      "accepteeId": "us-west-2:stid-nyoqq12fteqy1x48",  
      "eulaAcceptanceId": "V0JlpZQaSx6yHcUuX0qfQw",  
      "eulaId": "Rl-J0fM5S12hyIiwWIV6hw"  
    },  
    {  
      "acceptedAt": "2022-01-28T17:44:35+00:00",  
      "acceptedBy": "92677b4b19-e9fd012a-94ad-4f16-9866-c69a63ab6486",  
      "accepteeId": "us-west-2:stid-nyoqq12fteqy1x48",  
      "eulaAcceptanceId": "YY_uDFW-SVibc627qbug0Q",  
      "eulaId": "RvoNmVXiSrS4LhLTb6ybkw"  
    },  
    {  
      "acceptedAt": "2022-01-28T17:44:35+00:00",  
      "acceptedBy": "92677b4b19-e9fd012a-94ad-4f16-9866-c69a63ab6486",  
      "accepteeId": "us-west-2:stid-nyoqq12fteqy1x48",  
      "eulaAcceptanceId": "ov087PnhQ4-MpttiL5uN6Q",  
      "eulaId": "a-D9Wc0VQCKUfxAinCDxaw"  
    },  
    {  
      "acceptedAt": "2022-01-28T17:44:35+00:00",  
      "acceptedBy": "92677b4b19-e9fd012a-94ad-4f16-9866-c69a63ab6486",  
      "accepteeId": "us-west-2:stid-nyoqq12fteqy1x48",  
      "eulaAcceptanceId": "5YeXje4yR0amuTESGvqIAQ",  
      "eulaId": "gJZLygd-Srq_5NNbSfiaLg"  
    }  
  ]  
}
```

```

    },
    {
      "acceptedAt": "2022-01-28T17:44:35+00:00",
      "acceptedBy": "92677b4b19-e9fd012a-94ad-4f16-9866-c69a63ab6486",
      "accepteeId": "us-west-2:stid-nyoqq12fteqy1x48",
      "eulaAcceptanceId": "W1sIn8PtScqeJEn8sxxhgw",
      "eulaId": "ggK2eIw6RQyt8PIee01D3g"
    },
    {
      "acceptedAt": "2022-01-28T17:44:35+00:00",
      "acceptedBy": "92677b4b19-e9fd012a-94ad-4f16-9866-c69a63ab6486",
      "accepteeId": "us-west-2:stid-nyoqq12fteqy1x48",
      "eulaAcceptanceId": "Zq9KNEQPRMWJ7FolSoQgUA",
      "eulaId": "wtp85BcSTa2NZeNRnMKdjw"
    }
  ]
}

```

有关更多信息，请参阅 Amazon Nimble Studio 用户指南中的接受最终用户[许可协议](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListEulaAcceptances](#)中的。

## list-eulas

以下代码示例演示了如何使用 list-eulas。

### AWS CLI

列出可用的小部件

以下 list-eulas 示例列出了您 AWS 账户中的最终用户许可协议。

```
aws nimble list-eulas
```

输出：

```

{
  "eulas": [
    {
      "content": "https://www.mozilla.org/en-US/MPL/2.0/",
      "createdAt": "2021-04-20T16:45:23+00:00",
      "eulaId": "gJZLygd-Srq_5NNbSfiaLg",
      "name": "Mozilla-FireFox",

```

```
    "updatedAt": "2021-04-20T16:45:23+00:00"
  },
  {
    "content": "https://www.awsthinkbox.com/end-user-license-agreement",
    "createdAt": "2021-04-20T16:45:24+00:00",
    "eulaId": "RvoNmVXiSrS4LhLTb6ybkw",
    "name": "Thinkbox-Deadline",
    "updatedAt": "2021-04-20T16:45:24+00:00"
  },
  {
    "content": "https://www.videolan.org/legal.html",
    "createdAt": "2021-04-20T16:45:24+00:00",
    "eulaId": "R1-J0fM5S12hyIiwWIV6hw",
    "name": "Videolan-VLC",
    "updatedAt": "2021-04-20T16:45:24+00:00"
  },
  {
    "content": "https://code.visualstudio.com/license",
    "createdAt": "2021-04-20T16:45:23+00:00",
    "eulaId": "ggK2eIw6RQyt8PIee0lD3g",
    "name": "Microsoft-VSCode",
    "updatedAt": "2021-04-20T16:45:23+00:00"
  },
  {
    "content": "https://darbyjohnston.github.io/DJV/legal.html#License",
    "createdAt": "2021-04-20T16:45:23+00:00",
    "eulaId": "wtp85BcSTa2NZeNRnMKdjw",
    "name": "DJV-DJV",
    "updatedAt": "2021-04-20T16:45:23+00:00"
  },
  {
    "content": "https://www.sidefx.com/legal/license-agreement/",
    "createdAt": "2021-04-20T16:45:24+00:00",
    "eulaId": "uu2VDLo-QJeIGWwLBae_UA",
    "name": "SideFX-Houdini",
    "updatedAt": "2021-04-20T16:45:24+00:00"
  },
  {
    "content": "https://www.chaosgroup.com/eula",
    "createdAt": "2021-04-20T16:45:23+00:00",
    "eulaId": "L0HS4P3CRYKVXc2J2L07Vw",
    "name": "ChaosGroup-Vray",
    "updatedAt": "2021-04-20T16:45:23+00:00"
  },
}
```

```
{
  "content": "https://www.foundry.com/eula",
  "createdAt": "2021-04-20T16:45:23+00:00",
  "eulaId": "SAuhfHmMSAeUuq3wsMiMlw",
  "name": "Foundry-Nuke",
  "updatedAt": "2021-04-20T16:45:23+00:00"
},
{
  "content": "https://download.blender.org/release/GPL3-license.txt",
  "createdAt": "2021-04-20T16:45:23+00:00",
  "eulaId": "a-D9Wc0VQCKUfxAinCDxaw",
  "name": "BlenderFoundation-Blender",
  "updatedAt": "2021-04-20T16:45:23+00:00"
}
]
```

有关更多信息，请参阅 Amazon Nimble Studio 用户指南中的接受最终用户 [许可协议](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListEulas](#) 中的。

## list-launch-profiles

以下代码示例演示了如何使用 `list-launch-profiles`。

### AWS CLI

列出可用的小部件

以下 `list-launch-profiles` 示例列出了您 AWS 账户中的启动配置文件。

```
aws nimble list-launch-profiles \
  --studio-id "StudioID"
```

输出：

```
{
  "launchProfiles": [
    {
      "arn": "arn:aws:nimble:us-west-2:123456789012:launch-profile/
yeG7lDwNQEiwNTRT7DrV7Q",
      "createdAt": "2022-01-27T21:18:59+00:00",
      "createdBy": "AROA3002NEHCCYRNDIFT:i-EXAMPLE11111",

```

```
"description": "The Launch Profile for the Render workers created by
StudioBuilder.",
"ec2SubnetIds": [
  "subnet-EXAMPLE11111"
],
"launchProfileId": "yeG7lDwNQEiwNTRT7DrV7Q",
"launchProfileProtocolVersions": [
  "2021-03-31"
],
"name": "RenderWorker-Default",
"state": "READY",
"statusCode": "LAUNCH_PROFILE_CREATED",
"statusMessage": "Launch Profile has been created",
"streamConfiguration": {
  "clipboardMode": "ENABLED",
  "ec2InstanceTypes": [
    "g4dn.4xlarge",
    "g4dn.8xlarge"
  ],
  "maxSessionLengthInMinutes": 690,
  "maxStoppedSessionLengthInMinutes": 0,
  "streamingImageIds": [
    "Cw_jXnp1QcSSXhE2hkNRoQ",
    "YGXAqgoWTnCNSV8VP20sHQ"
  ]
},
"studioComponentIds": [
  "_hR_-RaAReS0jAnLakbX7Q",
  "vQ5w_TbIRayPkAZgcbyYRA",
  "ZQuMxN99Qfa_Js6ma9TwdA",
  "45Kj0SPPrzK20yvpCuQ6qw"
],
"tags": {},
"updatedAt": "2022-01-27T21:19:13+00:00",
"updatedBy": "AROA3002NEHCCYRNDDIFT:i-EXAMPLE11111",
"validationResults": [
  {
    "state": "VALIDATION_SUCCESS",
    "statusCode": "VALIDATION_SUCCESS",
    "statusMessage": "The validation succeeded.",
    "type": "VALIDATE_ACTIVE_DIRECTORY_STUDIO_COMPONENT"
  },
  {
    "state": "VALIDATION_SUCCESS",
```



```
        "statusCode": "VALIDATION_SUCCESS",
        "statusMessage": "The validation succeeded.",
        "type": "VALIDATE_SUBNET_ASSOCIATION"
    },
    {
        "state": "VALIDATION_SUCCESS",
        "statusCode": "VALIDATION_SUCCESS",
        "statusMessage": "The validation succeeded.",
        "type": "VALIDATE_NETWORK_ACL_ASSOCIATION"
    },
    {
        "state": "VALIDATION_SUCCESS",
        "statusCode": "VALIDATION_SUCCESS",
        "statusMessage": "The validation succeeded.",
        "type": "VALIDATE_SECURITY_GROUP_ASSOCIATION"
    }
]
},
{
    "arn": "arn:aws:nimble:us-west-2:123456789012:launch-profile/
jDCIm1jRSaa9e44PZ3w7gg",
    "createdAt": "2022-01-27T21:19:26+00:00",
    "createdBy": "AROA3002NEHCCYRNDDIFT:i-EXAMPLE11111",
    "description": "This Workstation Launch Profile was created by
StudioBuilder",
    "ec2SubnetIds": [
        "subnet-046f4205ae535b2cc"
    ],
    "launchProfileId": "jDCIm1jRSaa9e44PZ3w7gg",
    "launchProfileProtocolVersions": [
        "2021-03-31"
    ],
    "name": "Workstation-Default",
    "state": "READY",
    "statusCode": "LAUNCH_PROFILE_CREATED",
    "statusMessage": "Launch Profile has been created",
    "streamConfiguration": {
        "clipboardMode": "ENABLED",
        "ec2InstanceTypes": [
            "g4dn.4xlarge",
            "g4dn.8xlarge"
        ],
    },
    "maxSessionLengthInMinutes": 690,
    "maxStoppedSessionLengthInMinutes": 0,
```

```
    "streamingImageIds": [
      "Cw_jXnp1QcSSXhE2hkNRoQ",
      "YGXAqgoWTnCNSV8VP20sHQ"
    ]
  },
  "studioComponentIds": [
    "_hR_-RaAReS0jAnLakbX7Q",
    "vQ5w_TbIRayPkAZgcbyYRA",
    "ZQuMxN99Qfa_Js6ma9TwdA",
    "yJSbsHXAQYwk9FXLNusX1Q",
    "45Kj0SPPrzK20yvpCuQ6qw"
  ],
  "tags": {},
  "updatedAt": "2022-01-27T21:19:40+00:00",
  "updatedBy": "AROA3002NEHCCYRNDDIFT:i-EXAMPLE11111",
  "validationResults": [
    {
      "state": "VALIDATION_SUCCESS",
      "statusCode": "VALIDATION_SUCCESS",
      "statusMessage": "The validation succeeded.",
      "type": "VALIDATE_ACTIVE_DIRECTORY_STUDIO_COMPONENT"
    },
    {
      "state": "VALIDATION_SUCCESS",
      "statusCode": "VALIDATION_SUCCESS",
      "statusMessage": "The validation succeeded.",
      "type": "VALIDATE_SUBNET_ASSOCIATION"
    },
    {
      "state": "VALIDATION_SUCCESS",
      "statusCode": "VALIDATION_SUCCESS",
      "statusMessage": "The validation succeeded.",
      "type": "VALIDATE_NETWORK_ACL_ASSOCIATION"
    },
    {
      "state": "VALIDATION_SUCCESS",
      "statusCode": "VALIDATION_SUCCESS",
      "statusMessage": "The validation succeeded.",
      "type": "VALIDATE_SECURITY_GROUP_ASSOCIATION"
    }
  ]
}
```

```
}
```

有关更多信息，请参阅 Amazon Nimble Studio 用户指南中的[创建启动配置文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListLaunchProfiles](#)中的。

## list-studio-components

以下代码示例演示了如何使用 list-studio-components。

### AWS CLI

列出可用的小部件

以下 list-studio-components 示例列出了您 AWS 账户中的工作室组件。

```
aws nimble list-studio-components \  
  --studio-id "StudioID"
```

输出：

```
{  
  "studioComponents": [  
    {  
      "arn": "arn:aws:nimble:us-west-2:123456789012:studio-component/  
ZQuMxN99Qfa_Js6ma9TwdA",  
      "configuration": {  
        "sharedFileSystemConfiguration": {  
          "fileSystemId": "fs-EXAMPLE11111",  
          "linuxMountPoint": "/mnt/fsxshare",  
          "shareName": "share",  
          "windowsMountDrive": "Z"  
        }  
      },  
      "createdAt": "2022-01-27T21:15:34+00:00",  
      "createdBy": "AROA3002NEHCCYRNDDIFT:i-EXAMPLE11111",  
      "description": "FSx for Windows",  
      "ec2SecurityGroupIds": [  
        "sg-EXAMPLE11111"  
      ],  
      "name": "FSxWindows",  
      "state": "READY",  
      "statusCode": "STUDIO_COMPONENT_CREATED",  
    }  
  ]  
}
```

```
    "statusMessage": "Studio Component has been created",
    "studioComponentId": "ZQuMxN99Qfa_Js6ma9TwdA",
    "subtype": "AMAZON_FSX_FOR_WINDOWS",
    "tags": {},
    "type": "SHARED_FILE_SYSTEM",
    "updatedAt": "2022-01-27T21:15:35+00:00",
    "updatedBy": "AROA3002NEHCCYRNDDIFT:i-EXAMPLE11111"
  },
  ...
}
```

有关更多信息，请参阅亚马逊 [Nimble Studio 用户指南中的 StudioBuilder 如何使用亚马逊 Nimble Studio](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListStudioComponents](#) 中的。

## list-studio-members

以下代码示例演示了如何使用 list-studio-members。

### AWS CLI

列出可用的小部件

以下 list-studio-members 示例列出了您 AWS 账户中可用的工作室成员。

```
aws nimble list-studio-members \
  --studio-id "StudioID"
```

输出：

```
{
  "members": [
    {
      "identityStoreId": "d-EXAMPLE11111",
      "persona": "ADMINISTRATOR",
      "principalId": "EXAMPLE11111-e9fd012a-94ad-4f16-9866-c69a63ab6486"
    }
  ]
}
```

有关更多信息，请参阅 Amazon Nimble [Studio 用户指南中的添加工作室用户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStudioMembers](#)中的。

## list-studios

以下代码示例演示了如何使用 list-studios。

### AWS CLI

列出您的工作室

以下list-studios示例列出了您 AWS 账户中的工作室。

```
aws nimble list-studios
```

输出：

```
{
  "studios": [
    {
      "adminRoleArn": "arn:aws:iam::123456789012:role/studio-admin-role",
      "arn": "arn:aws:nimble:us-west-2:123456789012:studio/studio-id",
      "createdAt": "2022-01-27T20:29:35+00:00",
      "displayName": "studio-name",
      "homeRegion": "us-west-2",
      "ssoClientId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "state": "READY",
      "statusCode": "STUDIO_CREATED",
      "statusMessage": "The studio has been created successfully ",
      "studioEncryptionConfiguration": {
        "keyType": "AWS_OWNED_KEY"
      },
      "studioId": "us-west-2:studio-id",
      "studioName": "studio-name",
      "studioUrl": "https://studio-name.nimblestudio.us-west-2.amazonaws.com",
      "tags": {},
      "updatedAt": "2022-01-27T20:29:37+00:00",
      "userRoleArn": "arn:aws:iam::123456789012:role/studio-user-role"
    }
  ]
}
```

有关更多信息，请参阅[什么是 Amazon Nimble Studio](#)？在 Amazon Nimble Studio 用户指南中。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListStudios](#)中的。

## OpenSearch 使用的服务示例 AWS CLI

以下代码示例向您展示了如何使用 with S OpenSearch ervice 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-elasticsearch-domain**

以下代码示例演示了如何使用 create-elasticsearch-domain。

AWS CLI

创建 Amazon Elasticsearch Service 域名

以下create-elasticsearch-domain命令在 VPC 内创建一个新的 Amazon Elasticsearch Service 域并限制单个用户的访问权限。Amazon ES 根据指定的子网和安全组 ID 推断出 VPC ID。

```
aws es create-elasticsearch-domain \  
  --domain-name vpc-cli-example \  
  --elasticsearch-version 6.2 \  
  --elasticsearch-cluster-config  
  InstanceType=m4.large.elasticsearch,InstanceCount=1 \  
  --ebs-options EBSEnabled=true,VolumeType=standard,VolumeSize=10 \  
  --access-policies '{"Version": "2012-10-17", "Statement": [ { "Effect":  
"Allow", "Principal": { "AWS": "arn:aws:iam::123456789012:root" }, "Action": "es:*",  
"Resource": "arn:aws:es:us-west-1:123456789012:domain/vpc-cli-example/*" } ] }' \  
  --vpc-options SubnetIds=subnet-1a2a3a4a,SecurityGroupIds=sg-2a3a4a5a
```

输出：

```
{
  "DomainStatus": {
    "ElasticsearchClusterConfig": {
      "DedicatedMasterEnabled": false,
      "InstanceCount": 1,
      "ZoneAwarenessEnabled": false,
      "InstanceType": "m4.large.elasticsearch"
    },
    "DomainId": "123456789012/vpc-cli-example",
    "CognitoOptions": {
      "Enabled": false
    },
    "VPCOptions": {
      "SubnetIds": [
        "subnet-1a2a3a4a"
      ],
      "VPCId": "vpc-3a4a5a6a",
      "SecurityGroupIds": [
        "sg-2a3a4a5a"
      ],
      "AvailabilityZones": [
        "us-west-1c"
      ]
    },
    "Created": true,
    "Deleted": false,
    "EBSOptions": {
      "VolumeSize": 10,
      "VolumeType": "standard",
      "EBSEnabled": true
    },
    "Processing": true,
    "DomainName": "vpc-cli-example",
    "SnapshotOptions": {
      "AutomatedSnapshotStartHour": 0
    },
    "ElasticsearchVersion": "6.2",
    "AccessPolicies": "{\n\"Version\": \"2012-10-17\", \"Statement\": [{\n\"Effect\":\n\n\"Allow\", \"Principal\": {\n\"AWS\": \"arn:aws:iam::123456789012:root\"}, \"Action\":\n\n\"es:*\", \"Resource\": \"arn:aws:es:us-west-1:123456789012:domain/vpc-cli-example/*\n\n\"}]}",
    "AdvancedOptions": {
```

```
        "rest.action.multi.allow_explicit_index": "true"
    },
    "EncryptionAtRestOptions": {
        "Enabled": false
    },
    "ARN": "arn:aws:es:us-west-1:123456789012:domain/vpc-cli-example"
}
}
```

有关更多信息，请参阅[亚马逊 Elasticsearch Service 开发者指南中的创建和管理亚马逊 Elasticsearch 服务域](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateElasticsearchDomain](#) 中的。

## describe-elasticsearch-domain-config

以下代码示例演示了如何使用 describe-elasticsearch-domain-config。

### AWS CLI

获取域配置详细信息

以下 describe-elasticsearch-domain-config 示例提供了给定域的配置详细信息以及每个域组件的状态信息。

```
aws es describe-elasticsearch-domain-config \
  --domain-name cli-example
```

输出：

```
{
  "DomainConfig": {
    "ElasticsearchVersion": {
      "Options": "7.4",
      "Status": {
        "CreationDate": 1589395034.946,
        "UpdateDate": 1589395827.325,
        "UpdateVersion": 8,
        "State": "Active",
        "PendingDeletion": false
      }
    },
    "ElasticsearchClusterConfig": {
```



```
    "Options": {
      "InstanceType": "c5.large.elasticsearch",
      "InstanceCount": 1,
      "DedicatedMasterEnabled": true,
      "ZoneAwarenessEnabled": false,
      "DedicatedMasterType": "c5.large.elasticsearch",
      "DedicatedMasterCount": 3,
      "WarmEnabled": true,
      "WarmType": "ultrawarm1.medium.elasticsearch",
      "WarmCount": 2
    },
    "Status": {
      "CreationDate": 1589395034.946,
      "UpdateDate": 1589395827.325,
      "UpdateVersion": 8,
      "State": "Active",
      "PendingDeletion": false
    }
  },
  "EBSOptions": {
    "Options": {
      "EBSEnabled": true,
      "VolumeType": "gp2",
      "VolumeSize": 10
    },
    "Status": {
      "CreationDate": 1589395034.946,
      "UpdateDate": 1589395827.325,
      "UpdateVersion": 8,
      "State": "Active",
      "PendingDeletion": false
    }
  },
  "AccessPolicies": {
    "Options": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"*\"},\"Action\":\"es:*\",\"Resource\":\"arn:aws:es:us-east-1:123456789012:domain/cli-example/*\"}]}",
    "Status": {
      "CreationDate": 1589395034.946,
      "UpdateDate": 1589395827.325,
      "UpdateVersion": 8,
      "State": "Active",
      "PendingDeletion": false
    }
  }
}
```

```
    },
    "SnapshotOptions": {
      "Options": {
        "AutomatedSnapshotStartHour": 0
      },
      "Status": {
        "CreationDate": 1589395034.946,
        "UpdateDate": 1589395827.325,
        "UpdateVersion": 8,
        "State": "Active",
        "PendingDeletion": false
      }
    },
    "VPCOptions": {
      "Options": {},
      "Status": {
        "CreationDate": 1591210426.162,
        "UpdateDate": 1591210426.162,
        "UpdateVersion": 18,
        "State": "Active",
        "PendingDeletion": false
      }
    },
    "CognitoOptions": {
      "Options": {
        "Enabled": false
      },
      "Status": {
        "CreationDate": 1591210426.163,
        "UpdateDate": 1591210426.163,
        "UpdateVersion": 18,
        "State": "Active",
        "PendingDeletion": false
      }
    },
    "EncryptionAtRestOptions": {
      "Options": {
        "Enabled": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/1a2a3a4a-1a2a-1a2a-1a2a-1a2a3a4a5a6a"
      },
      "Status": {
        "CreationDate": 1589395034.946,
        "UpdateDate": 1589395827.325,
```

```
        "UpdateVersion": 8,
        "State": "Active",
        "PendingDeletion": false
    }
},
"NodeToNodeEncryptionOptions": {
    "Options": {
        "Enabled": true
    },
    "Status": {
        "CreationDate": 1589395034.946,
        "UpdateDate": 1589395827.325,
        "UpdateVersion": 8,
        "State": "Active",
        "PendingDeletion": false
    }
},
"AdvancedOptions": {
    "Options": {
        "rest.action.multi.allow_explicit_index": "true"
    },
    "Status": {
        "CreationDate": 1589395034.946,
        "UpdateDate": 1589395827.325,
        "UpdateVersion": 8,
        "State": "Active",
        "PendingDeletion": false
    }
},
"LogPublishingOptions": {
    "Options": {},
    "Status": {
        "CreationDate": 1591210426.164,
        "UpdateDate": 1591210426.164,
        "UpdateVersion": 18,
        "State": "Active",
        "PendingDeletion": false
    }
},
"DomainEndpointOptions": {
    "Options": {
        "EnforceHTTPS": true,
        "TLSSecurityPolicy": "Policy-Min-TLS-1-0-2019-07"
    },
}
```

```
    "Status": {
      "CreationDate": 1589395034.946,
      "UpdateDate": 1589395827.325,
      "UpdateVersion": 8,
      "State": "Active",
      "PendingDeletion": false
    }
  },
  "AdvancedSecurityOptions": {
    "Options": {
      "Enabled": true,
      "InternalUserDatabaseEnabled": true
    },
    "Status": {
      "CreationDate": 1589395034.946,
      "UpdateDate": 1589827485.577,
      "UpdateVersion": 14,
      "State": "Active",
      "PendingDeletion": false
    }
  }
}
```

有关更多信息，请参阅[亚马逊 Elasticsearch Service 开发者指南中的创建和管理亚马逊 Elasticsearch 服务域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeElasticsearchDomainConfig](#)中的。

## describe-elasticsearch-domain

以下代码示例演示了如何使用 describe-elasticsearch-domain。

### AWS CLI

获取单个域名的详细信息

以下describe-elasticsearch-domain示例提供了给定域的配置详细信息。

```
aws es describe-elasticsearch-domain \
  --domain-name cli-example
```

输出：

```

{
  "DomainStatus": {
    "DomainId": "123456789012/cli-example",
    "DomainName": "cli-example",
    "ARN": "arn:aws:es:us-east-1:123456789012:domain/cli-example",
    "Created": true,
    "Deleted": false,
    "Endpoint": "search-cli-example-1a2a3a4a5a6a7a8a9a0a.us-
east-1.es.amazonaws.com",
    "Processing": false,
    "UpgradeProcessing": false,
    "ElasticsearchVersion": "7.4",
    "ElasticsearchClusterConfig": {
      "InstanceType": "c5.large.elasticsearch",
      "InstanceCount": 1,
      "DedicatedMasterEnabled": true,
      "ZoneAwarenessEnabled": false,
      "DedicatedMasterType": "c5.large.elasticsearch",
      "DedicatedMasterCount": 3,
      "WarmEnabled": true,
      "WarmType": "ultrawarm1.medium.elasticsearch",
      "WarmCount": 2
    },
    "EBSOptions": {
      "EBSEnabled": true,
      "VolumeType": "gp2",
      "VolumeSize": 10
    },
    "AccessPolicies": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"*\"},\"Action\":\"es:*\",\"Resource\":\"arn:aws:es:us-east-1:123456789012:domain/cli-example/*\"}]}",
    "SnapshotOptions": {
      "AutomatedSnapshotStartHour": 0
    },
    "CognitoOptions": {
      "Enabled": false
    },
    "EncryptionAtRestOptions": {
      "Enabled": true,
      "KmsKeyId": "arn:aws:kms:us-
east-1:123456789012:key/1a2a3a4a-1a2a-1a2a-1a2a-1a2a3a4a5a6a"
    },
    "NodeToNodeEncryptionOptions": {

```

```
    "Enabled": true
  },
  "AdvancedOptions": {
    "rest.action.multi.allow_explicit_index": "true"
  },
  "ServiceSoftwareOptions": {
    "CurrentVersion": "R20200522",
    "NewVersion": "",
    "UpdateAvailable": false,
    "Cancellable": false,
    "UpdateStatus": "COMPLETED",
    "Description": "There is no software update available for this domain.",
    "AutomatedUpdateDate": 0.0
  },
  "DomainEndpointOptions": {
    "EnforceHTTPS": true,
    "TLSSecurityPolicy": "Policy-Min-TLS-1-0-2019-07"
  },
  "AdvancedSecurityOptions": {
    "Enabled": true,
    "InternalUserDatabaseEnabled": true
  }
}
```

有关更多信息，请参阅[亚马逊 Elasticsearch Service 开发者指南中的创建和管理亚马逊 Elasticsearch 服务域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeElasticsearchDomain](#)中的。

## describe-elasticsearch-domains

以下代码示例演示了如何使用 describe-elasticsearch-domains。

### AWS CLI

获取一个或多个域名的详细信息

以下describe-elasticsearch-domains示例提供了一个或多个域的配置详细信息。

```
aws es describe-elasticsearch-domains \
  --domain-names cli-example-1 cli-example-2
```

输出：

```
{
  "DomainStatusList": [{
    "DomainId": "123456789012/cli-example-1",
    "DomainName": "cli-example-1",
    "ARN": "arn:aws:es:us-east-1:123456789012:domain/cli-example-1",
    "Created": true,
    "Deleted": false,
    "Endpoint": "search-cli-example-1-1a2a3a4a5a6a7a8a9a0a.us-
east-1.es.amazonaws.com",
    "Processing": false,
    "UpgradeProcessing": false,
    "ElasticsearchVersion": "7.4",
    "ElasticsearchClusterConfig": {
      "InstanceType": "c5.large.elasticsearch",
      "InstanceCount": 1,
      "DedicatedMasterEnabled": true,
      "ZoneAwarenessEnabled": false,
      "DedicatedMasterType": "c5.large.elasticsearch",
      "DedicatedMasterCount": 3,
      "WarmEnabled": true,
      "WarmType": "ultrawarm1.medium.elasticsearch",
      "WarmCount": 2
    },
    "EBSOptions": {
      "EBSEnabled": true,
      "VolumeType": "gp2",
      "VolumeSize": 10
    },
    "AccessPolicies": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\": \"Allow\", \"Principal\": {\"AWS\": \"*\"}, \"Action\": \"es:*\", \"Resource\": \"arn:aws:es:us-east-1:123456789012:domain/cli-example-1/*\"}]}",
    "SnapshotOptions": {
      "AutomatedSnapshotStartHour": 0
    },
    "CognitoOptions": {
      "Enabled": false
    },
    "EncryptionAtRestOptions": {
      "Enabled": true,
      "KmsKeyId": "arn:aws:kms:us-
east-1:123456789012:key/1a2a3a4a-1a2a-1a2a-1a2a-1a2a3a4a5a6a"
    },
  ]
}
```

```
    "NodeToNodeEncryptionOptions": {
      "Enabled": true
    },
    "AdvancedOptions": {
      "rest.action.multi.allow_explicit_index": "true"
    },
    "ServiceSoftwareOptions": {
      "CurrentVersion": "R20200522",
      "NewVersion": "",
      "UpdateAvailable": false,
      "Cancellable": false,
      "UpdateStatus": "COMPLETED",
      "Description": "There is no software update available for this
domain.",
      "AutomatedUpdateDate": 0.0
    },
    "DomainEndpointOptions": {
      "EnforceHTTPS": true,
      "TLSSecurityPolicy": "Policy-Min-TLS-1-0-2019-07"
    },
    "AdvancedSecurityOptions": {
      "Enabled": true,
      "InternalUserDatabaseEnabled": true
    }
  },
  {
    "DomainId": "123456789012/cli-example-2",
    "DomainName": "cli-example-2",
    "ARN": "arn:aws:es:us-east-1:123456789012:domain/cli-example-2",
    "Created": true,
    "Deleted": false,
    "Processing": true,
    "UpgradeProcessing": false,
    "ElasticsearchVersion": "7.4",
    "ElasticsearchClusterConfig": {
      "InstanceType": "r5.large.elasticsearch",
      "InstanceCount": 1,
      "DedicatedMasterEnabled": false,
      "ZoneAwarenessEnabled": false,
      "WarmEnabled": false
    },
    "EBSOptions": {
      "EBSEnabled": true,
      "VolumeType": "gp2",
```



```

        "VolumeSize": 10
      },
      "AccessPolicies": "{ \"Version\": \"2012-10-17\", \"Statement\": [{ \"Effect\": \"Deny\", \"Principal\": { \"AWS\": \"*\" }, \"Action\": \"es:*\", \"Resource\": \"arn:aws:es:us-east-1:123456789012:domain/cli-example-2/*\" } ] }",
      "SnapshotOptions": {
        "AutomatedSnapshotStartHour": 0
      },
      "CognitoOptions": {
        "Enabled": false
      },
      "EncryptionAtRestOptions": {
        "Enabled": false
      },
      "NodeToNodeEncryptionOptions": {
        "Enabled": false
      },
      "AdvancedOptions": {
        "rest.action.multi.allow_explicit_index": "true"
      },
      "ServiceSoftwareOptions": {
        "CurrentVersion": "",
        "NewVersion": "",
        "UpdateAvailable": false,
        "Cancellable": false,
        "UpdateStatus": "COMPLETED",
        "Description": "There is no software update available for this
domain.",
        "AutomatedUpdateDate": 0.0
      },
      "DomainEndpointOptions": {
        "EnforceHTTPS": false,
        "TLSSecurityPolicy": "Policy-Min-TLS-1-0-2019-07"
      },
      "AdvancedSecurityOptions": {
        "Enabled": false,
        "InternalUserDatabaseEnabled": false
      }
    }
  ]
}

```

有关更多信息，请参阅[亚马逊 Elasticsearch Service 开发者指南中的创建和管理亚马逊 Elasticsearch 服务域](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeElasticsearchDomains](#)中的。

## describe-reserved-elasticsearch-instances

以下代码示例演示了如何使用 describe-reserved-elasticsearch-instances。

### AWS CLI

查看所有预留实例

以下 describe-elasticsearch-domains 示例汇总了您在某个地区预留的所有实例。

```
aws es describe-reserved-elasticsearch-instances
```

输出：

```
{
  "ReservedElasticsearchInstances": [{
    "FixedPrice": 100.0,
    "ReservedElasticsearchInstanceOfferingId":
"1a2a3a4a5-1a2a-3a4a-5a6a-1a2a3a4a5a6a",
    "ReservationName": "my-reservation",
    "PaymentOption": "PARTIAL_UPFRONT",
    "UsagePrice": 0.0,
    "ReservedElasticsearchInstanceId": "9a8a7a6a-5a4a-3a2a-1a0a-9a8a7a6a5a4a",
    "RecurringCharges": [{
      "RecurringChargeAmount": 0.603,
      "RecurringChargeFrequency": "Hourly"
    }],
    "State": "payment-pending",
    "StartTime": 1522872571.229,
    "ElasticsearchInstanceCount": 3,
    "Duration": 31536000,
    "ElasticsearchInstanceType": "m4.2xlarge.elasticsearch",
    "CurrencyCode": "USD"
  ]
}
```

有关更多信息，请参阅 Amazon Elasticsearch Service 开发者指南中的[预留实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeReservedElasticsearchInstances](#)中的。

## list-domain-names

以下代码示例演示了如何使用 list-domain-names。

### AWS CLI

列出所有域名

以下list-domain-names示例提供了该地区所有域名的快速摘要。

```
aws es list-domain-names
```

输出：

```
{
  "DomainNames": [{
    "DomainName": "cli-example-1"
  },
  {
    "DomainName": "cli-example-2"
  }
]
```

有关更多信息，请参阅[亚马逊 Elasticsearch Service 开发者指南中的创建和管理亚马逊 Elasticsearch 服务域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDomainNames](#)中的。

## AWS OpsWorks 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS OpsWorks。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### **assign-instance**

以下代码示例演示了如何使用 `assign-instance`。

#### AWS CLI

为图层分配注册实例

以下示例将注册的实例分配给自定义层。

```
aws opsworks --region us-east-1 assign-instance --instance-id 4d6d1710-ded9-42a1-b08e-b043ad7af1e2 --layer-ids 26cf1d32-6876-42fa-bbf1-9cadc0bff938
```

输出：无。

#### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的为层分配注册实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssignInstance](#)中的。

### **assign-volume**

以下代码示例演示了如何使用 `assign-volume`。

#### AWS CLI

为实例分配注册卷

以下示例将注册的亚马逊弹性区块存储 (Amazon EBS) 卷分配给实例。该卷由其卷 ID 标识，这是您在向堆栈注册卷时 AWS OpsWorks 分配的 GUID，而不是亚马逊弹性计算云 (Amazon EC2) 卷 ID。在运行之前 `assign-volume`，必须先运行 `update-volume` 为卷分配装入点。

```
aws opsworks --region us-east-1 assign-volume --instance-id 4d6d1710-ded9-42a1-b08e-b043ad7af1e2 --volume-id 26cf1d32-6876-42fa-bbf1-9cadc0bff938
```

输出：无。

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的将 Amazon EBS 卷分配给实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssignVolume](#)中的。

## associate-elastic-ip

以下代码示例演示了如何使用 `associate-elastic-ip`。

### AWS CLI

将弹性 IP 地址与实例关联

以下示例将弹性 IP 地址与指定实例相关联。

```
aws opsworks --region us-east-1 associate-elastic-ip --instance-id dfe18b02-5327-493d-91a4-c5c0c448927f --elastic-ip 54.148.130.96
```

输出：无。

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的资源管理。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateElasticIp](#)中的。

## attach-elastic-load-balancer

以下代码示例演示了如何使用 `attach-elastic-load-balancer`。

### AWS CLI

将负载均衡器连接到层

以下示例将一个由其名称标识的负载均衡器连接到指定层。

```
aws opsworks --region us-east-1 attach-elastic-load-balancer --elastic-load-balancer-name Java-LB --layer-id 888c5645-09a5-4d0e-95a8-812ef1db76a4
```

输出：无。

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的 [Elastic Load Balancing](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AttachElasticLoadBalancer](#) 中的。

## create-app

以下代码示例演示了如何使用 create-app。

### AWS CLI

#### 示例 1：创建应用程序

以下示例使用存储在存储库中的代码创建了一个名为 SimplephpApp 的 PHP 应用程序。GitHub 该命令使用应用程序源定义的简写形式。

```
aws opsworks create-app \  
  --region us-east-1 \  
  --stack-id f6673d70-32e6-4425-8999-265dd002fec7 \  
  --name SimplePHPApp \  
  --type php \  
  --app-source Type=git,Url=https://github.com/amazonwebservices/opsworks-demo-php-simple-app.git,Revision=version1
```

输出：

```
{  
  "AppId": "6cf5163c-a951-444f-a8f7-3716be75f2a2"  
}
```

#### 示例 2：创建带有附加数据库的应用程序

以下示例使用存储在公共 S3 存储桶中的 .zip 存档中的代码创建 JSP 应用程序。它附加 RDS 数据库实例以用作应用程序的数据存储。应用程序和数据库源在单独的 JSON 文件中定义，这些文件位于您运行命令的目录中。

```
aws opsworks create-app \  
  --region us-east-1 \  
  --stack-id 8c428b08-a1a1-46ce-a5f8-feddc43771b8 \  
  --name SimpleJSP \  
  --type java \  
  --app-source file://appsource.json \  
  --data-sources file://datasource.json
```

应用程序源信息位于其中appsource.json并包含以下内容。

```
{  
  "Type": "archive",  
  "Url": "https://s3.amazonaws.com/opsworks-demo-assets/simplejsp.zip"  
}
```

数据库源信息位于其中datasource.json并包含以下内容。

```
[  
  {  
    "Type": "RdsDbInstance",  
    "Arn": "arn:aws:rds:us-west-2:123456789012:db:clitestdb",  
    "DatabaseName": "mydb"  
  }  
]
```

注意：对于 RDS 数据库实例，必须先使用register-rds-db-instance向堆栈注册该实例。对于 MySQL 应用服务器实例，Type请设置为OpsworksMySQLInstance。这些实例由创建 AWS OpsWorks，因此不必注册。

输出：

```
{  
  "AppId": "26a61ead-d201-47e3-b55c-2a7c666942f8"  
}
```

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的“添加应用程序”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateApp](#)中的。

## create-deployment

以下代码示例演示了如何使用 create-deployment。

### AWS CLI

#### 示例 1：部署应用程序和运行堆栈命令

以下示例说明如何使用该 create-deployment 命令部署应用程序和运行堆栈命令。请注意，指定命令的 JSON 对象中的 quote (") 字符都以转义字符 (\) 开头。如果没有转义字符，该命令可能会返回无效的 JSON 错误。

以下 create-deployment 示例将应用程序部署到指定的堆栈。

```
aws opsworks create-deployment \  
  --stack-id cfb7e082-ad1d-4599-8e81-de1c39ab45bf \  
  --app-id 307be5c8-d55d-47b5-bd6e-7bd417c6c7eb \  
  --command "{\"Name\":\"deploy\"}"
```

输出：

```
{  
  "DeploymentId": "5746c781-df7f-4c87-84a7-65a119880560"  
}
```

#### 示例 2：部署 Rails 应用程序并迁移数据库

以下 create-deployment 命令将 Ruby on Rails 应用程序部署到指定的堆栈并迁移数据库。

```
aws opsworks create-deployment \  
  --stack-id cfb7e082-ad1d-4599-8e81-de1c39ab45bf \  
  --app-id 307be5c8-d55d-47b5-bd6e-7bd417c6c7eb \  
  --command "{\"Name\":\"deploy\", \"Args\":{\"migrate\":[\"true\"]}]}"
```

输出：

```
{  
  "DeploymentId": "5746c781-df7f-4c87-84a7-65a119880560"  
}
```

有关部署的更多信息，请参阅 AWS OpsWorks 用户指南中的 [部署应用程序](#)。



### 示例 3：运行食谱

以下 `create-deployment` 命令在指定堆栈中的实例上运行自定义配方。 `phpapp::appsetup`

```
aws opsworks create-deployment \  
  --stack-id 935450cc-61e0-4b03-a3e0-160ac817d2bb \  
  --command "{\"Name\":\"execute_recipes\", \"Args\":{\"recipes\":  
[\"phpapp::appsetup\"]}}"
```

输出：

```
{  
  "DeploymentId": "5cbaa7b9-4e09-4e53-aa1b-314fbd106038"  
}
```

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的 [“运行堆栈命令”](#)。

### 示例 4：安装依赖关系

以下 `create-deployment` 命令在指定堆栈中的实例上安装依赖项，例如软件包或 Ruby gem。

```
aws opsworks create-deployment \  
  --stack-id 935450cc-61e0-4b03-a3e0-160ac817d2bb \  
  --command "{\"Name\":\"install_dependencies\"}"
```

输出：

```
{  
  "DeploymentId": "aef5b255-8604-4928-81b3-9b0187f962ff"  
}
```

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的 [“运行堆栈命令”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDeployment](#) 中的。

## create-instance

以下代码示例演示了如何使用 `create-instance`。

### AWS CLI

#### 创建实例

以下 `create-instance` 命令在指定堆栈中创建名为 `myinstance1` 的 `m1.large` Amazon Linux 实例。该实例被分配给一个图层。

```
aws opsworks --region us-east-1 create-instance --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb --layer-ids 5c8c272a-f2d5-42e3-8245-5bf3927cb65b --hostname
myinstance1 --instance-type m1.large --os "Amazon Linux"
```

要使用自动生成的名称 `get-hostname-suggestion`，请调用，它会根据您在创建堆栈时指定的主题生成主机名。然后将该名称传递给主机名参数。

输出：

```
{
  "InstanceId": "5f9adeaa-c94c-42c6-aeef-28a5376002cd"
}
```

更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的向层添加实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateInstance](#)中的。

## create-layer

以下代码示例演示了如何使用 `create-layer`。

AWS CLI

创建图层

以下 `create-layer` 命令在指定堆栈中创建名为 `MyphPlayer` 的 PHP 应用服务器层。

```
aws opsworks create-layer --region us-east-1 --stack-id
f6673d70-32e6-4425-8999-265dd002fec7 --type php-app --name MyPHPLayer --shortname
myphpplayer
```

输出：

```
{
  "LayerId": "0b212672-6b4b-40e4-8a34-5a943cf2e07a"
}
```

## 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的如何创建图层。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLayer](#)中的。

## create-server

以下代码示例演示了如何使用 create-server。

### AWS CLI

#### 创建服务器

以下create-server示例创建了一个automate-06在您的默认区域中命名的新 Chef Automate 服务器。请注意，大多数其他设置都使用默认值，例如要保留的备份数量以及维护和备份的开始时间。在运行create-server命令之前，请完成 AWS Opsworks for Chef Automate 用户指南中 Chef Automate [入门](#)中的先决条件。 AWS OpsWorks

```
aws opsworks-cm create-server \
  --engine "ChefAutomate" \
  --instance-profile-arn "arn:aws:iam::012345678901:instance-profile/aws-opsworks-
cm-ec2-role" \
  --instance-type "t2.medium" \
  --server-name "automate-06" \
  --service-role-arn "arn:aws:iam::012345678901:role/aws-opsworks-cm-service-role"
```

输出：

```
{
  "Server": {
    "AssociatePublicIpAddress": true,
    "BackupRetentionCount": 10,
    "CreatedAt": 2019-12-29T13:38:47.520Z,
    "DisableAutomatedBackup": FALSE,
    "Endpoint": "https://opsworks-cm.us-east-1.amazonaws.com",
    "Engine": "ChefAutomate",
    "EngineAttributes": [
      {
        "Name": "CHEF_AUTOMATE_ADMIN_PASSWORD",
        "Value": "1Example1"
      }
    ],
  },
}
```

```
    "EngineModel": "Single",
    "EngineVersion": "2019-08",
    "InstanceProfileArn": "arn:aws:iam::012345678901:instance-profile/aws-opsworks-cm-ec2-role",
    "InstanceType": "t2.medium",
    "PreferredBackupWindow": "Sun:02:00",
    "PreferredMaintenanceWindow": "00:00",
    "SecurityGroupIds": [ "sg-12345678" ],
    "ServerArn": "arn:aws:iam::012345678901:instance/automate-06-1010V4UU2WRM2",
    "ServerName": "automate-06",
    "ServiceRoleArn": "arn:aws:iam::012345678901:role/aws-opsworks-cm-service-role",
    "Status": "CREATING",
    "SubnetIds": [ "subnet-12345678" ]
  }
}
```

有关更多信息，请参阅 for Chef Automate API 参考[CreateServer](#)中的。AWS OpsWorks

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateServer](#)中的。

## create-stack

以下代码示例演示了如何使用 create-stack。

### AWS CLI

#### 创建堆栈

以下create-stack命令创建一个名为 CLI Stack 的堆栈。

```
aws opsworks create-stack --name "CLI Stack" --stack-region "us-east-1" --service-role-arn arn:aws:iam::123456789012:role/aws-opsworks-service-role --default-instance-profile-arn arn:aws:iam::123456789012:instance-profile/aws-opsworks-ec2-role --region us-east-1
```

service-role-arn 和 default-instance-profile-arn 参数是必需的。在 AWS OpsWorks创建第一个堆栈时，您通常会使用为您创建的堆栈。要获取您账户的 Amazon 资源名称 (ARN)，请前往 IAM 控制台，Roles在导航面板中选择，选择角色或个人资料，然后选择Summary选项卡。

输出：

```
{
  "StackId": "f6673d70-32e6-4425-8999-265dd002fec7"
}
```

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的“创建新堆栈”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateStack](#)中的。

## create-user-profile

以下代码示例演示了如何使用 create-user-profile。

### AWS CLI

#### 创建用户个人资料

您可以 AWS OpsWorks 通过调用 create-user-profile 创建用户配置文件将 AWS 身份和访问管理器 (IAM) 用户导入。以下示例为由 Amazon 资源名称 (ARN) 标识的 cli-user-test IAM 用户创建了用户档案。该示例为用户分配一个 SSH 用户名myusername并启用自我管理，这允许用户指定 SSH 公钥。

```
aws opsworks --region us-east-1 create-user-profile --iam-user-arn
arn:aws:iam::123456789102:user/cli-user-test --ssh-username myusername --allow-
self-management
```

输出：

```
{
  "IamUserArn": "arn:aws:iam::123456789102:user/cli-user-test"
}
```

提示：此命令将 IAM 用户导入 AWS OpsWorks，但只能导入附加策略所授予的权限。您可以使用set-permissions命令授予每个堆栈的 AWS OpsWorks 权限。

### 更多信息

有关更多信息，请参阅《用户指南》AWS OpsWorks 中的将AWS OpsWorks 用户导入。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUserProfile](#)中的。

## delete-app

以下代码示例演示了如何使用 delete-app。

### AWS CLI

#### 删除应用程序

以下示例删除了由其应用程序 ID 标识的指定应用程序。您可以通过访问 AWS OpsWorks 控制台上的应用程序详细信息页面或运行 describe-apps 命令来获取应用程序 ID。

```
aws opsworks delete-app --region us-east-1 --app-id 577943b9-2ec1-4baf-  
a7bf-1d347601edc5
```

输出：无。

#### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的应用程序。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteApp](#) 中的。

## delete-instance

以下代码示例演示了如何使用 delete-instance。

### AWS CLI

#### 删除实例

以下 delete-instance 示例删除了由其实例 ID 标识的指定实例。您可以通过在 AWS OpsWorks 控制台中打开实例的详细信息页面或运行 describe-instances 命令来查找实例 ID。

如果实例处于联机状态，则必须先通过调用停止实例 stop-instance，然后必须等到实例停止。运行 describe-instances 以检查实例状态。

要移除实例的 Amazon EBS 卷或弹性 IP 地址，请分别添加 --delete-volumes 或 --delete-elastic-ip 参数。

```
aws opsworks delete-instance \  
  --region us-east-1 \  
  --instance-id i-12345678
```

```
--instance-id 3a21cfac-4a1f-4ce2-a921-b2cfba6f7771
```

此命令不生成任何输出。

有关更多信息，请参阅AWS OpsWorks 用户指南中的[删除 AWS OpsWorks 实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInstance](#)中的。

## delete-layer

以下代码示例演示了如何使用 delete-layer。

### AWS CLI

#### 删除图层

以下示例删除了由其层 ID 标识的指定层。您可以通过访问 AWS OpsWorks 控制台上的图层详细信息页面或运行describe-layers命令来获取图层 ID。

注意：在删除图层之前，delete-instance必须使用删除该图层的所有实例。

```
aws opsworks delete-layer --region us-east-1 --layer-id a919454e-b816-4598-  
b29a-5796afb498ed
```

输出：无。

#### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的[删除 AWS OpsWorks 实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLayer](#)中的。

## delete-stack

以下代码示例演示了如何使用 delete-stack。

### AWS CLI

#### 删除堆栈

以下示例删除指定的堆栈，该堆栈由其堆栈 ID 标识。您可以通过单击 AWS OpsWorks 控制台上的堆栈设置或运行describe-stacks命令来获取堆栈 ID。

注意：在删除图层之前，必须使用 `delete-appdelete-instance`、和 `delete-layer` 来删除堆栈的所有应用程序、实例和图层。

```
aws opsworks delete-stack --region us-east-1 --stack-id
154a9d89-7e9e-433b-8de8-617e53756c84
```

输出：无。

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的关闭堆栈。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteStack](#) 中的。

## delete-user-profile

以下代码示例演示了如何使用 `delete-user-profile`。

### AWS CLI

删除用户个人资料并从中移除 IAM 用户 AWS OpsWorks

以下示例删除了由亚马逊资源名称 (ARN) 标识的指定 AWS 身份和访问管理 (IAM) 用户的用户资料。该操作将用户从 IAM 用户中移除 AWS OpsWorks，但不会将其删除。您必须使用 IAM 控制台、CLI 或 API 来完成该任务。

```
aws opsworks --region us-east-1 delete-user-profile --iam-user-arn
arn:aws:iam::123456789102:user/cli-user-test
```

输出：无。

### 更多信息

有关更多信息，请参阅《用户指南》AWS OpsWorks 中的将 AWS OpsWorks 用户导入。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteUserProfile](#) 中的。

## deregister-elastic-ip

以下代码示例演示了如何使用 `deregister-elastic-ip`。



## AWS CLI

从堆栈中取消注册弹性 IP 地址

以下示例从堆栈中注销由其 IP 地址标识的弹性 IP 地址。

```
aws opsworks deregister-elastic-ip --region us-east-1 --elastic-ip 54.148.130.96
```

输出：无。

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的取消注册弹性 IP 地址。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterElasticIp](#)中的。

## deregister-instance

以下代码示例演示了如何使用 deregister-instance。

### AWS CLI

从堆栈中取消注册已注册的实例

以下deregister-instance命令从其堆栈中取消注册已注册的实例。

```
aws opsworks --region us-east-1 deregister-instance --instance-id 4d6d1710-ded9-42a1-b08e-b043ad7af1e2
```

输出：无。

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的注销注册实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterInstance](#)中的。

## deregister-rds-db-instance

以下代码示例演示了如何使用 deregister-rds-db-instance。

## AWS CLI

### 从堆栈中注销 Amazon RDS 数据库实例

以下示例从堆栈中注销 RDS 数据库实例（由其 ARN 标识）。

```
aws opsworks deregister-rds-db-instance --region us-east-1 --rds-db-instance-arn
arn:aws:rds:us-west-2:123456789012:db:clitestdb
```

输出：无。

### 更多信息

有关更多信息，请参阅 ASW OpsWorks 用户指南中的注销 Amazon RDS 实例。

实例 ID : clitestdb Master 用户名 : cliuser Master PWD : some23 ! pwd DB 名称 : mydb  
aws opsworks — region us-east-1 — arn: aws: rds: us-west deregister-rds-db-instance  
-2:645732743964: db: clitestdb rds-db-instance-arn

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeregisterRdsDbInstance](#) 中的。

## deregister-volume

以下代码示例演示了如何使用 deregister-volume。

## AWS CLI

### 取消注册 Amazon EBS 卷

以下示例从堆栈中注销 EBS 卷。该卷由其卷 ID 标识，该卷是您在堆栈中注册卷时 AWS OpsWorks 分配的 GUID，而不是 EC2 卷 ID。

```
aws opsworks deregister-volume --region us-east-1 --volume-id 5c48ef52-3144-4bf5-
beaa-fda4deb23d4d
```

输出：无。

### 更多信息

有关更多信息，请参阅 AWS OpsWorks 用户指南中的注销 Amazon EBS 卷。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeregisterVolume](#) 中的。

## describe-apps

以下代码示例演示了如何使用 describe-apps。

### AWS CLI

#### 描述应用程序

以下describe-apps命令描述了指定堆栈中的应用程序。

```
aws opsworks describe-apps \  
  --stack-id 38ee91e2-abdc-4208-a107-0b7168b3cc7a \  
  --region us-east-1
```

输出：

```
{  
  "Apps": [  
    {  
      "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",  
      "AppSource": {  
        "Url": "https://s3-us-west-2.amazonaws.com/opsworks-demo-assets/  
simplejsp.zip",  
        "Type": "archive"  
      },  
      "Name": "SimpleJSP",  
      "EnableSsl": false,  
      "SslConfiguration": {},  
      "AppId": "da1decc1-0dff-43ea-ad7c-bb667cd87c8b",  
      "Attributes": {  
        "RailsEnv": null,  
        "AutoBundleOnDeploy": "true",  
        "DocumentRoot": "ROOT"  
      },  
      "Shortname": "simplejsp",  
      "Type": "other",  
      "CreatedAt": "2013-08-01T21:46:54+00:00"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的应用程序。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeApps](#)中的。

## describe-commands

以下代码示例演示了如何使用 describe-commands。

### AWS CLI

#### 描述命令

以下 describe-commands 命令描述了指定实例中的命令。

```
aws opsworks describe-commands \  
  --instance-id 8c2673b9-3fe5-420d-9cfa-78d875ee7687 \  
  --region us-east-1
```

输出：

```
{  
  "Commands": [  
    {  
      "Status": "successful",  
      "CompletedAt": "2013-07-25T18:57:47+00:00",  
      "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",  
      "DeploymentId": "6ed0df4c-9ef7-4812-8dac-d54a05be1029",  
      "AcknowledgedAt": "2013-07-25T18:57:41+00:00",  
      "LogUrl": "https://s3.amazonaws.com/<bucket-name>/logs/008c1a91-ec59-4d51-971d-3adff54b00cc?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE  
&Expires=1375394373&Signature=HkXil6UuNfxTCC37EPQAA462E1E%3D&response-cache-control=private&response-content-encoding=gzip&response-content-type=text%2Fplain",  
      "Type": "undeploy",  
      "CommandId": "008c1a91-ec59-4d51-971d-3adff54b00cc",  
      "CreatedAt": "2013-07-25T18:57:34+00:00",  
      "ExitCode": 0  
    },  
    {  
      "Status": "successful",  
      "CompletedAt": "2013-07-25T18:55:40+00:00",  
      "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",  
      "DeploymentId": "19d3121e-d949-4ff2-9f9d-94eac087862a",  
      "AcknowledgedAt": "2013-07-25T18:55:32+00:00",  
      "LogUrl": "https://s3.amazonaws.com/<bucket-name>/logs/899d3d64-0384-47b6-a586-33433aad117c?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE  
&Expires=1375394373&Signature=xMsJvtLuUqWmsr8s%2FAjVru0BtRs%3D&response-cache-control=private&response-content-encoding=gzip&response-content-type=text%2Fplain",  
      "Type": "deploy",
```

```
        "CommandId": "899d3d64-0384-47b6-a586-33433aad117c",
        "CreatedAt": "2013-07-25T18:55:29+00:00",
        "ExitCode": 0
    }
]
}
```

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的 AWS OpsWorks 生命周期事件。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeCommands](#) 中的。

## describe-deployments

以下代码示例演示了如何使用 describe-deployments。

### AWS CLI

#### 描述部署

以下 describe-deployments 命令描述了指定堆栈中的部署。

```
aws opsworks --region us-east-1 describe-deployments --stack-id 38ee91e2-abdc-4208-
a107-0b7168b3cc7a
```

输出：

```
{
  "Deployments": [
    {
      "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
      "Status": "successful",
      "CompletedAt": "2013-07-25T18:57:49+00:00",
      "DeploymentId": "6ed0df4c-9ef7-4812-8dac-d54a05be1029",
      "Command": {
        "Args": {},
        "Name": "undeploy"
      },
      "CreatedAt": "2013-07-25T18:57:34+00:00",
      "Duration": 15,
      "InstanceIds": [
        "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
        "9e588a25-35b2-4804-bd43-488f85ebe5b7"
      ]
    }
  ]
}
```

```
    },
    {
      "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
      "Status": "successful",
      "CompletedAt": "2013-07-25T18:56:41+00:00",
      "IamUserArn": "arn:aws:iam::123456789012:user/someuser",
      "DeploymentId": "19d3121e-d949-4ff2-9f9d-94eac087862a",
      "Command": {
        "Args": {},
        "Name": "deploy"
      },
      "InstanceIds": [
        "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
        "9e588a25-35b2-4804-bd43-488f85ebe5b7"
      ],
      "Duration": 72,
      "CreatedAt": "2013-07-25T18:55:29+00:00"
    }
  ]
}
```

## 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的“部署应用程序”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDeployments](#) 中的。

## describe-elastic-ips

以下代码示例演示了如何使用 describe-elastic-ips。

### AWS CLI

#### 描述弹性 IP 实例

以下 describe-elastic-ips 命令描述了指定实例中的弹性 IP 地址。

```
aws opsworks --region us-east-1 describe-elastic-ips --instance-id b62f3e04-
e9eb-436c-a91f-d9e9a396b7b0
```

输出：

```
{
```

```
"ElasticIps": [  
  {  
    "Ip": "192.0.2.0",  
    "Domain": "standard",  
    "Region": "us-west-2"  
  }  
]  
}
```

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeElasticIps](#)中的。

## describe-elastic-load-balancers

以下代码示例演示了如何使用 describe-elastic-load-balancers。

### AWS CLI

#### 描述堆栈的弹性负载均衡器

以下describe-elastic-load-balancers命令描述了指定堆栈的负载均衡器。

```
aws opsworks --region us-west-2 describe-elastic-load-balancers --stack-id  
6f4660e5-37a6-4e42-bfa0-1358ebd9c182
```

输出：这个特定的堆栈有一个负载均衡器。

```
{  
  "ElasticLoadBalancers": [  
    {  
      "SubnetIds": [  
        "subnet-60e4ea04",  
        "subnet-66e1c110"  
      ],  
      "Ec2InstanceIds": [],  
      "ElasticLoadBalancerName": "my-balancer",  
      "Region": "us-west-2",  
      "LayerId": "344973cb-bf2b-4cd0-8d93-51cd819bab04",  
      "AvailabilityZones": [  
        "us-west-2a",  
        "us-west-2b"  
      ]  
    }  
  ]  
}
```

```
        "us-west-2b"  
    ],  
    "VpcId": "vpc-b319f9d4",  
    "StackId": "6f4660e5-37a6-4e42-bfa0-1358ebd9c182",  
    "DnsName": "my-balancer-2094040179.us-west-2.elb.amazonaws.com"  
  }  
]  
}
```

## 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的应用程序。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeElasticLoadBalancers](#) 中的。

## describe-instances

以下代码示例演示了如何使用 describe-instances。

### AWS CLI

#### 描述实例

以下 describe-instances 命令描述了指定堆栈中的实例：

```
aws opsworks --region us-east-1 describe-instances --stack-id 8c428b08-a1a1-46ce-a5f8-feddc43771b8
```

输出：以下输出示例适用于具有两个实例的堆栈。第一个是注册的 EC2 实例，第二个是由创建的 AWS OpsWorks。

```
{  
  "Instances": [  
    {  
      "StackId": "71c7ca72-55ae-4b6a-8ee1-a8dcdded3fa0f",  
      "PrivateDns": "ip-10-31-39-66.us-west-2.compute.internal",  
      "LayerIds": [  
        "26cf1d32-6876-42fa-bbf1-9cadcd0bff938"  
      ],  
      "EbsOptimized": false,  
      "ReportedOs": {  
        "Version": "14.04",  
        "Name": "ubuntu",  
      }  
    }  
  ]  
}
```



```
    "Family": "debian"
  },
  "Status": "online",
  "InstanceId": "4d6d1710-ded9-42a1-b08e-b043ad7af1e2",
  "SshKeyName": "US-West-2",
  "InfrastructureClass": "ec2",
  "RootDeviceVolumeId": "vol-d08ec6c1",
  "SubnetId": "subnet-b8de0ddd",
  "InstanceType": "t1.micro",
  "CreatedAt": "2015-02-24T20:52:49+00:00",
  "AmiId": "ami-35501205",
  "Hostname": "ip-192-0-2-0",
  "Ec2InstanceId": "i-5cd23551",
  "PublicDns": "ec2-192-0-2-0.us-west-2.compute.amazonaws.com",
  "SecurityGroupIds": [
    "sg-c4d3f0a1"
  ],
  "Architecture": "x86_64",
  "RootDeviceType": "ebs",
  "InstallUpdatesOnBoot": true,
  "Os": "Custom",
  "VirtualizationType": "paravirtual",
  "AvailabilityZone": "us-west-2a",
  "PrivateIp": "10.31.39.66",
  "PublicIp": "192.0.2.06",
  "RegisteredBy": "arn:aws:iam::123456789102:user/AWS/OpsWorks/OpsWorks-
EC2Register-i-5cd23551"
},
{
  "StackId": "71c7ca72-55ae-4b6a-8ee1-a8dcdded3fa0f",
  "PrivateDns": "ip-10-31-39-158.us-west-2.compute.internal",
  "SshHostRsaKeyFingerprint": "69:6b:7b:8b:72:f3:ed:23:01:00:05:bc:9f:a4:60:c1",
  "LayerIds": [
    "26cf1d32-6876-42fa-bbf1-9cad0bfff938"
  ],
  "EbsOptimized": false,
  "ReportedOs": {},
  "Status": "booting",
  "InstanceId": "9b137a0d-2f5d-4cc0-9704-13da4b31fdcb",
  "SshKeyName": "US-West-2",
  "InfrastructureClass": "ec2",
  "RootDeviceVolumeId": "vol-e09dd5f1",
  "SubnetId": "subnet-b8de0ddd",
```

```
    "InstanceProfileArn": "arn:aws:iam::123456789102:instance-profile/aws-opsworks-ec2-role",
    "InstanceType": "c3.large",
    "CreatedAt": "2015-02-24T21:29:33+00:00",
    "AmiId": "ami-9fc29baf",
    "SshHostDsaKeyFingerprint": "fc:87:95:c3:f5:e1:3b:9f:d2:06:6e:62:9a:35:27:e8",
    "Ec2InstanceId": "i-8d2dca80",
    "PublicDns": "ec2-192-0-2-1.us-west-2.compute.amazonaws.com",
    "SecurityGroupIds": [
      "sg-b022add5",
      "sg-b122add4"
    ],
    "Architecture": "x86_64",
    "RootDeviceType": "ebs",
    "InstallUpdatesOnBoot": true,
    "Os": "Amazon Linux 2014.09",
    "VirtualizationType": "paravirtual",
    "AvailabilityZone": "us-west-2a",
    "Hostname": "custom11",
    "PrivateIp": "10.31.39.158",
    "PublicIp": "192.0.2.0"
  }
]
}
```

## 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInstances](#)中的。

## describe-layers

以下代码示例演示了如何使用 describe-layers。

### AWS CLI

#### 描述堆栈的层

以下describe-layers命令描述了指定堆栈中的层：

```
aws opsworks --region us-east-1 describe-layers --stack-id 38ee91e2-abdc-4208-a107-0b7168b3cc7a
```

输出：

```
{
  "Layers": [
    {
      "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
      "Type": "db-master",
      "DefaultSecurityGroupNames": [
        "AWS-OpsWorks-DB-Master-Server"
      ],
      "Name": "MySQL",
      "Packages": [],
      "DefaultRecipes": {
        "Undeploy": [],
        "Setup": [
          "opsworks_initial_setup",
          "ssh_host_keys",
          "ssh_users",
          "mysql::client",
          "dependencies",
          "ebs",
          "opsworks_ganglia::client",
          "mysql::server",
          "dependencies",
          "deploy::mysql"
        ],
        "Configure": [
          "opsworks_ganglia::configure-client",
          "ssh_users",
          "agent_version",
          "deploy::mysql"
        ],
        "Shutdown": [
          "opsworks_shutdown::default",
          "mysql::stop"
        ],
        "Deploy": [
          "deploy::default",
          "deploy::mysql"
        ]
      },
      "CustomRecipes": {
        "Undeploy": [],
        "Setup": [],

```

```
    "Configure": [],
    "Shutdown": [],
    "Deploy": []
  },
  "EnableAutoHealing": false,
  "LayerId": "41a20847-d594-4325-8447-171821916b73",
  "Attributes": {
    "MysqlRootPasswordUbiquitous": "true",
    "RubygemsVersion": null,
    "RailsStack": null,
    "HaproxyHealthCheckMethod": null,
    "RubyVersion": null,
    "BundlerVersion": null,
    "HaproxyStatsPassword": null,
    "PassengerVersion": null,
    "MemcachedMemory": null,
    "EnableHaproxyStats": null,
    "ManageBundler": null,
    "NodejsVersion": null,
    "HaproxyHealthCheckUrl": null,
    "MysqlRootPassword": "*****FILTERED*****",
    "GangliaPassword": null,
    "GangliaUser": null,
    "HaproxyStatsUrl": null,
    "GangliaUrl": null,
    "HaproxyStatsUser": null
  },
  "Shortname": "db-master",
  "AutoAssignElasticIps": false,
  "CustomSecurityGroupIds": [],
  "CreatedAt": "2013-07-25T18:11:19+00:00",
  "VolumeConfigurations": [
    {
      "MountPoint": "/vol/mysql",
      "Size": 10,
      "NumberOfDisks": 1
    }
  ]
},
{
  "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
  "Type": "custom",
  "DefaultSecurityGroupNames": [
    "AWS-OpsWorks-Custom-Server"
```

```
],
  "Name": "TomCustom",
  "Packages": [],
  "DefaultRecipes": {
    "Undeploy": [],
    "Setup": [
      "opsworks_initial_setup",
      "ssh_host_keys",
      "ssh_users",
      "mysql::client",
      "dependencies",
      "ebs",
      "opsworks_ganglia::client"
    ],
    "Configure": [
      "opsworks_ganglia::configure-client",
      "ssh_users",
      "agent_version"
    ],
    "Shutdown": [
      "opsworks_shutdown::default"
    ],
    "Deploy": [
      "deploy::default"
    ]
  },
  "CustomRecipes": {
    "Undeploy": [],
    "Setup": [
      "tomcat::setup"
    ],
    "Configure": [
      "tomcat::configure"
    ],
    "Shutdown": [],
    "Deploy": [
      "tomcat::deploy"
    ]
  },
  "EnableAutoHealing": true,
  "LayerId": "e6cbcd29-d223-40fc-8243-2eb213377440",
  "Attributes": {
    "MysqlRootPasswordUbiquitous": null,
    "RubygemsVersion": null,
```

```

        "RailsStack": null,
        "HaproxyHealthCheckMethod": null,
        "RubyVersion": null,
        "BundlerVersion": null,
        "HaproxyStatsPassword": null,
        "PassengerVersion": null,
        "MemcachedMemory": null,
        "EnableHaproxyStats": null,
        "ManageBundler": null,
        "NodejsVersion": null,
        "HaproxyHealthCheckUrl": null,
        "MySQLRootPassword": null,
        "GangliaPassword": null,
        "GangliaUser": null,
        "HaproxyStatsUrl": null,
        "GangliaUrl": null,
        "HaproxyStatsUser": null
    },
    "Shortname": "tomcustom",
    "AutoAssignElasticIps": false,
    "CustomSecurityGroupIds": [],
    "CreatedAt": "2013-07-25T18:12:53+00:00",
    "VolumeConfigurations": []
}
]
}

```

## 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的图层。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLayers](#)中的。

## describe-load-based-auto-scaling

以下代码示例演示了如何使用 describe-load-based-auto-scaling。

### AWS CLI

#### 描述图层的基于负载的缩放配置

以下示例描述了指定层的基于负载的缩放配置。图层由其图层 ID 标识，您可以在图层的详细信息页面上找到该图层 ID，也可以通过运行来找到describe-layers。

```
aws opsworks describe-load-based-auto-scaling --region us-east-1 --layer-ids
6bec29c9-c866-41a0-aba5-fa3e374ce2a1
```

输出：示例层有一个基于负载的实例。

```
{
  "LoadBasedAutoScalingConfigurations": [
    {
      "DownScaling": {
        "IgnoreMetricsTime": 10,
        "ThresholdsWaitTime": 10,
        "InstanceCount": 1,
        "CpuThreshold": 30.0
      },
      "Enable": true,
      "UpScaling": {
        "IgnoreMetricsTime": 5,
        "ThresholdsWaitTime": 5,
        "InstanceCount": 1,
        "CpuThreshold": 80.0
      },
      "LayerId": "6bec29c9-c866-41a0-aba5-fa3e374ce2a1"
    }
  ]
}
```

## 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的基于负载的自动扩展的工作原理。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoadBasedAutoScaling](#)中的。

## describe-my-user-profile

以下代码示例演示了如何使用 describe-my-user-profile。

### AWS CLI

#### 获取用户的个人资料

以下示例说明如何获取正在运行该命令的 AWS 身份和访问管理 (IAM) Access Management 用户的个人资料。

```
aws opsworks --region us-east-1 describe-my-user-profile
```

输出：为简洁起见，用户的大部分 SSH 公钥都被省略号 (...) 所取代。

```
{
  "UserProfile": {
    "IamUserArn": "arn:aws:iam::123456789012:user/myusername",
    "SshPublicKey": "ssh-rsa AAAAB3NzaC1yc2EAAAABJQ...3LQ4aX9jpxQw== rsa-
key-20141104",
    "Name": "myusername",
    "SshUsername": "myusername"
  }
}
```

### 更多信息

有关更多信息，请参阅《用户指南》AWS OpsWorks 中的将AWS OpsWorks 用户导入。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeMyUserProfile](#)中的。

## describe-permissions

以下代码示例演示了如何使用 describe-permissions。

### AWS CLI

获取用户的每堆栈 AWS OpsWorks 权限级别

以下示例说明如何获取 AWS 身份和访问管理 (IAM) 用户对指定堆栈的权限级别。

```
aws opsworks --region us-east-1 describe-permissions --iam-user-arn
arn:aws:iam::123456789012:user/cli-user-test --stack-id d72553d4-8727-448c-9b00-
f024f0ba1b06
```

输出：

```
{
  "Permissions": [
    {
      "StackId": "d72553d4-8727-448c-9b00-f024f0ba1b06",
      "IamUserArn": "arn:aws:iam::123456789012:user/cli-user-test",
```



```
    "Level": "manage",
    "AllowSudo": true,
    "AllowSsh": true
  }
]
```

## 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的授予每堆栈权限级别。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePermissions](#)中的。

## describe-raid-arrays

以下代码示例演示了如何使用 describe-raid-arrays。

### AWS CLI

#### 描述 RAID 阵列

以下示例描述了连接到指定堆栈中实例的 RAID 阵列。

```
aws opsworks --region us-east-1 describe-raid-arrays --stack-id
d72553d4-8727-448c-9b00-f024f0ba1b06
```

输出：以下是带有一个 RAID 阵列的堆栈的输出。

```
{
  "RaidArrays": [
    {
      "StackId": "d72553d4-8727-448c-9b00-f024f0ba1b06",
      "AvailabilityZone": "us-west-2a",
      "Name": "Created for php-app1",
      "NumberOfDisks": 2,
      "InstanceId": "9f14adbc-ced5-43b6-bf01-e7d0db6cf2f7",
      "RaidLevel": 0,
      "VolumeType": "standard",
      "RaidArrayId": "f2d4e470-5972-4676-b1b8-bae41ec3e51c",
      "Device": "/dev/md0",
      "MountPoint": "/mnt/workspace",
      "CreatedAt": "2015-02-26T23:53:09+00:00",
      "Size": 100
    }
  ]
}
```

```
    }  
  ]  
}
```

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的 EBS 卷。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeRaidArrays](#)中的。

## describe-rds-db-instances

以下代码示例演示了如何使用 `describe-rds-db-instances`。

### AWS CLI

描述堆栈中已注册的 Amazon RDS 实例

以下示例描述了在指定堆栈中注册的 Amazon RDS 实例。

```
aws opsworks --region us-east-1 describe-rds-db-instances --stack-id  
d72553d4-8727-448c-9b00-f024f0ba1b06
```

输出：以下是包含一个已注册 RDS 实例的堆栈的输出。

```
{  
  "RdsDbInstances": [  
    {  
      "Engine": "mysql",  
      "StackId": "d72553d4-8727-448c-9b00-f024f0ba1b06",  
      "MissingOnRds": false,  
      "Region": "us-west-2",  
      "RdsDbInstanceArn": "arn:aws:rds:us-west-2:123456789012:db:clitestdb",  
      "DbPassword": "*****FILTERED*****",  
      "Address": "clitestdb.cdlqlk5uwd0k.us-west-2.rds.amazonaws.com",  
      "DbUser": "cliuser",  
      "DbInstanceIdentifier": "clitestdb"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的资源管理。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeRdsDbInstances](#)中的。

## describe-stack-provisioning-parameters

以下代码示例演示了如何使用 describe-stack-provisioning-parameters。

### AWS CLI

#### 返回堆栈的配置参数

以下describe-stack-provisioning-parameters示例返回指定堆栈的配置参数。配置参数包括代理安装位置和 OpsWorks 用于管理堆栈中实例上的代理的公钥等设置。

```
aws opsworks describe-stack-provisioning-parameters \  
  --stack-id 62744d97-6faf-4ecb-969b-a086fEXAMPLE
```

#### 输出：

```
{  
  "AgentInstallerUrl": "https://opsworks-instance-agent-us-  
west-2.s3.amazonaws.com/ID_number/opsworks-agent-installer.tgz",  
  "Parameters": {  
    "agent_installer_base_url": "https://opsworks-instance-agent-us-  
west-2.s3.amazonaws.com",  
    "agent_installer_tgz": "opsworks-agent-installer.tgz",  
    "assets_download_bucket": "opsworks-instance-assets-us-  
west-2.s3.amazonaws.com",  
    "charlie_public_key": "-----BEGIN PUBLIC KEY-----PUBLIC_KEY_EXAMPLE\n-----  
END PUBLIC KEY-----",  
    "instance_service_endpoint": "opsworks-instance-service.us-  
west-2.amazonaws.com",  
    "instance_service_port": "443",  
    "instance_service_region": "us-west-2",  
    "instance_service_ssl_verify_peer": "true",  
    "instance_service_use_ssl": "true",  
    "ops_works_endpoint": "opsworks.us-west-2.amazonaws.com",  
    "ops_works_port": "443",  
    "ops_works_region": "us-west-2",  
    "ops_works_ssl_verify_peer": "true",  
    "ops_works_use_ssl": "true",  
    "verbose": "false",  
    "wait_between_runs": "30"  
  }  
}
```

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的“[运行堆栈命令](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStackProvisioningParameters](#)中的。

## describe-stack-summary

以下代码示例演示了如何使用 describe-stack-summary。

### AWS CLI

#### 描述堆栈的配置

以下describe-stack-summary命令返回指定堆栈配置的摘要。

```
aws opsworks --region us-east-1 describe-stack-summary --stack-id 8c428b08-
a1a1-46ce-a5f8-feddc43771b8
```

输出：

```
{
  "StackSummary": {
    "StackId": "8c428b08-a1a1-46ce-a5f8-feddc43771b8",
    "InstancesCount": {
      "Booting": 1
    },
    "Name": "CLITest",
    "AppsCount": 1,
    "LayersCount": 1,
    "Arn": "arn:aws:opsworks:us-west-2:123456789012:stack/8c428b08-a1a1-46ce-a5f8-
feddc43771b8/"
  }
}
```

#### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的堆栈。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStackSummary](#)中的。

## describe-stacks

以下代码示例演示了如何使用 describe-stacks。

## AWS CLI

### 描述堆栈

以下describe-stacks命令描述了账户的堆栈。

```
aws opsworks --region us-east-1 describe-stacks
```

输出：

```
{
  "Stacks": [
    {
      "ServiceRoleArn": "arn:aws:iam::444455556666:role/aws-opsworks-service-role",
      "StackId": "aeb7523e-7c8b-49d4-b866-03aae9d4fbcf",
      "DefaultRootDeviceType": "instance-store",
      "Name": "TomStack-sd",
      "ConfigurationManager": {
        "Version": "11.4",
        "Name": "Chef"
      },
      "UseCustomCookbooks": true,
      "CustomJson": "{\n  \"tomcat\": {\n    \"base_version\": 7,\n    \"java_opts\n\": \"-Djava.awt.headless=true -Xmx256m\"\n  },\n  \"datasources\": {\n    \"ROOT\":\n  \"jdbc/mydb\"\n  }\n}",
      "Region": "us-east-1",
      "DefaultInstanceProfileArn": "arn:aws:iam::444455556666:instance-profile/aws-opsworks-ec2-role",
      "CustomCookbooksSource": {
        "Url": "git://github.com/example-repo/tomcustom.git",
        "Type": "git"
      },
      "DefaultAvailabilityZone": "us-east-1a",
      "HostnameTheme": "Layer_Dependent",
      "Attributes": {
        "Color": "rgb(45, 114, 184)"
      },
      "DefaultOs": "Amazon Linux",
      "CreatedAt": "2013-08-01T22:53:42+00:00"
    },
    {
      "ServiceRoleArn": "arn:aws:iam::444455556666:role/aws-opsworks-service-role",
      "StackId": "40738975-da59-4c5b-9789-3e422f2cf099",
```

```
"DefaultRootDeviceType": "instance-store",
  "Name": "MyStack",
  "ConfigurationManager": {
    "Version": "11.4",
    "Name": "Chef"
  },
  "UseCustomCookbooks": false,
  "Region": "us-east-1",
  "DefaultInstanceProfileArn": "arn:aws:iam::444455556666:instance-profile/aws-opsworks-ec2-role",
  "CustomCookbooksSource": {},
  "DefaultAvailabilityZone": "us-east-1a",
  "HostnameTheme": "Layer_Dependent",
  "Attributes": {
    "Color": "rgb(45, 114, 184)"
  },
  "DefaultOs": "Amazon Linux",
  "CreatedAt": "2013-10-25T19:24:30+00:00"
}
]
```

## 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的堆栈。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStacks](#)中的。

## describe-timebased-auto-scaling

以下代码示例演示了如何使用 describe-timebased-auto-scaling。

### AWS CLI

#### 描述实例的基于时间的扩展配置

以下示例描述了指定实例的基于时间的扩展配置。该实例由其实例 ID 进行标识，您可以在实例的详细信息页面上找到该实例 ID，也可以通过运行来找到describe-instances。

```
aws opsworks describe-time-based-auto-scaling --region us-east-1 --instance-ids
701f2ffe-5d8e-4187-b140-77b75f55de8d
```

输出：该示例只有一个基于时间的实例。

```
{
  "TimeBasedAutoScalingConfigurations": [
    {
      "InstanceId": "701f2ffe-5d8e-4187-b140-77b75f55de8d",
      "AutoScalingSchedule": {
        "Monday": {
          "11": "on",
          "10": "on",
          "13": "on",
          "12": "on"
        },
        "Tuesday": {
          "11": "on",
          "10": "on",
          "13": "on",
          "12": "on"
        }
      }
    }
  ]
}
```

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的基于时间的自动缩放的工作原理。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTimebasedAutoScaling](#)中的。

## describe-user-profiles

以下代码示例演示了如何使用 describe-user-profiles。

### AWS CLI

描述用户个人资料

以下describe-user-profiles命令描述了该账户的用户个人资料。

```
aws opsworks --region us-east-1 describe-user-profiles
```

输出：

```
{
```

```
"UserProfiles": [  
  {  
    "IamUserArn": "arn:aws:iam::123456789012:user/someuser",  
    "SshPublicKey": "ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEak0uP7i80q3Cko...",  
    "AllowSelfManagement": true,  
    "Name": "someuser",  
    "SshUsername": "someuser"  
  },  
  {  
    "IamUserArn": "arn:aws:iam::123456789012:user/cli-user-test",  
    "AllowSelfManagement": true,  
    "Name": "cli-user-test",  
    "SshUsername": "myusername"  
  }  
]  
}
```

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的管理AWS OpsWorks 用户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeUserProfiles](#)中的。

## describe-volumes

以下代码示例演示了如何使用 describe-volumes。

### AWS CLI

#### 描述堆栈的体积

以下示例描述了堆栈的 EBS 卷。

```
aws opsworks --region us-east-1 describe-volumes --stack-id 8c428b08-a1a1-46ce-a5f8-  
feddc43771b8
```

#### 输出：

```
{  
  "Volumes": [  
    {  
      "Status": "in-use",  
      "AvailabilityZone": "us-west-2a",
```



```
    "Name": "CLITest",
    "InstanceId": "dfe18b02-5327-493d-91a4-c5c0c448927f",
    "VolumeType": "standard",
    "VolumeId": "56b66fbd-e1a1-4aff-9227-70f77118d4c5",
    "Device": "/dev/sdi",
    "Ec2VolumeId": "vol-295c1638",
    "MountPoint": "/mnt/myvolume",
    "Size": 1
  }
]
```

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的资源管理。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeVolumes](#)中的。

## detach-elastic-load-balancer

以下代码示例演示了如何使用 detach-elastic-load-balancer。

### AWS CLI

将负载均衡器与其层分离

以下示例将以其名称标识的负载均衡器与其层分离。

```
aws opsworks --region us-east-1 detach-elastic-load-balancer --elastic-load-balancer-name Java-LB --layer-id 888c5645-09a5-4d0e-95a8-812ef1db76a4
```

输出：无。

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的 Elastic Load Balancing。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachElasticLoadBalancer](#)中的。

## disassociate-elastic-ip

以下代码示例演示了如何使用 disassociate-elastic-ip。

## AWS CLI

### 取消弹性 IP 地址与实例的关联

以下示例取消弹性 IP 地址与指定实例的关联。

```
aws opsworks --region us-east-1 disassociate-elastic-ip --elastic-ip 54.148.130.96
```

输出：无。

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的资源管理。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateElasticIp](#)中的。

## get-hostname-suggestion

以下代码示例演示了如何使用 get-hostname-suggestion。

## AWS CLI

### 获取图层的下一个主机名

以下示例获取指定层的下一个生成的主机名。本示例中使用的层是带有一个实例的 Java 应用程序服务器层。堆栈的主机名主题是默认的主机名主题，即 Layer\_Dependention。

```
aws opsworks --region us-east-1 get-hostname-suggestion --layer-id  
888c5645-09a5-4d0e-95a8-812ef1db76a4
```

输出：

```
{  
  "Hostname": "java-app2",  
  "LayerId": "888c5645-09a5-4d0e-95a8-812ef1db76a4"  
}
```

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的“创建新堆栈”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetHostnameSuggestion](#)中的。

## reboot-instance

以下代码示例演示了如何使用 `reboot-instance`。

### AWS CLI

#### 重启实例

以下示例重启实例。

```
aws opsworks --region us-east-1 reboot-instance --instance-id
dfe18b02-5327-493d-91a4-c5c0c448927f
```

输出：无。

#### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的重启实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RebootInstance](#)中的。

## register-elastic-ip

以下代码示例演示了如何使用 `register-elastic-ip`。

### AWS CLI

#### 向堆栈注册弹性 IP 地址

以下示例将弹性 IP 地址（由其 IP 地址标识）注册到指定的堆栈。

注意：弹性 IP 地址必须与堆栈位于同一区域。

```
aws opsworks register-elastic-ip --region us-east-1 --stack-id
d72553d4-8727-448c-9b00-f024f0ba1b06 --elastic-ip 54.148.130.96
```

#### 输出

```
{
  "ElasticIp": "54.148.130.96"
}
```

#### 更多信息

有关更多信息，请参阅OpsWorks 用户指南中的向堆栈注册弹性 IP 地址。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterElasticIp](#)中的。

## register-rds-db-instance

以下代码示例演示了如何使用 register-rds-db-instance。

### AWS CLI

使用堆栈注册 Amazon RDS 实例

以下示例向指定的堆栈注册一个 Amazon RDS 数据库实例，该实例由其亚马逊资源名称 (ARN) 标识。它还指定了实例的主用户名和密码。请注意，这并 AWS OpsWorks 不能验证这两个值中的任何一个。如果其中一个不正确，则您的应用程序将无法连接到数据库。

```
aws opsworks register-rds-db-instance --region us-east-1 --stack-id
d72553d4-8727-448c-9b00-f024f0ba1b06 --rds-db-instance-arn arn:aws:rds:us-
west-2:123456789012:db:clitestdb --db-user cliuser --db-password some23!pwd
```

输出：无。

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的使用堆栈注册 Amazon RDS 实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterRdsDbInstance](#)中的。

## register-volume

以下代码示例演示了如何使用 register-volume。

### AWS CLI

使用堆栈注册 Amazon EBS 卷

以下示例将一个由其卷 ID 标识的 Amazon EBS 卷注册到指定的堆栈。

```
aws opsworks register-volume --region us-east-1 --stack-id d72553d4-8727-448c-9b00-
f024f0ba1b06 --ec-2-volume-id vol-295c1638
```

输出：

```
{
  "VolumeId": "ee08039c-7cb7-469f-be10-40fb7f0c05e8"
}
```

## 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的使用堆栈注册 Amazon EBS 卷。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterVolume](#)中的。

## register

以下代码示例演示了如何使用 register。

### AWS CLI

#### 向堆栈注册实例

以下示例显示了向 AWS Opsworks 之外创建的堆栈注册实例的各种方法。您可以register从待注册的实例运行，也可以从单独的工作站运行。有关更多信息，请参阅AWS OpsWorks 用户指南中的注册 Amazon EC2 和本地实例。

注意：为简洁起见，示例省略了参数。region

#### 注册 Amazon EC2 实例

要表示您正在注册 EC2 实例，请将--infrastructure-class参数设置为ec2。

以下示例使用来自单独工作站的指定堆栈注册一个 EC2 实例。该实例通过其 EC2 ID 进行标识i-12345678。该示例使用工作站的默认 SSH 用户名，并尝试使用不需要密码的身份验证技术登录实例，例如默认 SSH 私钥。如果失败，则会register查询密码。

```
aws opsworks register --infrastructure-class=ec2 --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb i-12345678
```

以下示例使用来自不同工作站的指定堆栈注册一个 EC2 实例。它使用--ssh-username和--ssh-private-key参数明确指定命令用于登录实例的 SSH 用户名和私钥文件。ec2-user是亚马逊 Linux 实例的标准用户名。用ubuntu于 Ubuntu 实例。

```
aws opsworks register --infrastructure-class=ec2 --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb --ssh-username ec2-user --ssh-private-key ssh_private_key
i-12345678
```

以下示例注册了正在运行该`register`命令的 EC2 实例。使用 SSH 登录实例，然后`register`使用`--local`参数而不是实例 ID 或主机名运行。

```
aws opsworks register --infrastructure-class ec2 --stack-id 935450cc-61e0-4b03-a3e0-160ac817d2bb --local
```

## 注册本地实例

要表示您正在注册本地实例，请将`--infrastructure-class`参数设置为`on-premises`。

以下示例使用来自不同工作站的指定堆栈注册现有本地实例。实例由其 IP 地址标识`192.0.2.3`。该示例使用工作站的默认 SSH 用户名，并尝试使用不需要密码的身份验证技术登录实例，例如默认 SSH 私钥。如果失败，则会`register`查询密码。

```
aws opsworks register --infrastructure-class on-premises --stack-id 935450cc-61e0-4b03-a3e0-160ac817d2bb 192.0.2.3
```

以下示例使用来自不同工作站的指定堆栈注册本地实例。该实例由其主机名标识`host1`。`--override-...`参数分别显示 AWS OpsWorks`webserver1`为主机名`192.0.2.3`和`10.0.0.2`实例的公用和私有 IP 地址。

```
aws opsworks register --infrastructure-class on-premises --stack-id 935450cc-61e0-4b03-a3e0-160ac817d2bb --override-hostname webserver1 --override-public-ip 192.0.2.3 --override-private-ip 10.0.0.2 host1
```

以下示例使用来自不同工作站的指定堆栈注册本地实例。实例由其 IP 地址标识。`register`使用指定的 SSH 用户名和私钥文件登录实例。

```
aws opsworks register --infrastructure-class on-premises --stack-id 935450cc-61e0-4b03-a3e0-160ac817d2bb --ssh-username admin --ssh-private-key ssh_private_key 192.0.2.3
```

以下示例使用来自不同工作站的指定堆栈注册现有本地实例。该命令使用指定 SSH 密码和实例的 IP 地址的自定义 SSH 命令字符串登录实例。

```
aws opsworks register --infrastructure-class on-premises --stack-id 935450cc-61e0-4b03-a3e0-160ac817d2bb --override-ssh "sshpass -p 'mypassword' ssh your-user@192.0.2.3"
```

以下示例注册正在运行该register命令的本地实例。使用 SSH 登录实例，然后register使用--local参数而不是实例 ID 或主机名运行。

```
aws opsworks register --infrastructure-class on-premises --stack-id
935450cc-61e0-4b03-a3e0-160ac817d2bb --local
```

输出：以下是注册 EC2 实例的典型输出。

```
Warning: Permanently added '52.11.41.206' (ECDSA) to the list of known hosts.
% Total      % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left  Speed
100 6403k  100 6403k    0     0  2121k      0  0:00:03  0:00:03  --:--:-- 2121k
[Tue, 24 Feb 2015 20:48:37 +0000] opsworks-init: Initializing AWS OpsWorks
environment
[Tue, 24 Feb 2015 20:48:37 +0000] opsworks-init: Running on Ubuntu
[Tue, 24 Feb 2015 20:48:37 +0000] opsworks-init: Checking if OS is supported
[Tue, 24 Feb 2015 20:48:37 +0000] opsworks-init: Running on supported OS
[Tue, 24 Feb 2015 20:48:37 +0000] opsworks-init: Setup motd
[Tue, 24 Feb 2015 20:48:37 +0000] opsworks-init: Executing: ln -sf --backup /etc/
motd.opsworks-static /etc/motd
[Tue, 24 Feb 2015 20:48:37 +0000] opsworks-init: Enabling multiverse repositories
[Tue, 24 Feb 2015 20:48:37 +0000] opsworks-init: Customizing APT environment
[Tue, 24 Feb 2015 20:48:37 +0000] opsworks-init: Installing system packages
[Tue, 24 Feb 2015 20:48:37 +0000] opsworks-init: Executing: dpkg --configure -a
[Tue, 24 Feb 2015 20:48:37 +0000] opsworks-init: Executing with retry: apt-get
update
[Tue, 24 Feb 2015 20:49:13 +0000] opsworks-init: Executing: apt-get install -y ruby
ruby-dev libicu-dev libssl-dev libxslt-dev libxml2-dev libyaml-dev monit
[Tue, 24 Feb 2015 20:50:13 +0000] opsworks-init: Using assets bucket from
environment: 'opsworks-instance-assets-us-east-1.s3.amazonaws.com'.
[Tue, 24 Feb 2015 20:50:13 +0000] opsworks-init: Installing Ruby for the agent
[Tue, 24 Feb 2015 20:50:13 +0000] opsworks-init: Executing: /tmp/opsworks-
agent-installer.YgGq8wF3UUre6yDy/opsworks-agent-installer/opsworks-agent/bin/
installer_wrapper.sh -r -R opsworks-instance-assets-us-east-1.s3.amazonaws.com
[Tue, 24 Feb 2015 20:50:44 +0000] opsworks-init: Starting the installer
Instance successfully registered. Instance ID: 4d6d1710-ded9-42a1-b08e-b043ad7af1e2
Connection to 52.11.41.206 closed.
```

## 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的向 AWS OpsWorks 堆栈注册实例。

- 有关 API 的详细信息，请参阅在《AWS CLI 命令参考》中[注册](#)。

## set-load-based-auto-scaling

以下代码示例演示了如何使用 `set-load-based-auto-scaling`。

### AWS CLI

为图层设置基于负载的缩放配置

以下示例为指定层启用基于负载的缩放并设置该层的配置。必须使用 `create-instance` 向层中添加基于负载的实例。

```
aws opsworks --region us-east-1 set-load-based-auto-scaling --layer-id
523569ae-2faf-47ac-b39e-f4c4b381f36d --enable --up-scaling file://upscale.json --
down-scaling file://downscale.json
```

该示例将升级阈值设置放在名为的工作目录中的一个单独文件中 `upscale.json`，该文件包含以下内容。

```
{
  "InstanceCount": 2,
  "ThresholdsWaitTime": 3,
  "IgnoreMetricsTime": 3,
  "CpuThreshold": 85,
  "MemoryThreshold": 85,
  "LoadThreshold": 85
}
```

该示例将缩减阈值设置放在名为的工作目录中的一个单独文件中 `downscale.json`，该文件包含以下内容。

```
{
  "InstanceCount": 2,
  "ThresholdsWaitTime": 3,
  "IgnoreMetricsTime": 3,
  "CpuThreshold": 35,
  "MemoryThreshold": 30,
  "LoadThreshold": 30
}
```

输出：无。

[更多信息](#)



有关更多信息，请参阅AWS OpsWorks 用户指南中的使用基于负载的自动扩展。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetLoadBasedAutoScaling](#)中的。

## set-permission

以下代码示例演示了如何使用 set-permission。

### AWS CLI

授予每个堆栈的 AWS OpsWorks 权限级别

当您通过create-user-profile调用将 AWS 身份和访问管理 (IAM) Management 用户导入 AWS OpsWorks 时，该用户仅拥有由所附的 IAM 策略授予的权限。您可以通过修改用户的策略来授予 AWS OpsWorks 权限。但是，对于用户需要访问的每个堆栈，导入用户，然后使用set-permission命令向用户授予一个标准权限级别，通常会更容易。

以下示例向由 Amazon 资源名称 (ARN) 标识的用户授予对指定堆栈的权限。该示例向用户授予管理权限级别，在堆栈的实例上具有 sudo 和 SSH 权限。

```
aws opsworks set-permission --region us-east-1 --stack-id 71c7ca72-55ae-4b6a-8ee1-
a8dcded3fa0f --level manage --iam-user-arn arn:aws:iam::123456789102:user/cli-user-
test --allow-ssh --allow-sudo
```

输出：无。

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的向用户授予每堆栈权限。AWS OpsWorks

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetPermission](#)中的。

## set-time-based-auto-scaling

以下代码示例演示了如何使用 set-time-based-auto-scaling。

### AWS CLI

为图层设置基于时间的缩放配置

以下示例为指定实例设置基于时间的配置。必须先使用将实例create-instance添加到图层中。

```
aws opsworks --region us-east-1 set-time-based-auto-scaling --instance-id
69b6237c-08c0-4edb-a6af-78f3d01cedf2 --auto-scaling-schedule file://schedule.json
```

该示例将计划放在工作目录中名为的单独文件中schedule.json。在此示例中，实例在星期一和星期二中午 UTC（协调世界时）左右开启了几个小时。

```
{
  "Monday": {
    "10": "on",
    "11": "on",
    "12": "on",
    "13": "on"
  },
  "Tuesday": {
    "10": "on",
    "11": "on",
    "12": "on",
    "13": "on"
  }
}
```

输出：无。

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的使用基于时间的自动缩放。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetTimeBasedAutoScaling](#)中的。

## start-instance

以下代码示例演示了如何使用 start-instance。

### AWS CLI

#### 启动实例

以下start-instance命令启动指定的全天候实例。

```
aws opsworks start-instance --instance-id f705ee48-9000-4890-8bd3-20eb05825aaf
```

输出：无。使用描述实例来检查实例的状态。

提示您可以调用 `start-stack`，使用一条命令启动堆栈中的每个离线实例。

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的手动启动、停止和重启全天候实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartInstance](#)中的。

## **start-stack**

以下代码示例演示了如何使用 `start-stack`。

### AWS CLI

#### 启动堆栈的实例

以下示例启动堆栈的所有全天候实例。要启动特定实例，请使用`start-instance`。

```
aws opsworks --region us-east-1 start-stack --stack-id 8c428b08-a1a1-46ce-a5f8-  
feddc43771b8
```

输出：无。

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的启动实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartStack](#)中的。

## **stop-instance**

以下代码示例演示了如何使用 `stop-instance`。

### AWS CLI

#### 停止实例

以下示例停止了由其实例 ID 标识的指定实例。您可以通过访问 AWS OpsWorks 控制台上的实例详细信息页面或运行`describe-instances`命令来获取实例 ID。

```
aws opsworks stop-instance --region us-east-1 --instance-id 3a21cfac-4a1f-4ce2-a921-  
b2cfba6f7771
```

您可以通过调用`start-instance`或通过调用删除实例来重启已停止的实例`delete-instance`。

输出：无。

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的停止实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopInstance](#)中的。

## stop-stack

以下代码示例演示了如何使用 `stop-stack`。

### AWS CLI

#### 停止堆栈的实例

以下示例停止堆栈的所有全天候实例。要停止特定实例，请使用`stop-instance`。

```
aws opsworks --region us-east-1 stop-stack --stack-id 8c428b08-a1a1-46ce-a5f8-  
feddc43771b8
```

输出：无输出。

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的停止实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopStack](#)中的。

## unassign-instance

以下代码示例演示了如何使用 `unassign-instance`。

### AWS CLI

#### 取消分配已注册实例的图层

以下`unassign-instance`命令将实例从其附加层中取消分配。

```
aws opsworks --region us-east-1 unassign-instance --instance-id 4d6d1710-ded9-42a1-  
b08e-b043ad7af1e2
```

输出：无。

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的取消分配注册实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UnassignInstance](#)中的。

## unassign-volume

以下代码示例演示了如何使用 unassign-volume。

### AWS CLI

#### 取消对卷实例的分配

以下示例从其实例中取消分配已注册的亚马逊弹性区块存储 (Amazon EBS) Block Store 卷。该卷由其卷 ID 标识，这是您在向堆栈注册卷时 AWS OpsWorks 分配的 GUID，而不是亚马逊弹性计算云 (Amazon EC2) 卷 ID。

```
aws opsworks --region us-east-1 unassign-volume --volume-id 8430177d-52b7-4948-9c62-e195af4703df
```

输出：无。

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的取消分配 Amazon EBS 卷。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UnassignVolume](#)中的。

## update-app

以下代码示例演示了如何使用 update-app。

### AWS CLI

#### 更新应用程序

以下示例更新了指定的应用程序以更改其名称。

```
aws opsworks --region us-east-1 update-app --app-id 26a61ead-d201-47e3-b55c-2a7c666942f8 --name NewAppName
```

输出：无。

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的编辑应用程序。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateApp](#)中的。

## update-elastic-ip

以下代码示例演示了如何使用 update-elastic-ip。

### AWS CLI

#### 更新弹性 IP 地址名称

以下示例更新了指定弹性 IP 地址的名称。

```
aws opsworks --region us-east-1 update-elastic-ip --elastic-ip 54.148.130.96 --name  
NewIPName
```

输出：无。

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的资源管理。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateElasticIp](#)中的。

## update-instance

以下代码示例演示了如何使用 update-instance。

### AWS CLI

#### 更新实例

以下示例更新了指定实例的类型。

```
aws opsworks --region us-east-1 update-instance --instance-id  
dfe18b02-5327-493d-91a4-c5c0c448927f --instance-type c3.xlarge
```

输出：无。

## 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的编辑实例配置。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateInstance](#)中的。

## update-layer

以下代码示例演示了如何使用 update-layer。

### AWS CLI

#### 更新图层

以下示例将指定层更新为使用 Amazon EBS 优化实例。

```
aws opsworks --region us-east-1 update-layer --layer-id
888c5645-09a5-4d0e-95a8-812ef1db76a4 --use-efs-optimized-instances
```

输出：无。

#### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的编辑 OpsWorks 图层配置。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLayer](#)中的。

## update-my-user-profile

以下代码示例演示了如何使用 update-my-user-profile。

### AWS CLI

#### 更新用户的个人资料

以下示例更新development用户的配置文件以使用指定的 SSH 公钥。用户的 AWS 凭据由credentials文件 (~\.aws\credentials) 中的development配置文件表示，密钥位于工作目录中的.pem文件中。

```
aws opsworks --region us-east-1 --profile development update-my-user-profile --ssh-
public-key file:///development_key.pem
```

输出：无。

## 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的编辑AWS OpsWorks 用户设置。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateMyUserProfile](#)中的。

## update-rds-db-instance

以下代码示例演示了如何使用 `update-rds-db-instance`。

### AWS CLI

#### 更新已注册的 Amazon RDS 数据库实例

以下示例更新了 Amazon RDS 实例的主密码值。请注意，此命令不会更改 RDS 实例的主密码，只会更改您向其提供的密码 AWS OpsWorks。如果此密码与 RDS 实例的密码不匹配，则您的应用程序将无法连接到数据库。

```
aws opsworks --region us-east-1 update-rds-db-instance --db-password 123456789
```

输出：无。

## 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的使用堆栈注册 Amazon RDS 实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRdsDbInstance](#)中的。

## update-volume

以下代码示例演示了如何使用 `update-volume`。

### AWS CLI

#### 更新已注册的卷

以下示例更新已注册的亚马逊 Elastic Block Store (Amazon EBS) 卷的挂载点。该卷由其卷 ID 标识，这是您在向堆栈注册卷时 AWS OpsWorks 分配给该卷的 GUID，而不是亚马逊弹性计算云 (Amazon EC2) 卷 ID。：

```
aws opsworks --region us-east-1 update-volume --volume-id 8430177d-52b7-4948-9c62-e195af4703df --mount-point /mnt/myvol
```



输出：无。

### 更多信息

有关更多信息，请参阅AWS OpsWorks 用户指南中的将 Amazon EBS 卷分配给实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateVolume](#)中的。

## AWS OpsWorks CM 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS OpsWorks CM。 AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **associate-node**

以下代码示例演示了如何使用 associate-node。

### AWS CLI

#### 关联节点

以下associate-node命令将名为的节点i-44de882p与名为的 Chef Automate 服务器相关联automate-06，这意味着automate-06服务器管理该节点，并通过通过 associate-node 命令安装在节点上的chef-client代理软件将配方命令传达给该节点。有效的节点名称是 EC2 实例 ID。：

```
aws opsworks-cm associate-node --server-name "automate-06" --node-name
  "i-43de882p" --engine-attributes "Name=CHEF_ORGANIZATION,Value='MyOrganization'
  Name=CHEF_NODE_PUBLIC_KEY,Value='Public_key_contents'"
```

该命令返回的输出类似于以下内容。输出：

```
{
  "NodeAssociationStatusToken": "AHUY8wFe4pdXtZC5DiJa5S0Lp5o14DH//
rHRqHDWXxwVoNBxcEy4V7R0N0Fymh7E/1Hum0BPsemPQFE6dcGaiFk"
}
```

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的 AWS OpsWorks “Chef Automate 自动添加节点”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateNode](#) 中的。

## create-backup

以下代码示例演示了如何使用 create-backup。

### AWS CLI

#### 创建备份

以下 create-backup 命令启动 us-east-1 对该区域 automate-06 中名为 Chef Automate 服务器的手动备份。该命令在 --description 参数中向备份中添加一条描述性消息。

```
aws opsworks-cm create-backup \
  --server-name 'automate-06' \
  --description "state of my infrastructure at launch"
```

输出显示了与以下内容类似的新备份的信息。

输出：

```
{
  "Backups": [
    {
      "BackupArn": "string",
      "BackupId": "automate-06-20160729133847520",
      "BackupType": "MANUAL",
      "CreatedAt": "2016-07-29T13:38:47.520Z",
      "Description": "state of my infrastructure at launch",
      "Engine": "Chef",
      "EngineModel": "Single",
    }
  ]
}
```

```

        "EngineVersion": "12",
        "InstanceProfileArn": "arn:aws:iam::1019881987024:instance-profile/
automate-06-1010V4UU2WRM2",
        "InstanceType": "m4.large",
        "KeyPair": "",
        "PreferredBackupWindow": "",
        "PreferredMaintenanceWindow": "",
        "S3LogUrl": "https://s3.amazonaws.com/<bucket-name>/
automate-06-20160729133847520",
        "SecurityGroupIds": [ "sg-1a24c270" ],
        "ServerName": "automate-06",
        "ServiceRoleArn": "arn:aws:iam::1019881987024:role/aws-opsworks-cm-
service-role.1114810729735",
        "Status": "OK",
        "StatusDescription": "",
        "SubnetIds": [ "subnet-49436a18" ],
        "ToolsVersion": "string",
        "UserArn": "arn:aws:iam::1019881987024:user/opsworks-user"
    }
],
}

```

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的“备份和恢复 f AWS OpsWorks or Chef Automate 服务器”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateBackup](#) 中的。

## create-server

以下代码示例演示了如何使用 create-server。

### AWS CLI

#### 创建服务器

以下 create-server 示例创建了一个 automate-06 在您的默认区域中命名的新 Chef Automate 服务器。请注意，大多数其他设置都使用默认值，例如要保留的备份数量以及维护和备份的开始时间。在运行 create-server 命令之前，请完成 AWS Opsworks for Chef Automate 用户指南中 Chef Automate [入门](#) 中的先决条件。AWS OpsWorks

```

aws opsworks-cm create-server \
  --engine "Chef" \
  --engine-model "Single" \

```

```
--engine-version "12" \  
--server-name "automate-06" \  
--instance-profile-arn "arn:aws:iam::1019881987024:instance-profile/aws-  
opsworks-cm-ec2-role" \  
--instance-type "t2.medium" \  
--key-pair "amazon-test" \  
--service-role-arn "arn:aws:iam::044726508045:role/aws-opsworks-cm-service-role"
```

输出显示的有关新服务器的信息与以下内容类似：

```
{  
  "Server": {  
    "BackupRetentionCount": 10,  
    "CreatedAt": 2016-07-29T13:38:47.520Z,  
    "DisableAutomatedBackup": FALSE,  
    "Endpoint": "https://opsworks-cm.us-east-1.amazonaws.com",  
    "Engine": "Chef",  
    "EngineAttributes": [  
      {  
        "Name": "CHEF_DELIVERY_ADMIN_PASSWORD",  
        "Value": "1Password1"  
      }  
    ],  
    "EngineModel": "Single",  
    "EngineVersion": "12",  
    "InstanceProfileArn": "arn:aws:iam::1019881987024:instance-profile/aws-  
opsworks-cm-ec2-role",  
    "InstanceType": "t2.medium",  
    "KeyPair": "amazon-test",  
    "MaintenanceStatus": "",  
    "PreferredBackupWindow": "Sun:02:00",  
    "PreferredMaintenanceWindow": "00:00",  
    "SecurityGroupIds": [ "sg-1a24c270" ],  
    "ServerArn": "arn:aws:iam::1019881987024:instance/  
automate-06-1010V4UU2WRM2",  
    "ServerName": "automate-06",  
    "ServiceRoleArn": "arn:aws:iam::1019881987024:role/aws-opsworks-cm-service-  
role",  
    "Status": "CREATING",  
    "StatusReason": "",  
    "SubnetIds": [ "subnet-49436a18" ]  
  }  
}
```

有关更多信息，请参阅 [for Chef Automate API 参考UpdateServer](#) 中的。AWS OpsWorks

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考CreateServer](#) 中的。

## delete-backup

以下代码示例演示了如何使用 delete-backup。

### AWS CLI

#### 删除备份

以下delete-backup命令删除由备份 ID 标识的 Chef Automate 服务器的手动或自动备份。当您接近可以保存的最大备份数量或想要最大限度地降低 Amazon S3 存储成本时，此命令非常有用。：

```
aws opsworks-cm delete-backup --backup-id "automate-06-2016-11-19T23:42:40.240Z"
```

输出显示备份删除是否成功。

#### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的“备份和恢复 f AWS OpsWorks or Chef Automate 服务器”。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考DeleteBackup](#) 中的。

## delete-server

以下代码示例演示了如何使用 delete-server。

### AWS CLI

#### 删除服务器

以下delete-server命令删除由服务器名称标识的 Chef Automate 服务器。服务器被删除后，DescribeServer请求将不再返回该服务器。：

```
aws opsworks-cm delete-server --server-name "automate-06"
```

输出显示服务器删除是否成功。

## 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的删除 Chef Automate 服务器。AWS OpsWorks

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteServer](#)中的。

## describe-account-attributes

以下代码示例演示了如何使用 describe-account-attributes。

### AWS CLI

#### 描述账户属性

以下describe-account-attributes命令返回有关您的账户使用 Chef Automate 资源的信息。AWS OpsWorks：

```
aws opsworks-cm describe-account-attributes
```

该命令返回的每个账户属性条目的输出如下所示。输出：

```
{
  "Attributes": [
    {
      "Maximum": 5,
      "Name": "ServerLimit",
      "Used": 2
    }
  ]
}
```

## 更多信息

有关更多信息，请参阅 for Chef Automate API 参考 DescribeAccountAttributes 中的。AWS OpsWorks

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAccountAttributes](#)中的。

## describe-backups

以下代码示例演示了如何使用 describe-backups。

## AWS CLI

### 描述备份

以下describe-backups命令返回有关默认区域中与您的账户关联的所有备份的信息。

```
aws opsworks-cm describe-backups
```

该命令返回的每个备份条目的输出类似于以下内容。

输出：

```
{
  "Backups": [
    {
      "BackupArn": "string",
      "BackupId": "automate-06-20160729133847520",
      "BackupType": "MANUAL",
      "CreatedAt": "2016-07-29T13:38:47.520Z",
      "Description": "state of my infrastructure at launch",
      "Engine": "Chef",
      "EngineModel": "Single",
      "EngineVersion": "12",
      "InstanceProfileArn": "arn:aws:iam::1019881987024:instance-profile/
automate-06-1010V4UU2WRM2",
      "InstanceType": "m4.large",
      "KeyPair": "",
      "PreferredBackupWindow": "",
      "PreferredMaintenanceWindow": "",
      "S3LogUrl": "https://s3.amazonaws.com/<bucket-name>/
automate-06-20160729133847520",
      "SecurityGroupIds": [ "sg-1a24c270" ],
      "ServerName": "automate-06",
      "ServiceRoleArn": "arn:aws:iam::1019881987024:role/aws-opsworks-cm-
service-role.1114810729735",
      "Status": "Successful",
      "StatusDescription": "",
      "SubnetIds": [ "subnet-49436a18" ],
      "ToolsVersion": "string",
      "UserArn": "arn:aws:iam::1019881987024:user/opsworks-user"
    }
  ],
}
```

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的“备份和恢复 f AWS OpsWorks or Chef Automate 服务器”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeBackups](#)中的。

## describe-events

以下代码示例演示了如何使用 describe-events。

### AWS CLI

#### 描述事件

以下describe-events示例返回与指定的 Chef Automate 服务器关联的所有事件的相关信息。

```
aws opsworks-cm describe-events \  
  --server-name 'automate-06'
```

该命令返回的每个事件条目的输出与以下示例类似：

```
{  
  "ServerEvents": [  
    {  
      "CreatedAt": "2016-07-29T13:38:47.520Z",  
      "LogUrl": "https://s3.amazonaws.com/<bucket-name>/  
automate-06-20160729133847520",  
      "Message": "Updates successfully installed.",  
      "ServerName": "automate-06"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的“[一般故障排除提示](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEvents](#)中的。

## describe-node-association-status

以下代码示例演示了如何使用 describe-node-association-status。

### AWS CLI

#### 描述节点关联状态



以下describe-node-association-status命令返回将节点与名为的 Chef Automate 服务器关联的请求的状态automate-06。：

```
aws opsworks-cm describe-node-association-status --server-  
name "automate-06" --node-association-status-token "Af1JKl+/  
GoKLZJBdDQEx0065CDi57b1Qe9nKM8joSok0pQ9xr8DqApBN9/106sLdSv1fDEKkEx+eoCHvjowHa0s="
```

该命令返回的每个账户属性条目的输出如下所示。输出：

```
{  
  "NodeAssociationStatus": "IN_PROGRESS"  
}
```

### 更多信息

有关更多信息，请参阅 [for Chef Automate API 参考 DescribeNodeAssociationStatus](#) 中的。AWS OpsWorks

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeNodeAssociationStatus](#)中的。

## describe-servers

以下代码示例演示了如何使用 describe-servers。

### AWS CLI

#### 描述服务器

以下describe-servers命令返回与您的账户关联的所有服务器以及您的默认区域中的所有服务器的信息。：

```
aws opsworks-cm describe-servers
```

该命令返回的每个服务器条目的输出类似于以下内容。输出：

```
{  
  "Servers": [  
    {  
      "BackupRetentionCount": 8,  
      "CreatedAt": 2016-07-29T13:38:47.520Z,
```

```
    "DisableAutomatedBackup": FALSE,
    "Endpoint": "https://opsworks-cm.us-east-1.amazonaws.com",
    "Engine": "Chef",
    "EngineAttributes": [
      {
        "Name": "CHEF_DELIVERY_ADMIN_PASSWORD",
        "Value": "1Password1"
      }
    ],
    "EngineModel": "Single",
    "EngineVersion": "12",
    "InstanceProfileArn": "arn:aws:iam::1019881987024:instance-profile/
automate-06-1010V4UU2WRM2",
    "InstanceType": "m4.large",
    "KeyPair": "",
    "MaintenanceStatus": "SUCCESS",
    "PreferredBackupWindow": "03:00",
    "PreferredMaintenanceWindow": "Mon:09:00",
    "SecurityGroupIds": [ "sg-1a24c270" ],
    "ServerArn": "arn:aws:iam::1019881987024:instance/automate-06-1010V4UU2WRM2",
    "ServerName": "automate-06",
    "ServiceRoleArn": "arn:aws:iam::1019881987024:role/aws-opsworks-cm-service-
role.1114810729735",
    "Status": "HEALTHY",
    "StatusReason": "",
    "SubnetIds": [ "subnet-49436a18" ]
  }
]
}
```

## 更多信息

有关更多信息，请参阅 [for Chef Automate API 指南 DescribeServers](#) 中的。AWS OpsWorks

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 DescribeServers](#) 中的。

## disassociate-node

以下代码示例演示了如何使用 disassociate-node。

### AWS CLI

#### 取消与节点的关联

以下 `disassociate-node` 命令取消名为的节点的关联 `i-44de882p`，将该节点从名为 `Chef Automate` 服务器的管理中删除。 `automate-06` 有效的节点名称是 EC2 实例 ID。：

```
aws opsworks-cm disassociate-node --server-name "automate-06" --node-name
  "i-43de882p" --engine-attributes "Name=CHEF_ORGANIZATION,Value='MyOrganization'
  Name=CHEF_NODE_PUBLIC_KEY,Value='Public_key_contents'"
```

该命令返回的输出类似于以下内容。输出：

```
{
  "NodeAssociationStatusToken": "AHUY8wFe4pdXtZC5DiJa5S0Lp5o14DH//
  rHRqHDWXxwVoNBxcEy4V7R0N0Fymh7E/1Hum0BPsemPQFE6dcGaiFk"
}
```

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的删除 Chef Automate 服务器。AWS OpsWorks

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateNode](#) 中的。

## restore-server

以下代码示例演示了如何使用 `restore-server`。

### AWS CLI

#### 恢复服务器

以下 `restore-server` 命令从 ID 为的备份就地恢复默认区域 `automate-06` 中命名的 Chef Automate 服务器。 `automate-06-2016-11-22T16:13:27.998Z` 恢复服务器会恢复与执行指定备份时 Chef Automate 服务器正在管理的节点连接。

```
aws opsworks-cm restore-server --backup-id "automate-06-2016-11-22T 16:13:27.998 Z" --server
  名称 "automate-06"
```

输出仅为命令 ID。输出：

```
(None)
```

### 更多信息

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的“还原失败 AWS OpsWorks 的 Chef Automate 服务器”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RestoreServer](#) 中的。

## start-maintenance

以下代码示例演示了如何使用 start-maintenance。

### AWS CLI

#### 要开始维护

以下 start-maintenance 示例手动启动默认区域中指定的 Chef Automate 或 Puppet Enterprise 服务器的维护。如果之前的自动维护尝试失败，并且维护失败的根本原因已得到解决，则此命令很有用。

```
aws opsworks-cm start-maintenance \  
  --server-name 'automate-06'
```

输出：

```
{  
  "Server": {  
    "AssociatePublicIpAddress": true,  
    "BackupRetentionCount": 10,  
    "ServerName": "automate-06",  
    "CreatedAt": 1569229584.842,  
    "CloudFormationStackArn": "arn:aws:cloudformation:us-  
west-2:123456789012:stack/aws-opsworks-cm-instance-automate-06-1606611794746/  
EXAMPLE0-31de-11eb-bdb0-0a5b0a1353b8",  
    "DisableAutomatedBackup": false,  
    "Endpoint": "automate-06-EXAMPLEvr8gjfk5f.us-west-2.opsworks-cm.io",  
    "Engine": "ChefAutomate",  
    "EngineModel": "Single",  
    "EngineAttributes": [],  
    "EngineVersion": "2020-07",  
    "InstanceProfileArn": "arn:aws:iam::123456789012:instance-profile/aws-  
opsworks-cm-ec2-role",  
    "InstanceType": "m5.large",  
    "PreferredMaintenanceWindow": "Sun:01:00",  
    "PreferredBackupWindow": "Sun:15:00",  
    "SecurityGroupIds": [  

```

```

        "sg-EXAMPLE"
    ],
    "ServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/aws-opsworks-
cm-service-role",
    "Status": "UNDER_MAINTENANCE",
    "SubnetIds": [
        "subnet-EXAMPLE"
    ],
    "ServerArn": "arn:aws:opsworks-cm:us-west-2:123456789012:server/
automate-06/0148382d-66b0-4196-8274-d1a2b6dff8d1"
    }
}

```

有关更多信息，请参阅《AWS OpsWorks 用户指南》中的[系统维护 \( Puppet Enterprise 服务器 \)](#) 或[系统维护 \( Chef Automate 服务器 \)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[StartMaintenance](#)中的。

## update-server-engine-attributes

以下代码示例演示了如何使用 `update-server-engine-attributes`。

### AWS CLI

#### 更新服务器引擎属性

以下 `update-server-engine-attributes` 命令更新名为 Chef Automate 服务器的 `CHEF_PIVOTAL_KEY` 引擎属性的值 `automate-06`。目前无法更改其他引擎属性的值。

```

aws opsworks-cm update-server-engine-attributes \
  --attribute-name CHEF_PIVOTAL_KEY \
  --attribute-value "new key value" \
  --server-name "automate-06"

```

输出显示了与以下内容类似的有关更新的服务器的信息。

```

{
  "Server": {
    "BackupRetentionCount": 2,
    "CreatedAt": 2016-07-29T13:38:47.520Z,
    "DisableAutomatedBackup": FALSE,
    "Endpoint": "https://opsworks-cm.us-east-1.amazonaws.com",
    "Engine": "Chef",

```

```

    "EngineAttributes": [
      {
        "Name": "CHEF_PIVOTAL_KEY",
        "Value": "new key value"
      }
    ],
    "EngineModel": "Single",
    "EngineVersion": "12",
    "InstanceProfileArn": "arn:aws:iam::1019881987024:instance-profile/
automate-06-1010V4UU2WRM2",
    "InstanceType": "m4.large",
    "KeyPair": "",
    "MaintenanceStatus": "SUCCESS",
    "PreferredBackupWindow": "Mon:09:15",
    "PreferredMaintenanceWindow": "03:00",
    "SecurityGroupIds": [ "sg-1a24c270" ],
    "ServerArn": "arn:aws:iam::1019881987024:instance/
automate-06-1010V4UU2WRM2",
    "ServerName": "automate-06",
    "ServiceRoleArn": "arn:aws:iam::1019881987024:role/aws-opsworks-cm-service-
role.1114810729735",
    "Status": "HEALTHY",
    "StatusReason": "",
    "SubnetIds": [ "subnet-49436a18" ]
  }
}

```

有关更多信息，请参阅 for Chef Automate API 参考[UpdateServerEngineAttributes](#)中的。AWS OpsWorks

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateServerEngineAttributes](#)中的。

## update-server

以下代码示例演示了如何使用 update-server。

### AWS CLI

#### 更新服务器

以下update-server命令更新默认区域中指定 Chef Automate 服务器的维护开始时间。添加该--preferred-maintenance-window参数是为了将服务器维护的开始日期和时间更改为星期一上午 9:15。世界标准时间。：

```
aws opsworks-cm update-server \  
  --server-name "automate-06" \  
  --preferred-maintenance-window "Mon:09:15"
```

输出显示了与以下内容类似的有关更新的服务器的信息。

```
{  
  "Server": {  
    "BackupRetentionCount": 8,  
    "CreatedAt": 2016-07-29T13:38:47.520Z,  
    "DisableAutomatedBackup": TRUE,  
    "Endpoint": "https://opsworks-cm.us-east-1.amazonaws.com",  
    "Engine": "Chef",  
    "EngineAttributes": [  
      {  
        "Name": "CHEF_DELIVERY_ADMIN_PASSWORD",  
        "Value": "1Password1"  
      }  
    ],  
    "EngineModel": "Single",  
    "EngineVersion": "12",  
    "InstanceProfileArn": "arn:aws:iam::1019881987024:instance-profile/  
automate-06-1010V4UU2WRM2",  
    "InstanceType": "m4.large",  
    "KeyPair": "",  
    "MaintenanceStatus": "OK",  
    "PreferredBackupWindow": "Mon:09:15",  
    "PreferredMaintenanceWindow": "03:00",  
    "SecurityGroupIds": [ "sg-1a24c270" ],  
    "ServerArn": "arn:aws:iam::1019881987024:instance/  
automate-06-1010V4UU2WRM2",  
    "ServerName": "automate-06",  
    "ServiceRoleArn": "arn:aws:iam::1019881987024:role/aws-opsworks-cm-service-  
role.1114810729735",  
    "Status": "HEALTHY",  
    "StatusReason": "",  
    "SubnetIds": [ "subnet-49436a18" ]  
  }  
}
```

有关更多信息，请参阅 for Chef Automate API 参考[UpdateServer](#)中的。AWS OpsWorks

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateServer](#)中的。

## 使用 Organizati AWS CLI

以下代码示例向您展示了如何使用 with Organizations 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### accept-handshake

以下代码示例演示了如何使用 accept-handshake。

AWS CLI

接受来自其他账户的握手

组织的所有者比尔此前曾邀请胡安的账户加入他的组织。以下示例显示 Juan 的账户接受了握手并因此同意了邀请。

```
aws organizations accept-handshake --handshake-id h-examplehandshakeid111
```

输出显示以下内容：

```
{
  "Handshake": {
    "Action": "INVITE",
    "Arn": "arn:aws:organizations::111111111111:handshake/o-exampleorgid/invite/h-examplehandshakeid111",
    "RequestedTimestamp": 1481656459.257,
    "ExpirationTimestamp": 1482952459.257,
    "Id": "h-examplehandshakeid111",
```



```
    "Parties": [
      {
        "Id": "o-exampleorgid",
        "Type": "ORGANIZATION"
      },
      {
        "Id": "juan@example.com",
        "Type": "EMAIL"
      }
    ],
    "Resources": [
      {
        "Resources": [
          {
            "Type": "MASTER_EMAIL",
            "Value": "bill@amazon.com"
          },
          {
            "Type": "MASTER_NAME",
            "Value": "Org Master Account"
          },
          {
            "Type": "ORGANIZATION_FEATURE_SET",
            "Value": "ALL"
          }
        ],
        "Type": "ORGANIZATION",
        "Value": "o-exampleorgid"
      },
      {
        "Type": "EMAIL",
        "Value": "juan@example.com"
      }
    ],
    "State": "ACCEPTED"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AcceptHandshake](#)中的。

## attach-policy

以下代码示例演示了如何使用 attach-policy。

## AWS CLI

将策略附加到根、OU 或账户

### 示例 1

以下示例演示如何将服务控制策略 ( SCP ) 附加到 OU :

```
aws organizations attach-policy
    --policy-id p-examplepolicyid111
    --target-id ou-examplerootid111-exampleouid111
```

### 示例 2

以下示例演示如何将服务控制策略直接附加到账户 :

```
aws organizations attach-policy
    --policy-id p-examplepolicyid111
    --target-id 333333333333
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachPolicy](#)中的。

## cancel-handshake

以下代码示例演示了如何使用 cancel-handshake。

## AWS CLI

取消从其他账户发送的握手

Bill 之前曾向 Susan 的账户发送过加入其组织的邀请。他改变了主意，决定在苏珊接受邀请之前取消邀请。以下示例显示了 Bill 的取消：

```
aws organizations cancel-handshake --handshake-id h-examplehandshakeid111
```

输出包括一个握手对象，该对象显示现在CANCELED的状态为：

```
{
  "Handshake": {
    "Id": "h-examplehandshakeid111",
    "State": "CANCELED",
    "Action": "INVITE",
```

```

    "Arn": "arn:aws:organizations::111111111111:handshake/o-
exampleorgid/invite/h-examplehandshakeid111",
    "Parties": [
        {
            "Id": "o-exampleorgid",
            "Type": "ORGANIZATION"
        },
        {
            "Id": "susan@example.com",
            "Type": "EMAIL"
        }
    ],
    "Resources": [
        {
            "Type": "ORGANIZATION",
            "Value": "o-exampleorgid",
            "Resources": [
                {
                    "Type": "MASTER_EMAIL",
                    "Value": "bill@example.com"
                },
                {
                    "Type": "MASTER_NAME",
                    "Value": "Master Account"
                },
                {
                    "Type": "ORGANIZATION_FEATURE_SET",
                    "Value": "CONSOLIDATED_BILLING"
                }
            ]
        },
        {
            "Type": "EMAIL",
            "Value": "anika@example.com"
        },
        {
            "Type": "NOTES",
            "Value": "This is a request for Susan's account to
join Bob's organization."
        }
    ],
    "RequestedTimestamp": 1.47008383521E9,
    "ExpirationTimestamp": 1.47137983521E9
}

```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelHandshake](#)中的。

## create-account

以下代码示例演示了如何使用 create-account。

### AWS CLI

#### 创建自动属于组织的成员账户

以下示例演示如何创建组织的成员账户。为成员账户配置的名称为 Production Account，电子邮件地址为 susan@example.com。OrganizationAccountAccessRole 由于未指定 roleName 参数，Organizations 会使用默认名称自动创建 IAM 角色。此外，由于未指定 lamUserAccessToBilling 参数，允许具有足够权限的 IAM 用户或角色访问账户账单数据的设置被设置为默认值 ALLOW。Organizations 会自动向 Susan 发送一封“欢迎来到 AWS”电子邮件：

```
aws organizations create-account --email susan@example.com --account-name
"Production Account"
```

输出包括一个请求对象，以显示状态目前为 IN\_PROGRESS：

```
{
  "CreateAccountStatus": {
    "State": "IN_PROGRESS",
    "Id": "car-examplecreateaccountrequestid111"
  }
}
```

稍后，您可以通过向 describe-create-account-status 命令提供 Id 响应值作为 create-account-request-id 参数值来查询请求的当前状态。

有关更多信息，请参阅《Organizations 用户指南》中的在AWS 组织中创建帐户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAccount](#)中的。

## create-organization

以下代码示例演示了如何使用 create-organization。

## AWS CLI

### 示例 1：创建新组织

Bill 想使用账户 111111111111 中的凭证创建一个组织。以下示例显示该账户成为新组织中的主账户。由于他没有指定功能集，因此，新组织默认为在根上启用所有功能并启用服务控制策略。

```
aws organizations create-organization
```

输出包括一个组织对象，其中包含有关新组织的详细信息：

```
{
  "Organization": {
    "AvailablePolicyTypes": [
      {
        "Status": "ENABLED",
        "Type": "SERVICE_CONTROL_POLICY"
      }
    ],
    "MasterAccountId": "111111111111",
    "MasterAccountArn": "arn:aws:organizations::111111111111:account/o-exampleorgid/111111111111",
    "MasterAccountEmail": "bill@example.com",
    "FeatureSet": "ALL",
    "Id": "o-exampleorgid",
    "Arn": "arn:aws:organizations::111111111111:organization/o-exampleorgid"
  }
}
```

### 示例 2：创建仅启用整合账单功能的新组织

以下示例创建仅支持整合账单功能的组织：

```
aws organizations create-organization --feature-set CONSOLIDATED_BILLING
```

输出包括一个组织对象，其中包含有关新组织的详细信息：

```
{
  "Organization": {
```

```

        "Arn": "arn:aws:organizations::111111111111:organization/o-
exampleorgid",
        "AvailablePolicyTypes": [],
        "Id": "o-exampleorgid",
        "MasterAccountArn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/111111111111",
        "MasterAccountEmail": "bill@example.com",
        "MasterAccountId": "111111111111",
        "FeatureSet": "CONSOLIDATED_BILLING"
    }
}

```

有关更多信息，请参阅《AWS Organizations 用户指南》中的“创建组织”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateOrganization](#)中的。

## create-organizational-unit

以下代码示例演示了如何使用 create-organizational-unit。

### AWS CLI

在根 OU 或父 OU 中创建 OU

以下示例演示如何创建名为 AccountingOU 的 OU：

```
aws organizations create-organizational-unit --parent-id r-examplerootid111 --name
AccountingOU
```

输出包括一个 organizationalUnit 对象，其中包含有关新 OU 的详细信息：

```

{
  "OrganizationalUnit": {
    "Id": "ou-examplerootid111-exampleoid111",
    "Arn": "arn:aws:organizations::111111111111:ou/o-exampleorgid/ou-
examplerootid111-exampleoid111",
    "Name": "AccountingOU"
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateOrganizationalUnit](#)中的。

## create-policy

以下代码示例演示了如何使用 create-policy。

### AWS CLI

#### 示例 1：使用 JSON 策略的文本源文件创建策略

以下示例演示如何创建名为 AllowAllS3Actions 的服务控制策略 (SCP)。策略内容取自本地计算机上名为 policy.json 的文件。

```
aws organizations create-policy --content file://policy.json --name
AllowAllS3Actions, --type SERVICE_CONTROL_POLICY --description "Allows delegation
of all S3 actions"
```

输出包括一个策略对象，其中包含有关新策略的详细信息：

```
{
  "Policy": {
    "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":
\\\"Allow\\\", \"Action\":[\"s3:*\"], \"Resource\":[\"*\"]}]",
    "PolicySummary": {
      "Arn": "arn:aws:organizations::o-exampleorgid:policy/
service_control_policy/p-examplepolicyid111",
      "Description": "Allows delegation of all S3 actions",
      "Name": "AllowAllS3Actions",
      "Type": "SERVICE_CONTROL_POLICY"
    }
  }
}
```

#### 示例 2：创建以 JSON 策略作为参数的策略

以下示例演示了如何创建相同的 SCP，这次是将策略内容作为 JSON 字符串嵌入到参数中。字符串必须在双引号前使用反斜杠进行转义，以确保在参数中将其视为文本，参数本身用双引号引起来：

```
aws organizations create-policy --content "{\"Version\":\"2012-10-17\", \"Statement
\": [{\"Effect\": \"Allow\", \"Action\": [\"s3:*\"], \"Resource\": [\"*\"]}]" --name
AllowAllS3Actions --type SERVICE_CONTROL_POLICY --description "Allows delegation of
all S3 actions"
```

有关在组织中创建和使用策略的更多信息，请参阅《AWS Organizations 用户指南》中的“管理组织策略”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePolicy](#)中的。

## decline-handshake

以下代码示例演示了如何使用 decline-handshake。

### AWS CLI

拒绝从其他账户发送的握手

以下示例显示，账号 222222222222 的所有者 Susan 拒绝了加入比尔组织的邀请。该 DeclineHandshake 操作返回一个 handshake 对象，显示该状态现在为“已拒绝”：

```
aws organizations decline-handshake --handshake-id h-examplehandshakeid111
```

输出包括一个握手对象，该对象显示以下新状态：DECLINED

```
{
  "Handshake": {
    "Id": "h-examplehandshakeid111",
    "State": "DECLINED",
    "Resources": [
      {
        "Type": "ORGANIZATION",
        "Value": "o-exampleorgid",
        "Resources": [
          {
            "Type": "MASTER_EMAIL",
            "Value": "bill@example.com"
          },
          {
            "Type": "MASTER_NAME",
            "Value": "Master Account"
          }
        ]
      },
      {
        "Type": "EMAIL",
        "Value": "susan@example.com"
      }
    ]
  }
}
```



```

        },
        {
            "Type": "NOTES",
            "Value": "This is an invitation to Susan's account
to join the Bill's organization."
        }
    ],
    "Parties": [
        {
            "Type": "EMAIL",
            "Id": "susan@example.com"
        },
        {
            "Type": "ORGANIZATION",
            "Id": "o-exampleorgid"
        }
    ],
    "Action": "INVITE",
    "RequestedTimestamp": 1470684478.687,
    "ExpirationTimestamp": 1471980478.687,
    "Arn": "arn:aws:organizations::111111111111:handshake/o-
exampleorgid/invite/h-examplehandshakeid111"
    }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeclineHandshake](#)中的。

## delete-organization

以下代码示例演示了如何使用 delete-organization。

### AWS CLI

#### 删除组织

以下示例演示如何删除组织。要执行此操作，您必须是组织中主账户的管理员。该示例假设您之前已从组织中删除所有成员账户、OU 和策略：

```
aws organizations delete-organization
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteOrganization](#)中的。

## delete-organizational-unit

以下代码示例演示了如何使用 delete-organizational-unit。

### AWS CLI

#### 删除 OU

以下示例说明如何删除 OU。该示例假设您之前已从 OU 中删除所有账户和其他 OU：

```
aws organizations delete-organizational-unit --organizational-unit-id ou-examplerootid111-exampleouid111
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteOrganizationalUnit](#)中的。

## delete-policy

以下代码示例演示了如何使用 delete-policy。

### AWS CLI

#### 删除策略

以下示例演示如何删除组织的策略。该示例假设您之前已将策略与所有实体分离：

```
aws organizations delete-policy --policy-id p-examplepolicyid111
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePolicy](#)中的。

## describe-account

以下代码示例演示了如何使用 describe-account。

### AWS CLI

#### 获取有关账户的详细信息

以下示例向您展示了如何请求有关账户的详细信息：

```
aws organizations describe-account --account-id 555555555555
```

输出显示了一个账户对象，其中包含有关该账户的详细信息：

```
{
  "Account": {
    "Id": "555555555555",
    "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/555555555555",
    "Name": "Beta account",
    "Email": "anika@example.com",
    "JoinedMethod": "INVITED",
    "JoinedTimeStamp": 1481756563.134,
    "Status": "ACTIVE"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAccount](#)中的。

## describe-create-account-status

以下代码示例演示了如何使用 describe-create-account-status。

### AWS CLI

获取有关创建账户请求的最新状态

以下示例说明如何为先前在组织中创建账户的请求请求最新状态。指定的--request-id 来自最初调用 create-account 的响应。账户创建请求通过状态字段显示 Organizations 已成功完成账户的创建。

命令:

```
aws organizations describe-create-account-status --create-account-request-id car-
examplecreateaccountrequestid111
```

输出：

```
{
  "CreateAccountStatus": {
    "State": "SUCCEEDED",
    "AccountId": "555555555555",
    "AccountName": "Beta account",
    "RequestedTimestamp": 1470684478.687,
```

```

    "CompletedTimestamp": 1470684532.472,
    "Id": "car-examplecreateaccountrequestid111"
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCreateAccountStatus](#)中的。

## describe-handshake

以下代码示例演示了如何使用 describe-handshake。

### AWS CLI

获取有关握手的信息

以下示例向您展示了如何请求有关握手的详细信息。握手 ID 要么来自最初的呼叫InviteAccountToOrganization，要么来自对或的呼叫：ListHandshakesForAccountListHandshakesForOrganization

```
aws organizations describe-handshake --handshake-id h-examplehandshakeid111
```

输出包括一个握手对象，其中包含有关请求的握手的所有详细信息：

```

{
  "Handshake": {
    "Id": "h-examplehandshakeid111",
    "State": "OPEN",
    "Resources": [
      {
        "Type": "ORGANIZATION",
        "Value": "o-exampleorgid",
        "Resources": [
          {
            "Type": "MASTER_EMAIL",
            "Value": "bill@example.com"
          },
          {
            "Type": "MASTER_NAME",
            "Value": "Master Account"
          }
        ]
      }
    ]
  }
}

```

```

    },
    {
      "Type": "EMAIL",
      "Value": "anika@example.com"
    }
  ],
  "Parties": [
    {
      "Type": "ORGANIZATION",
      "Id": "o-exampleorgid"
    },
    {
      "Type": "EMAIL",
      "Id": "anika@example.com"
    }
  ],
  "Action": "INVITE",
  "RequestedTimestamp": 1470158698.046,
  "ExpirationTimestamp": 1471454698.046,
  "Arn": "arn:aws:organizations::111111111111:handshake/o-
exampleorgid/invite/h-examplehandshakeid111"
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeHandshake](#)中的。

## describe-organization

以下代码示例演示了如何使用 describe-organization。

### AWS CLI

获取有关当前组织的信息

以下示例向您展示了如何请求有关组织的详细信息：

```
aws organizations describe-organization
```

输出包括一个组织对象，其中包含有关该组织的详细信息：

```
{
  "Organization": {
```

```

    "MasterAccountArn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/111111111111",
    "MasterAccountEmail": "bill@example.com",
    "MasterAccountId": "111111111111",
    "Id": "o-exampleorgid",
    "FeatureSet": "ALL",
    "Arn": "arn:aws:organizations::111111111111:organization/o-
exampleorgid",
    "AvailablePolicyTypes": [
        {
            "Status": "ENABLED",
            "Type": "SERVICE_CONTROL_POLICY"
        }
    ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeOrganization](#)中的。

## describe-organizational-unit

以下代码示例演示了如何使用 describe-organizational-unit。

### AWS CLI

获取有关 OU 的信息

以下describe-organizational-unit示例请求有关 OU 的详细信息。

```
aws organizations describe-organizational-unit \
  --organizational-unit-id ou-examplerootid111-exampleoid111
```

输出：

```

{
  "OrganizationalUnit": {
    "Name": "Accounting Group",
    "Arn": "arn:aws:organizations::123456789012:ou/o-exampleorgid/ou-
examplerootid111-exampleoid111",
    "Id": "ou-examplerootid111-exampleoid111"
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeOrganizationalUnit](#)中的。

## describe-policy

以下代码示例演示了如何使用 describe-policy。

### AWS CLI

获取有关策略的信息

以下示例演示如何请求有关策略的信息：

```
aws organizations describe-policy --policy-id p-examplepolicyid111
```

输出包括一个策略对象，其中包含有关策略的详细信息：

```
{
  "Policy": {
    "Content": "{\n  \"Version\": \"2012-10-17\",\n  \"Statement\": [\n    {\n      \"Effect\": \"Allow\",\n      \"Action\": \"*\",\n      \"Resource\": \"*\"\n    }\n  ]\n}",
    "PolicySummary": {
      "Arn": "arn:aws:organizations::111111111111:policy/o-exampleorgid/service_control_policy/p-examplepolicyid111",
      "Type": "SERVICE_CONTROL_POLICY",
      "Id": "p-examplepolicyid111",
      "AwsManaged": false,
      "Name": "AllowAllS3Actions",
      "Description": "Enables admins to delegate S3 permissions"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePolicy](#)中的。

## detach-policy

以下代码示例演示了如何使用 detach-policy。

### AWS CLI

从根、OU 或账户分离策略

以下示例演示了如何从 OU 分离策略：

```
aws organizations detach-policy --target-id ou-examplerootid111-exampleouid111 --policy-id p-examplepolicyid111
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachPolicy](#)中的。

## disable-policy-type

以下代码示例演示了如何使用 disable-policy-type。

### AWS CLI

在根目录中禁用策略类型

以下示例说明如何在根目录中禁用服务控制策略 (SCP) 策略类型：

```
aws organizations disable-policy-type --root-id r-examplerootid111 --policy-type SERVICE_CONTROL_POLICY
```

输出显示 PolicyTypes 响应元素不再包含 SERVICE\_CONTROL\_POLICY：

```
{
  "Root": {
    "PolicyTypes": [],
    "Name": "Root",
    "Id": "r-examplerootid111",
    "Arn": "arn:aws:organizations::111111111111:root/o-exampleorgid/r-examplerootid111"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisablePolicyType](#)中的。

## enable-all-features

以下代码示例演示了如何使用 enable-all-features。

### AWS CLI

启用组织中的所有功能



此示例显示管理员要求组织中所有受邀账户批准启用组织中的所有功能。AWS Organizations 会向每个受邀成员账户注册的地址发送一封电子邮件，要求所有者通过接受发送的握手来批准对所有功能的更改。在所有受邀成员账户接受握手后，组织管理员可以完成对所有功能的更改，拥有适当权限的成员可以创建策略并将其应用于根、OU 和账户：

```
aws organizations enable-all-features
```

输出是一个握手对象，发送到所有受邀成员账户进行审批：

```
{
  "Handshake": {
    "Action": "ENABLE_ALL_FEATURES",
    "Arn": "arn:aws:organizations::111111111111:handshake/o-exampleorgid/enable_all_features/h-examplehandshakeid111",
    "ExpirationTimestamp": 1.483127868609E9,
    "Id": "h-examplehandshakeid111",
    "Parties": [
      {
        "id": "o-exampleorgid",
        "type": "ORGANIZATION"
      }
    ],
    "requestedTimestamp": 1.481831868609E9,
    "resources": [
      {
        "type": "ORGANIZATION",
        "value": "o-exampleorgid"
      }
    ],
    "state": "REQUESTED"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableAllFeatures](#)中的。

## enable-policy-type

以下代码示例演示了如何使用 enable-policy-type。

### AWS CLI

允许在根目录中使用策略类型

以下示例说明如何在根目录中启用服务控制策略 (SCP) 策略类型：

```
aws organizations enable-policy-type --root-id r-examplerootid111 --policy-type
SERVICE_CONTROL_POLICY
```

输出显示了一个带有 PolicyTypes 响应元素的根对象，该元素显示 SCP 现已启用：

```
{
  "Root": {
    "PolicyTypes": [
      {
        "Status": "ENABLED",
        "Type": "SERVICE_CONTROL_POLICY"
      }
    ],
    "Id": "r-examplerootid111",
    "Name": "Root",
    "Arn": "arn:aws:organizations::111111111111:root/o-exampleorgid/r-
examplerootid111"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnablePolicyType](#)中的。

## invite-account-to-organization

以下代码示例演示了如何使用 invite-account-to-organization。

### AWS CLI

#### 邀请账号加入组织

以下示例显示了 bill@example.com 拥有的主账户邀请 juan@example.com 拥有的账户加入组织：

```
aws organizations invite-account-to-organization --target '{"Type": "EMAIL", "Id":
"juan@example.com"}' --notes "This is a request for Juan's account to join Bill's
organization."
```

输出包括一个握手结构，该结构显示了发送到受邀账户的内容：

```
{
  "Handshake": {
```

```
    "Action": "INVITE",
    "Arn": "arn:aws:organizations::111111111111:handshake/o-
exampleorgid/invite/h-examplehandshakeid111",
    "ExpirationTimestamp": 1482952459.257,
    "Id": "h-examplehandshakeid111",
    "Parties": [
      {
        "Id": "o-exampleorgid",
        "Type": "ORGANIZATION"
      },
      {
        "Id": "juan@example.com",
        "Type": "EMAIL"
      }
    ],
    "RequestedTimestamp": 1481656459.257,
    "Resources": [
      {
        "Resources": [
          {
            "Type": "MASTER_EMAIL",
            "Value": "bill@amazon.com"
          },
          {
            "Type": "MASTER_NAME",
            "Value": "Org Master Account"
          },
          {
            "Type": "ORGANIZATION_FEATURE_SET",
            "Value": "FULL"
          }
        ],
        "Type": "ORGANIZATION",
        "Value": "o-exampleorgid"
      },
      {
        "Type": "EMAIL",
        "Value": "juan@example.com"
      }
    ],
    "State": "OPEN"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[InviteAccountToOrganization](#)中的。

## leave-organization

以下代码示例演示了如何使用 leave-organization。

### AWS CLI

以成员账户的身份离开组织

以下示例显示了一个成员账户的管理员请求离开其当前所属的组织：

```
aws organizations leave-organization
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[LeaveOrganization](#)中的。

## list-accounts-for-parent

以下代码示例演示了如何使用 list-accounts-for-parent。

### AWS CLI

检索指定父根或 OU 中所有账户的列表

以下示例说明如何请求 OU 中的账户列表：

```
aws organizations list-accounts-for-parent --parent-id ou-examplerootid111-exampleouid111
```

输出包含账户摘要对象的列表。

```
{
  "Accounts": [
    {
      "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/333333333333",
      "JoinedMethod": "INVITED",
      "JoinedTimestamp": 1481835795.536,
      "Id": "333333333333",
      "Name": "Development Account",
      "Email": "juan@example.com",
```

```

        "Status": "ACTIVE"
    },
    {
        "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/44444444444444",
        "JoinedMethod": "INVITED",
        "JoinedTimestamp": 1481835812.143,
        "Id": "44444444444444",
        "Name": "Test Account",
        "Email": "anika@example.com",
        "Status": "ACTIVE"
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAccountsForParent](#)中的。

## list-accounts

以下代码示例演示了如何使用 list-accounts。

### AWS CLI

检索组织中所有账户的列表

以下示例演示了如何请求组织中的账户列表：

```
aws organizations list-accounts
```

输出包含账户摘要对象的列表。

```

{
    "Accounts": [
        {
            "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/111111111111",
            "JoinedMethod": "INVITED",
            "JoinedTimestamp": 1481830215.45,
            "Id": "111111111111",
            "Name": "Master Account",
            "Email": "bill@example.com",
            "Status": "ACTIVE"
        }
    ]
}

```

```
    },
    {
      "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/222222222222",
      "JoinedMethod": "INVITED",
      "JoinedTimestamp": 1481835741.044,
      "Id": "222222222222",
      "Name": "Production Account",
      "Email": "alice@example.com",
      "Status": "ACTIVE"
    },
    {
      "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/333333333333",
      "JoinedMethod": "INVITED",
      "JoinedTimestamp": 1481835795.536,
      "Id": "333333333333",
      "Name": "Development Account",
      "Email": "juan@example.com",
      "Status": "ACTIVE"
    },
    {
      "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/444444444444",
      "JoinedMethod": "INVITED",
      "JoinedTimestamp": 1481835812.143,
      "Id": "444444444444",
      "Name": "Test Account",
      "Email": "anika@example.com",
      "Status": "ACTIVE"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAccounts](#)中的。

## list-children

以下代码示例演示了如何使用 list-children。

### AWS CLI

检索父 OU 或 root 的子账户和 OU

以下示例说明如何列出包含该帐户 444444444444 的根或 OU：

```
aws organizations list-children --child-type ORGANIZATIONAL_UNIT --parent-id ou-examplerootid111-exampleouid111
```

输出显示父OU包含的两个子OU：

```
{
  "Children": [
    {
      "Id": "ou-examplerootid111-exampleouid111",
      "Type": "ORGANIZATIONAL_UNIT"
    },
    {
      "Id": "ou-examplerootid111-exampleouid222",
      "Type": "ORGANIZATIONAL_UNIT"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListChildren](#)中的。

## list-create-account-status

以下代码示例演示了如何使用 list-create-account-status。

### AWS CLI

示例 1：检索在当前组织中提出的账户创建请求列表

以下示例说明如何为已成功完成的组织请求账户创建请求列表：

```
aws organizations list-create-account-status --states SUCCEEDED
```

输出包括一个对象数组，其中包含有关每个请求的信息。

```
{
  "CreateAccountStatuses": [
    {
      "AccountId": "444444444444",
      "AccountName": "Developer Test Account",

```

```
        "CompletedTimeStamp": 1481835812.143,  
        "Id": "car-examplecreateaccountrequestid111",  
        "RequestedTimeStamp": 1481829432.531,  
        "State": "SUCCEEDED"  
    }  
]  
}
```

示例 2：检索当前组织中正在进行的账户创建请求的列表

以下示例获取组织正在处理的账户创建请求列表：

```
aws organizations list-create-account-status --states IN_PROGRESS
```

输出包括一个对象数组，其中包含有关每个请求的信息。

```
{  
  "CreateAccountStatuses": [  
    {  
      "State": "IN_PROGRESS",  
      "Id": "car-examplecreateaccountrequestid111",  
      "RequestedTimeStamp": 1481829432.531,  
      "AccountName": "Production Account"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCreateAccountStatus](#)中的。

## list-handshakes-for-account

以下代码示例演示了如何使用 list-handshakes-for-account。

### AWS CLI

检索发送到账户的握手清单

以下示例说明如何获取与用于调用操作的凭据账户关联的所有握手列表：

```
aws organizations list-handshakes-for-account
```



输出包括握手结构列表，其中包含有关每次握手的信息，包括其当前状态：

```
{
  "Handshake": {
    "Action": "INVITE",
    "Arn": "arn:aws:organizations::111111111111:handshake/o-
exampleorgid/invite/h-examplehandshakeid111",
    "ExpirationTimestamp": 1482952459.257,
    "Id": "h-examplehandshakeid111",
    "Parties": [
      {
        "Id": "o-exampleorgid",
        "Type": "ORGANIZATION"
      },
      {
        "Id": "juan@example.com",
        "Type": "EMAIL"
      }
    ],
    "RequestedTimestamp": 1481656459.257,
    "Resources": [
      {
        "Resources": [
          {
            "Type": "MASTER_EMAIL",
            "Value": "bill@amazon.com"
          },
          {
            "Type": "MASTER_NAME",
            "Value": "Org Master Account"
          },
          {
            "Type": "ORGANIZATION_FEATURE_SET",
            "Value": "FULL"
          }
        ],
        "Type": "ORGANIZATION",
        "Value": "o-exampleorgid"
      },
      {
        "Type": "EMAIL",
        "Value": "juan@example.com"
      }
    ],
  },
}
```

```
        "State": "OPEN"
    }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListHandshakesForAccount](#)中的。

## list-handshakes-for-organization

以下代码示例演示了如何使用 `list-handshakes-for-organization`。

### AWS CLI

检索与组织相关的握手列表

以下示例说明如何获取与当前组织关联的握手列表：

```
aws organizations list-handshakes-for-organization
```

输出显示两次握手。第一个是对Juan账户的邀请，显示的状态为OPEN。第二个是对 Anika 账户的邀请，显示的状态为“已接受”：

```
{
  "Handshakes": [
    {
      "Action": "INVITE",
      "Arn": "arn:aws:organizations::111111111111:handshake/o-exampleorgid/invite/h-examplehandshakeid111",
      "ExpirationTimestamp": 1482952459.257,
      "Id": "h-examplehandshakeid111",
      "Parties": [
        {
          "Id": "o-exampleorgid",
          "Type": "ORGANIZATION"
        },
        {
          "Id": "juan@example.com",
          "Type": "EMAIL"
        }
      ],
      "RequestedTimestamp": 1481656459.257,
      "Resources": [
        {
```

```

        "Resources": [
            {
                "Type": "MASTER_EMAIL",
                "Value": "bill@amazon.com"
            },
            {
                "Type": "MASTER_NAME",
                "Value": "Org Master"
            },
            {
                "Type": "ORGANIZATION_FEATURE_SET",
                "Value": "FULL"
            }
        ],
        "Type": "ORGANIZATION",
        "Value": "o-exampleorgid"
    },
    {
        "Type": "EMAIL",
        "Value": "juan@example.com"
    },
    {
        "Type": "NOTES",
        "Value": "This is an invitation to Juan's
account to join Bill's organization."
    }
],
"State": "OPEN"
},
{
    "Action": "INVITE",
    "State": "ACCEPTED",
    "Arn": "arn:aws:organizations::111111111111:handshake/o-
exampleorgid/invite/h-examplehandshakeid111",
    "ExpirationTimestamp": 1.471797437427E9,
    "Id": "h-examplehandshakeid222",
    "Parties": [
        {
            "Id": "o-exampleorgid",
            "Type": "ORGANIZATION"
        }
    ]
}

```

```

        "Id": "anika@example.com",
        "Type": "EMAIL"
      }
    ],
    "RequestedTimestamp": 1.469205437427E9,
    "Resources": [
      {
        "Resources": [
          {
            "Type": "MASTER_EMAIL",
            "Value": "bill@example.com"
          },
          {
            "Type": "MASTER_NAME",
            "Value": "Master Account"
          }
        ],
        "Type": "ORGANIZATION",
        "Value": "o-exampleorgid"
      },
      {
        "Type": "EMAIL",
        "Value": "anika@example.com"
      },
      {
        "Type": "NOTES",
        "Value": "This is an invitation to Anika's
account to join Bill's organization."
      }
    ]
  }
]
}

```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListHandshakesForOrganization](#) 中的。

## list-organizational-units-for-parent

以下代码示例演示了如何使用 list-organizational-units-for-parent。

### AWS CLI

检索父 OU 或根中的 OU 列表

以下示例说明如何获取指定根目录下的 OU 列表：

```
aws organizations list-organizational-units-for-parent --parent-id r-
examplerootid111
```

输出显示指定的根包含两个 OU，并显示了每个 OU 的详细信息：

```
{
  "OrganizationalUnits": [
    {
      "Name": "AccountingDepartment",
      "Arn": "arn:aws:organizations::o-exampleorgid:ou/r-
examplerootid111/ou-examplerootid111-exampleoid111"
    },
    {
      "Name": "ProductionDepartment",
      "Arn": "arn:aws:organizations::o-exampleorgid:ou/r-
examplerootid111/ou-examplerootid111-exampleoid222"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListOrganizationalUnitsForParent](#)中的。

## list-parents

以下代码示例演示了如何使用 list-parents。

### AWS CLI

列出账户或子 OU 的父 OU 或根

以下示例说明如何列出包含该账户 444444444444 的根组织单位或父 OU：

```
aws organizations list-parents --child-id 444444444444
```

输出显示指定账户位于具有指定 ID 的 OU 中：

```
{
  "Parents": [
    {
```

```

        "Id": "ou-examplerootid111-exampleoid111",
        "Type": "ORGANIZATIONAL_UNIT"
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListParents](#)中的。

## list-policies-for-target

以下代码示例演示了如何使用 `list-policies-for-target`。

### AWS CLI

检索直接关联到账户的 SCP 列表

以下示例说明如何获取直接关联到账户的所有服务控制策略 (SCP) 的列表，这些策略由 `Filter` 参数指定：

```
aws organizations list-policies-for-target --filter SERVICE_CONTROL_POLICY --target-id 44444444444444
```

输出包括策略结构列表以及有关策略的摘要信息。该列表不包括适用于该账户的策略，因为这些策略是从账户在 OU 层次结构中的位置继承的：

```

{
    "Policies": [
        {
            "Type": "SERVICE_CONTROL_POLICY",
            "Name": "AllowAllEC2Actions",
            "AwsManaged": false,
            "Id": "p-examplepolicyid222",
            "Arn": "arn:aws:organizations::o-exampleorgid:policy/service_control_policy/p-examplepolicyid222",
            "Description": "Enables account admins to delegate permissions for any EC2 actions to users and roles in their accounts."
        }
    ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPoliciesForTarget](#)中的。

## list-policies

以下代码示例演示了如何使用 `list-policies`。

### AWS CLI

检索特定类型组织中所有策略的列表

以下示例演示了如何获取由筛选器参数指定的 SCP 列表：

```
aws organizations list-policies --filter SERVICE_CONTROL_POLICY
```

输出包括含摘要信息的策略列表：

```
{
  "Policies": [
    {
      "Type": "SERVICE_CONTROL_POLICY",
      "Name": "AllowAllS3Actions",
      "AwsManaged": false,
      "Id": "p-examplepolicyid111",
      "Arn": "arn:aws:organizations::111111111111:policy/
service_control_policy/p-examplepolicyid111",
      "Description": "Enables account admins to delegate
permissions for any S3 actions to users and roles in their accounts."
    },
    {
      "Type": "SERVICE_CONTROL_POLICY",
      "Name": "AllowAllEC2Actions",
      "AwsManaged": false,
      "Id": "p-examplepolicyid222",
      "Arn": "arn:aws:organizations::111111111111:policy/
service_control_policy/p-examplepolicyid222",
      "Description": "Enables account admins to delegate
permissions for any EC2 actions to users and roles in their accounts."
    },
    {
      "AwsManaged": true,
      "Description": "Allows access to every operation",
      "Type": "SERVICE_CONTROL_POLICY",
      "Id": "p-FullAWSAccess",
      "Arn": "arn:aws:organizations::aws:policy/
service_control_policy/p-FullAWSAccess",
      "Name": "FullAWSAccess"
    }
  ]
}
```

```

    ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPolicies](#)中的。

## list-roots

以下代码示例演示了如何使用 `list-roots`。

### AWS CLI

检索组织中的根列表

此示例说明如何获取组织的根列表：

```
aws organizations list-roots
```

输出包括带有摘要信息的根结构列表：

```

{
  "Roots": [
    {
      "Name": "Root",
      "Arn": "arn:aws:organizations::111111111111:root/o-
exampleorgid/r-examplerootid111",
      "Id": "r-examplerootid111",
      "PolicyTypes": [
        {
          "Status": "ENABLED",
          "Type": "SERVICE_CONTROL_POLICY"
        }
      ]
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRoots](#)中的。

## list-targets-for-policy

以下代码示例演示了如何使用 `list-targets-for-policy`。



## AWS CLI

检索策略所关联的根、OU 和账户的列表

以下示例说明如何获取指定策略所关联的根、OU 和账户的列表：

```
aws organizations list-targets-for-policy --policy-id p-FullAWSAccess
```

输出包括附件对象列表，其中包含有关策略所关联的根、OU 和账户的摘要信息：

```
{
  "Targets": [
    {
      "Arn": "arn:aws:organizations::111111111111:root/o-
exampleorgid/r-examplerootid111",
      "Name": "Root",
      "TargetId": "r-examplerootid111",
      "Type": "ROOT"
    },
    {
      "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/333333333333;",
      "Name": "Developer Test Account",
      "TargetId": "333333333333",
      "Type": "ACCOUNT"
    },
    {
      "Arn": "arn:aws:organizations::111111111111:ou/o-
exampleorgid/ou-examplerootid111-exampleouid111",
      "Name": "Accounting",
      "TargetId": "ou-examplerootid111-exampleouid111",
      "Type": "ORGANIZATIONAL_UNIT"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTargetsForPolicy](#)中的。

### move-account

以下代码示例演示了如何使用 move-account。

## AWS CLI

在 root 或 OU 之间移动帐户

以下示例说明如何将组织中的主帐户从根帐户移至 OU：

```
aws organizations move-account --account-id 333333333333 --source-parent-id r-exemplerootid111 --destination-parent-id ou-exemplerootid111-exampleouid111
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[MoveAccount](#)中的。

## remove-account-from-organization

以下代码示例演示了如何使用 remove-account-from-organization。

## AWS CLI

将帐户作为主帐户从组织中移除

以下示例向您展示了如何从组织中移除帐户：

```
aws organizations remove-account-from-organization --account-id 333333333333
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveAccountFromOrganization](#)中的。

## update-organizational-unit

以下代码示例演示了如何使用 update-organizational-unit。

## AWS CLI

重命名 OU

此示例向您展示如何重命名 OU：在此示例中，组织单位重命名为“AccountingOU”：

```
aws organizations update-organizational-unit --organizational-unit-id ou-exemplerootid111-exampleouid111 --name AccountingOU
```

输出显示了新名称：

```
{
```

```

    "OrganizationalUnit": {
      "Id": "ou-examplerootid111-exampleoid111"
      "Name": "AccountingOU",
      "Arn": "arn:aws:organizations::111111111111:ou/o-exampleorgid/ou-
    exemplerooid111-exampleoid111"
    }
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateOrganizationalUnit](#)中的。

## update-policy

以下代码示例演示了如何使用 update-policy。

### AWS CLI

#### 示例 1：重命名策略

以下update-policy示例重命名了策略并对其进行了新的描述。

```

aws organizations update-policy \
  --policy-id p-examplepolicyid111 \
  --name Renamed-Policy \
  --description "This description replaces the original."

```

输出显示了新的名称和描述。

```

{
  "Policy": {
    "Content": "{\n  \"Version\": \"2012-10-17\",\n  \"Statement\": {\n
    \"Effect\": \"Allow\",\n    \"Action\": \"ec2:*\",\n    \"Resource\": \"*\"\n
  }\n}\n",
    "PolicySummary": {
      "Id": "p-examplepolicyid111",
      "AwsManaged": false,
      "Arn": "arn:aws:organizations::111111111111:policy/o-exampleorgid/
service_control_policy/p-examplepolicyid111",
      "Description": "This description replaces the original.",
      "Name": "Renamed-Policy",
      "Type": "SERVICE_CONTROL_POLICY"
    }
  }
}

```

```
}
```

## 示例 2：替换政策的 JSON 文本内容

以下示例向您展示了如何将上一个示例中的 SCP 的 JSON 文本替换为允许 S3 而不是 EC2 的新 JSON 策略文本字符串：

```
aws organizations update-policy \  
  --policy-id p-examplepolicyid111 \  
  --content "{\"Version\":\"2012-10-17\",\"Statement\":{\"Effect\":\"Allow\",\  
  \"Action\":\"s3:*\",\"Resource\":\"*\"}}"
```

输出显示了新内容：

```
{  
  "Policy": {  
    "Content": "{ \"Version\": \"2012-10-17\", \"Statement\": { \"Effect\":  
  \"Allow\", \"Action\": \"s3:*\", \"Resource\": \"*\" } }",  
    "PolicySummary": {  
      "Arn": "arn:aws:organizations::111111111111:policy/o-exampleorgid/  
service_control_policy/p-examplepolicyid111",  
      "AwsManaged": false;  
      "Description": "This description replaces the original.",  
      "Id": "p-examplepolicyid111",  
      "Name": "Renamed-Policy",  
      "Type": "SERVICE_CONTROL_POLICY"  
    }  
  }  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdatePolicy](#) 中的。

## AWS Outposts 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS Outposts。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### get-outpost-instance-types

以下代码示例演示了如何使用 `get-outpost-instance-types`。

#### AWS CLI

要获取前哨基地上的实例类型

以下 `get-outpost-instance-types` 示例获取了指定 Outpost 的实例类型。

```
aws outposts get-outpost-instance-types \  
  --outpost-id op-0ab23c4567EXAMPLE
```

输出：

```
{  
  "InstanceTypes": [  
    {  
      "InstanceType": "c5d.large"  
    },  
    {  
      "InstanceType": "i3en.24xlarge"  
    },  
    {  
      "InstanceType": "m5d.large"  
    },  
    {  
      "InstanceType": "r5d.large"  
    }  
  ],  
  "OutpostId": "op-0ab23c4567EXAMPLE",  
  "OutpostArn": "arn:aws:outposts:us-west-2:123456789012:outpost/  
op-0ab23c4567EXAMPLE"  
}
```

欲了解更多信息，请参阅 [Outposts 用户指南中的在你的 Outpost 上启动实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetOutpostInstanceTypes](#) 中的。

## get-outpost

以下代码示例演示了如何使用 get-outpost。

### AWS CLI

要获取 Outpost 的详细信息

以下 get-outpost 示例显示了指定 Outpost 的详细信息。

```
aws outposts get-outpost \
  --outpost-id op-0ab23c4567EXAMPLE
```

输出：

```
{
  "Outpost": {
    "OutpostId": "op-0ab23c4567EXAMPLE",
    "OwnerId": "123456789012",
    "OutpostArn": "arn:aws:outposts:us-west-2:123456789012:outpost/
op-0ab23c4567EXAMPLE",
    "SiteId": "os-0ab12c3456EXAMPLE",
    "Name": "EXAMPLE",
    "LifecycleStatus": "ACTIVE",
    "AvailabilityZone": "us-west-2a",
    "AvailabilityZoneId": "usw2-az1",
    "Tags": {}
  }
}
```

有关更多信息，请参阅 [Outposts 用户指南中的使用 Outposts AWS](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetOutpost](#) 中的。

## list-outposts

以下代码示例演示了如何使用 list-outposts。

## AWS CLI

### 列出 Outposts

以下 `list-outposts` 示例列出了您 AWS 账户中的 Outposts。

```
aws outposts list-outposts
```

输出：

```
{
  "Outposts": [
    {
      "OutpostId": "op-0ab23c4567EXAMPLE",
      "OwnerId": "123456789012",
      "OutpostArn": "arn:aws:outposts:us-west-2:123456789012:outpost/
op-0ab23c4567EXAMPLE",
      "SiteId": "os-0ab12c3456EXAMPLE",
      "Name": "EXAMPLE",
      "Description": "example",
      "LifecycleStatus": "ACTIVE",
      "AvailabilityZone": "us-west-2a",
      "AvailabilityZoneId": "usw2-az1",
      "Tags": {
        "Name": "EXAMPLE"
      }
    },
    {
      "OutpostId": "op-4fe3dc21baEXAMPLE",
      "OwnerId": "123456789012",
      "OutpostArn": "arn:aws:outposts:us-west-2:123456789012:outpost/
op-4fe3dc21baEXAMPLE",
      "SiteId": "os-0ab12c3456EXAMPLE",
      "Name": "EXAMPLE2",
      "LifecycleStatus": "ACTIVE",
      "AvailabilityZone": "us-west-2a",
      "AvailabilityZoneId": "usw2-az1",
      "Tags": {}
    }
  ]
}
```

有关更多信息，请参阅 [Outposts 用户指南中的使用 Outposts AWS](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListOutposts](#)中的。

## list-sites

以下代码示例演示了如何使用 `list-sites`。

### AWS CLI

列出网站

以下`list-sites`示例列出了您 AWS 账户中可用的 Outpost 站点。

```
aws outposts list-sites
```

输出：

```
{
  "Sites": [
    {
      "SiteId": "os-0ab12c3456EXAMPLE",
      "AccountId": "123456789012",
      "Name": "EXAMPLE",
      "Description": "example",
      "Tags": {}
    }
  ]
}
```

有关更多信息，请参阅 [Outposts 用户指南中的使用 Outposts AWS](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSites](#)中的。

## AWS Payment Cryptography 使用示例 AWS CLI

以下代码示例向您展示了如何使用`with`来执行操作和实现常见场景 AWS Payment Cryptography。

### AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。



每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## create-alias

以下代码示例演示了如何使用 create-alias。

AWS CLI

为密钥创建别名

以下create-alias示例为密钥创建别名。

```
aws payment-cryptography create-alias \  
  --alias-name alias/sampleAlias1 \  
  --key-arn arn:aws:payment-cryptography:us-east-2:123456789012:key/  
  kwapwa6qaifllw2h
```

输出：

```
{  
  "Alias": {  
    "AliasName": "alias/sampleAlias1",  
    "KeyArn": "arn:aws:payment-cryptography:us-west-2:123456789012:key/  
    kwapwa6qaifllw2h"  
  }  
}
```

有关更多信息，请参阅《AWS 支付密码学用户指南》中的[关于别名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAlias](#)中的。

## create-key

以下代码示例演示了如何使用 create-key。

## AWS CLI

### 创建密钥

以下`create-key`示例生成一个 2KEY TDES 密钥，您可以使用该密钥生成和验证 CVV/CVV2 值。

```
aws payment-cryptography create-key \  
  --exportable \  
  --key-attributes KeyAlgorithm=TDES_2KEY,  
  KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,  
  KeyModesOfUse={Generate=true,Verify=true}
```

输出：

```
{  
  "Key": {  
    "CreateTimestamp": "1686800690",  
    "Enabled": true,  
    "Exportable": true,  
    "KeyArn": "arn:aws:payment-cryptography:us-west-2:123456789012:key/  
kwapwa6qaifllw2h",  
    "KeyAttributes": {  
      "KeyAlgorithm": "TDES_2KEY",  
      "KeyClass": "SYMMETRIC_KEY",  
      "KeyModesOfUse": {  
        "Decrypt": false,  
        "DeriveKey": false,  
        "Encrypt": false,  
        "Generate": true,  
        "NoRestrictions": false,  
        "Sign": false,  
        "Unwrap": false,  
        "Verify": true,  
        "Wrap": false  
      },  
      "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY"  
    },  
    "KeyCheckValue": "F2E50F",  
    "KeyCheckValueAlgorithm": "ANSI_X9_24",  
    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",  
    "KeyState": "CREATE_COMPLETE",  
    "UsageStartTimestamp": "1686800690"  
  }  
}
```

```
}  
}
```

有关更多信息，请参阅《AWS 支付加密用户指南》中的[生成密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateKey](#)中的。

## delete-alias

以下代码示例演示了如何使用 delete-alias。

### AWS CLI

#### 删除别名

以下delete-alias示例删除了别名。它不会影响密钥。

```
aws payment-cryptography delete-alias \  
  --alias-name alias/sampleAlias1
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 支付密码学用户指南》中的[关于别名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAlias](#)中的。

## delete-key

以下代码示例演示了如何使用 delete-key。

### AWS CLI

#### 删除密钥

以下delete-key示例计划在 7 天后删除密钥，这是默认的等待期。

```
aws payment-cryptography delete-key \  
  --key-identifier arn:aws:payment-cryptography:us-west-2:123456789012:key/  
  kwapwa6qaif1lw2h
```

输出：

```
{
  "Key": {
    "CreateTimestamp": "1686801198",
    "DeletePendingTimestamp": "1687405998",
    "Enabled": true,
    "Exportable": true,
    "KeyArn": "arn:aws:payment-cryptography:us-west-2:123456789012:key/kwapwa6qaifllw2h",
    "KeyAttributes": {
      "KeyAlgorithm": "TDES_2KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyModesOfUse": {
        "Decrypt": false,
        "DeriveKey": false,
        "Encrypt": false,
        "Generate": true,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": false,
        "Verify": true,
        "Wrap": false
      },
      "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY"
    },
    "KeyCheckValue": "F2E50F",
    "KeyCheckValueAlgorithm": "ANSI_X9_24",
    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
    "KeyState": "DELETE_PENDING",
    "UsageStartTimestamp": "1686801190"
  }
}
```

有关更多信息，请参阅《AWS 支付加密用户指南》中的[删除密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteKey](#) 中的。

## export-key

以下代码示例演示了如何使用 export-key。

### AWS CLI

#### 导出密钥

以下 `export-key` 示例导出密钥。

```
aws payment-cryptography export-key \  
  --export-key-identifier arn:aws:payment-cryptography:us-west-2:123456789012:key/  
lco3w6agsk7zgu2l \  
  --key-material '{"Tr34KeyBlock": { \  
    "CertificateAuthorityPublicKeyIdentifier": "arn:aws:payment-cryptography:us-  
west-2:123456789012:key/ftobshq7pvioc5fx", \  
    "ExportToken": "export-token-cu4lg26ofcziixny", \  
    "KeyBlockFormat": "X9_TR34_2012", \  
    "WrappingKeyCertificate": file://wrapping-key-certificate.pem } }'
```

`wrapping-key-certificate.pem` 的内容：

```
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV2VENDQXFXZ0F3SUJBZ01SQU1ZZS8xMXFUK2svVz1RUDJQOE1V
```

输出：

```
{  
  "WrappedKey": {  
    "KeyMaterial":  
    "308205A106092A864886F70D010702A08205923082058E020101310D300B06096086480165030402013082031F  
    "WrappedKeyMaterialFormat": "TR34_KEY_BLOCK"  
  }  
}
```

有关更多信息，请参阅《AWS 付款加密用户指南》中的[导出密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ExportKey](#) 中的。

## get-alias

以下代码示例演示了如何使用 `get-alias`。

### AWS CLI

要获取别名

以下 `get-alias` 示例返回与别名关联的密钥的 ARN。

```
aws payment-cryptography get-alias \  
  --alias-name alias/sampleAlias1
```

输出：

```
{
  "Alias": {
    "AliasName": "alias/sampleAlias1",
    "KeyArn": "arn:aws:payment-cryptography:us-west-2:123456789012:key/
kwapwa6qaifllw2h"
  }
}
```

有关更多信息，请参阅《AWS 支付密码学用户指南》中的[关于别名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAlias](#)中的。

## get-key

以下代码示例演示了如何使用 get-key。

### AWS CLI

获取密钥的元数据

以下get-key示例返回与别名关联的密钥的元数据。此操作不返回加密材料。

```
aws payment-cryptography get-key \
  --key-identifier alias/sampleAlias1
```

输出：

```
{
  "Key": {
    "CreateTimestamp": "1686800690",
    "DeletePendingTimestamp": "1687405998",
    "Enabled": true,
    "Exportable": true,
    "KeyArn": "arn:aws:payment-cryptography:us-west-2:123456789012:key/
kwapwa6qaifllw2h",
    "KeyAttributes": {
      "KeyAlgorithm": "TDES_2KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyModesOfUse": {
        "Decrypt": false,
        "DeriveKey": false,

```

```

        "Encrypt": false,
        "Generate": true,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": false,
        "Verify": true,
        "Wrap": false
    },
    "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY"
},
"KeyCheckValue": "F2E50F",
"KeyCheckValueAlgorithm": "ANSI_X9_24",
"KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
"KeyState": "DELETE_PENDING",
"UsageStartTimestamp": "1686801190"
}
}

```

有关更多信息，请参阅《AWS 支付密码学用户指南》中的[获取密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetKey](#) 中的。

## get-parameters-for-export

以下代码示例演示了如何使用 get-parameters-for-export。

### AWS CLI

#### 初始化导出过程

以下 get-parameters-for-export 示例生成密钥对，对密钥进行签名，然后返回证书和证书根。

```

aws payment-cryptography get-parameters-for-export \
  --signing-key-algorithm RSA_2048 \
  --key-material-type TR34_KEY_BLOCK

```

#### 输出：

```

{
  "ExportToken": "export-token-ep5cwyzone7oya53",
  "ParametersValidUntilTimestamp": "1687415640",
  "SigningKeyAlgorithm": "RSA_2048",

```

```
"SigningKeyCertificate":
```

```
"MIICiTCCAfICcQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBA5TC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWVxHmAd
BgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC0lBTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWVxHmAdBgkqhkiG9w0BCQEWEG5vb251QGft
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZncvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVvxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFbjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=",
```

```
"SigningKeyCertificateChain":
```

```
"NIICiTCCAfICcQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBA5TC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWVxHmAd
BgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC0lBTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWVxHmAdBgkqhkiG9w0BCQEWEG5vb251QGft
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZncvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVvxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFbjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE="
```

```
}
```

有关更多信息，请参阅《AWS 付款加密用户指南》中的[导出密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetParametersForExport](#)中的。

## get-parameters-for-import

以下代码示例演示了如何使用 `get-parameters-for-import`。

### AWS CLI

#### 初始化导入过程



以下`get-parameters-for-import`示例生成密钥对，对密钥进行签名，然后返回证书和证书根。

```
aws payment-cryptography get-parameters-for-import \
  --key-material-type TR34_KEY_BLOCK \
  --wrapping-key-algorithm RSA_2048
```

输出：

```
{
  "ImportToken": "import-token-qgmafpaa7nt2kfbb",
  "ParametersValidUntilTimestamp": "1687415640",
  "WrappingKeyAlgorithm": "RSA_2048",
  "WrappingKeyCertificate":
  "MIICiTCCAfICCD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMakGA1UEBhMC
  VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
  b24xFDASBgNVBA5TC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWVxHmAd
  BgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
  MTIwNDI0MjA0NTIxWjCBiDELMakGA1UEBhMCMVVMxCzAJBgNVBAGTAldBMRAwDgYD
  VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC0lBTSBDb25z
  b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWVxHmAdBgkqhkiG9w0BCQEWEG5vb251QGft
  YXpvbi5jb20wZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
  21uUSfwfEvYswtC2XADZ4nB+BLygVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
  rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
  Ibb30hjZnczvQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
  nUHVvXyUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
  FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJ10ZxBHjJnyp3780D8uTs7fLvJx79LjStB
  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=",
  "WrappingKeyCertificateChain":
  "MIICiTCCAfICCD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMakGA1UEBhMC
  VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
  b24xFDASBgNVBA5TC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWVxHmAd
  BgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
  MTIwNDI0MjA0NTIxWjCBiDELMakGA1UEBhMCMVVMxCzAJBgNVBAGTAldBMRAwDgYD
  VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC0lBTSBDb25z
  b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWVxHmAdBgkqhkiG9w0BCQEWEG5vb251QGft
  YXpvbi5jb20wZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
  21uUSfwfEvYswtC2XADZ4nB+BLygVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
  rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
  Ibb30hjZnczvQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
  nUHVvXyUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
  FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJ10ZxBHjJnyp3780D8uTs7fLvJx79LjStB
  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE="
```

```
}

```

有关更多信息，请参阅《AWS 支付加密用户指南》中的[导入密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetParametersForImport](#) 中的。

## get-public-key-certificate

以下代码示例演示了如何使用 `get-public-key-certificate`。

### AWS CLI

#### 返回公钥

以下 `get-public-key-certificate` 示例返回 key pair 的公钥部分。

```
aws payment-cryptography get-public-key-certificate \
  --key-identifier arn:aws:payment-cryptography:us-east-2:123456789012:key/
  kwapwa6qaif1lw2h
```

#### 输出：

```
{
  "KeyCertificate":
  "MIICiTCCAfICCD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMakGA1UEBhMC
  VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
  b24xZDASBgNVBAstC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWx1eHAd
  BgkqhkiG9w0BCQEWEG5vb25lQGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
  MTIwNDI0MjA0NTIxWjCBiDELMakGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
  VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xZDASBgNVBAstC01BTSBDb25z
  b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWx1eHAdBgkqhkiG9w0BCQEWEG5vb25lQGft
  YXpvbi5jb20wZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMak0dn+a4GmWIWJ
  21uUSfwfEvySWtC2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
  rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
  Ibb30hjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
  nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
  FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjStB
  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=",
  "KeyCertificateChain":
  "MIICiTCCAfICCD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMakGA1UEBhMC
  VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
  b24xZDASBgNVBAstC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWx1eHAd
  BgkqhkiG9w0BCQEWEG5vb25lQGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
```

```

MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCMVVMxCzAJBgNVBAGTA1dBMRawDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAwTC01BTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWx1ZAdBgkqhkiG9w0BCQEWEG5vb251QGft
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySwTC2XADZ4nB+BLyYgVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUHvVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE="
}

```

有关更多信息，请参阅《AWS 支付密码学用户指南》中的“[获取与密钥 pair 关联的公钥/证书](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetPublicKeyCertificate](#) 中的。

## import-key

以下代码示例演示了如何使用 import-key。

### AWS CLI

#### 导入 TR-34 密钥

以下 import-key 示例导入一个 TR-34 密钥。

```

aws payment-cryptography import-key \
  --key-material='{ "Tr34KeyBlock": {" \
    CertificateAuthorityPublicKeyIdentifier": "arn:aws:payment-
cryptography:us-west-2:123456789012:key/rmm5wn2q564njinjm", \
    "ImportToken": "import-token-5ott6ho5nts7bbc", \
    "KeyBlockFormat": "X9_TR34_2012", \
    "SigningKeyCertificate": file://signing-key-certificate.pem, \
    "WrappedKeyBlock": file://wrapped-key-block.pem } }'

```

signing-key-certificate.pem 的内容：

```
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV2RENDQXFTZ0F3SUJBZ01RYWVCK25IbE1WZU1PR1ZiNjU1Q2Jz
```

wrapped-key-block.pem 的内容：

```
3082059806092A864886F70D010702A082058930820585020101310D300B06096086480165030402013082031606
```

输出：

```
{
  "Key": {
    "CreateTimestamp": "2023-06-09T16:56:27.621000-07:00",
    "Enabled": true,
    "KeyArn": "arn:aws:payment-cryptography:us-west-2:123456789012:key/
bzmvgyxgdg3sktwxd",
    "KeyAttributes": {
      "KeyAlgorithm": "TDES_2KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyModesOfUse": {
        "Decrypt": false,
        "DeriveKey": false,
        "Encrypt": false,
        "Generate": true,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": false,
        "Verify": true,
        "Wrap": false
      },
      "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY"
    },
    "KeyCheckValue": "D9B20E",
    "KeyCheckValueAlgorithm": "ANSI_X9_24",
    "KeyOrigin": "EXTERNAL",
    "KeyState": "CREATE_COMPLETE",
    "UsageStartTimestamp": "2023-06-09T16:56:27.621000-07:00"
  }
}
```

有关更多信息，请参阅《AWS 支付加密用户指南》中的[导入密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ImportKey](#) 中的。

## list-aliases

以下代码示例演示了如何使用 list-aliases。

### AWS CLI

#### 获取别名列表

以下`list-aliases`示例显示了您在该地区账户中的所有别名。

```
aws payment-cryptography list-aliases
```

输出：

```
{
  "Aliases": [
    {
      "AliasName": "alias/sampleAlias1",
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:123456789012:key/kwapwa6qaiifllw2h"
    },
    {
      "AliasName": "alias/sampleAlias2",
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:123456789012:key/kwapwa6qaiifllw2h"
    }
  ]
}
```

有关更多信息，请参阅《AWS 支付密码学用户指南》中的[关于别名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAliases](#)中的。

## list-keys

以下代码示例演示了如何使用 `list-keys`。

AWS CLI

要获取密钥清单

以下`list-keys`示例显示了您在该区域的账户中的所有密钥。

```
aws payment-cryptography list-keys
```

输出：

```
{
  "Keys": [
    {
```

```

    "CreateTimestamp": "1666506840",
    "Enabled": false,
    "Exportable": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:123456789012:key/
kwapwa6qaiifllw2h",
    "KeyAttributes": {
      "KeyAlgorithm": "TDES_3KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyModesOfUse": {
        "Decrypt": true,
        "DeriveKey": false,
        "Encrypt": true,
        "Generate": false,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": true,
        "Verify": false,
        "Wrap": true
      },
      "KeyUsage": "TR31_P1_PIN_GENERATION_KEY"
    },
    "KeyCheckValue": "369D",
    "KeyCheckValueAlgorithm": "ANSI_X9_24",
    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
    "KeyState": "CREATE_COMPLETE",
    "UsageStopTimestamp": "1666938840"
  }
]
}

```

有关更多信息，请参阅《AWS 支付加密用户指南》中的[列出密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListKeys](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

获取密钥的标签列表

以下 `list-tags-for-resource` 示例获取密钥的标签。

```
aws payment-cryptography list-tags-for-resource \  
  --resource-arn arn:aws:payment-cryptography:us-east-2:123456789012:key/  
  kwapwa6qaiifllw2h
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "BIN",  
      "Value": "20151120"  
    },  
    {  
      "Key": "Project",  
      "Value": "Production"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS 支付加密用户指南》中的[使用 API 操作管理密钥标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## restore-key

以下代码示例演示了如何使用 restore-key。

### AWS CLI

恢复计划删除的密钥

以下restore-key示例取消了对密钥的删除。

```
aws payment-cryptography restore-key \  
  --key-identifier arn:aws:payment-cryptography:us-east-2:123456789012:key/  
  kwapwa6qaiifllw2h
```

输出：

```
{  
  "Key": {
```

```
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:123456789012:key/
    kwapwa6qaiifllw2h",
    "KeyAttributes": {
      "KeyUsage": "TR31_V2_VISA_PIN_VERIFICATION_KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyAlgorithm": "TDES_3KEY",
      "KeyModesOfUse": {
        "Encrypt": false,
        "Decrypt": false,
        "Wrap": false,
        "Unwrap": false,
        "Generate": true,
        "Sign": false,
        "Verify": true,
        "DeriveKey": false,
        "NoRestrictions": false
      }
    },
    "KeyCheckValue": "",
    "KeyCheckValueAlgorithm": "ANSI_X9_24",
    "Enabled": false,
    "Exportable": true,
    "KeyState": "CREATE_COMPLETE",
    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
    "CreateTimestamp": "1686800690",
    "UsageStopTimestamp": "1687405998"
  }
}
```

有关更多信息，请参阅《AWS 支付加密用户指南》中的[删除密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RestoreKey](#)中的。

## start-key-usage

以下代码示例演示了如何使用 start-key-usage。

### AWS CLI

启用密钥

以下 start-key-usage 示例允许使用密钥。

```
aws payment-cryptography start-key-usage \
```



```
--key-identifier arn:aws:payment-cryptography:us-east-2:123456789012:key/
kwapwa6qaifl1lw2h
```

输出：

```
{
  "Key": {
    "CreateTimestamp": "1686800690",
    "Enabled": true,
    "Exportable": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
alsuwfxug3pgy6xh",
    "KeyAttributes": {
      "KeyAlgorithm": "TDES_3KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyModesOfUse": {
        "Decrypt": true,
        "DeriveKey": false,
        "Encrypt": true,
        "Generate": false,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": true,
        "Verify": false,
        "Wrap": true
      },
      "KeyUsage": "TR31_P1_PIN_GENERATION_KEY"
    },
    "KeyCheckValue": "369D",
    "KeyCheckValueAlgorithm": "ANSI_X9_24",
    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
    "KeyState": "CREATE_COMPLETE",
    "UsageStartTimestamp": "1686800690"
  }
}
```

有关更多信息，请参阅《AWS 支付加密用户指南》中的[启用和禁用密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartKeyUsage](#) 中的。

## stop-key-usage

以下代码示例演示了如何使用 stop-key-usage。

## AWS CLI

### 禁用密钥

以下`stop-key-usage`示例禁用密钥。

```
aws payment-cryptography stop-key-usage \  
  --key-identifier arn:aws:payment-cryptography:us-east-2:123456789012:key/  
  kwapwa6qaif1lw2h
```

输出：

```
{  
  "Key": {  
    "CreateTimestamp": "1686800690",  
    "Enabled": true,  
    "Exportable": true,  
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/  
alsuwxug3pgy6xh",  
    "KeyAttributes": {  
      "KeyAlgorithm": "TDES_3KEY",  
      "KeyClass": "SYMMETRIC_KEY",  
      "KeyModesOfUse": {  
        "Decrypt": true,  
        "DeriveKey": false,  
        "Encrypt": true,  
        "Generate": false,  
        "NoRestrictions": false,  
        "Sign": false,  
        "Unwrap": true,  
        "Verify": false,  
        "Wrap": true  
      },  
      "KeyUsage": "TR31_P1_PIN_GENERATION_KEY"  
    },  
    "KeyCheckValue": "369D",  
    "KeyCheckValueAlgorithm": "ANSI_X9_24",  
    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",  
    "KeyState": "CREATE_COMPLETE",  
    "UsageStartTimestamp": "1686800690"  
  }  
}
```

有关更多信息，请参阅《AWS 支付加密用户指南》中的[启用和禁用密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopKeyUsage](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为密钥添加标签

以下tag-resource示例为密钥添加了标签。

```
aws payment-cryptography tag-resource \  
  --resource-arn arn:aws:payment-cryptography:us-east-2:123456789012:key/  
  kwapwa6qaif1lw2h \  
  --tags Key=sampleTag,Value=sampleValue
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 支付加密用户指南》中的[管理密钥标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从密钥中移除标签

以下untag-resource示例从密钥中删除标签。

```
aws payment-cryptography untag-resource \  
  --resource-arn arn:aws:payment-cryptography:us-east-2:123456789012:key/  
  kwapwa6qaif1lw2h \  
  --tag-keys sampleTag
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 支付加密用户指南》中的[管理密钥标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-alias

以下代码示例演示了如何使用 update-alias。

### AWS CLI

#### 更新别名

以下update-alias示例将别名与其他密钥相关联。

```
aws payment-cryptography update-alias \  
  --alias-name alias/sampleAlias1 \  
  --key-arn arn:aws:payment-cryptography:us-east-2:123456789012:key/  
  tqv5yij6wtxx64pi
```

输出：

```
{  
  "Alias": {  
    "AliasName": "alias/sampleAlias1",  
    "KeyArn": "arn:aws:payment-cryptography:us-west-2:123456789012:key/  
    tqv5yij6wtxx64pi "  
  }  
}
```

有关更多信息，请参阅《AWS 支付密码学用户指南》中的[关于别名](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateAlias](#)中的。

## AWS Payment Cryptography 使用数据平面示例 AWS CLI

以下代码示例向您展示了如何使用 with D AWS Payment Cryptography ata Plane 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## decrypt-data

以下代码示例演示了如何使用 decrypt-data。

AWS CLI

解密密文

以下decrypt-data示例使用对称密钥解密密文数据。要执行此操作，密钥必须KeyModesOfUse设置为，Decrypt并KeyUsage设置为TR31\_D0\_SYMMETRIC\_DATA\_ENCRYPTION\_KEY。

```
aws payment-cryptography-data decrypt-data \
  --key-identifier arn:aws:payment-cryptography:us-east-2:123456789012:key/
  kwapwa6qaif1lw2h \
  --cipher-text 33612AB9D6929C3A828EB6030082B2BD \
  --decryption-attributes 'Symmetric={Mode=CBC}'
```

输出：

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:123456789012:key/
  kwapwa6qaif1lw2h",
  "KeyCheckValue": "71D7AE",
  "PlainText": "31323334313233343132333431323334"
}
```

有关更多信息，请参阅《AWS 支付加密[用户指南](#)》中的[解密数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DecryptData](#)中的。

## encrypt-data

以下代码示例演示了如何使用 encrypt-data。

### AWS CLI

#### 要加密数据

以下encrypt-data示例使用对称密钥对纯文本数据进行加密。要执行此操作，密钥必须KeyModesOfUse设置为，Encrypt并KeyUsage设置为TR31\_D0\_SYMMETRIC\_DATA\_ENCRYPTION\_KEY。

```
aws payment-cryptography-data encrypt-data \
  --key-identifier arn:aws:payment-cryptography:us-east-2:123456789012:key/
  kwapwa6qaifllw2h \
  --plain-text 31323334313233343132333431323334 \
  --encryption-attributes 'Symmetric={Mode=CBC}'
```

#### 输出：

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:123456789012:key/
  kwapwa6qaifllw2h",
  "KeyCheckValue": "71D7AE",
  "CipherText": "33612AB9D6929C3A828EB6030082B2BD"
}
```

有关更多信息，请参阅《AWS 支付加密用户指南》中的加密[数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EncryptData](#)中的。

## generate-card-validation-data

以下代码示例演示了如何使用 generate-card-validation-data。

### AWS CLI

#### 生成 CVV

以下generate-card-validation-data示例生成 CVV/CVV2。

```
aws payment-cryptography-data generate-card-validation-data \
```

```
--key-identifier arn:aws:payment-cryptography:us-east-2:123456789012:key/
kwapwa6qaiifllw2h \
--primary-account-number=171234567890123 \
--generation-attributes CardVerificationValue2={CardExpiryDate=0123}
```

输出：

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:123456789012:key/
kwapwa6qaiifllw2h",
  "KeyCheckValue": "CADD1",
  "ValidationData": "801"
}
```

有关更多信息，请参阅《AWS 支付密码学用户指南》中的[生成银行卡数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GenerateCardValidationData](#)中的。

## generate-mac

以下代码示例演示了如何使用 generate-mac。

### AWS CLI

#### 生成 MAC

以下generate-card-validation-data示例使用算法 HMAC\_SHA256 和 HMAC 加密密钥生成用于卡数据身份验证的基于哈希的消息身份验证码 (HMAC)。密钥必须KeyUsage设置为，TR31\_M7\_HMAC\_KEY并设置KeyModesOfUse为Generate。

```
aws payment-cryptography-data generate-mac \
  --key-identifier arn:aws:payment-cryptography:us-east-2:123456789012:key/
kwapwa6qaiifllw2h \
  --message-data
  "3b313038383439303031303733393431353d32343038323236303030373030303f33" \
  --generation-attributes Algorithm=HMAC_SHA256
```

输出：

```
{
```

```
"KeyArn": "arn:aws:payment-cryptography:us-east-2:123456789012:key/kwapwa6qaiflw2h,
"KeyCheckValue": "2976E7",
"Mac": "ED87F26E961C6D0DDB78DA5038AA2BDDEA0DCE03E5B5E96BDDD494F4A7AA470C"
}
```

有关更多信息，请参阅《AWS 支付加密用户指南》中的“[生成 MAC](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GenerateMac](#) 中的。

## generate-pin-data

以下代码示例演示了如何使用 generate-pin-data。

### AWS CLI

#### 生成 PIN

以下 generate-card-validation-data 示例使用 Visa PIN 方案生成一个新的随机 PIN 码。

```
aws payment-cryptography-data generate-pin-data \
  --generation-key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2 \
  --encryption-key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsuplneuyt \
  --primary-account-number 171234567890123 \
  --pin-block-format ISO_FORMAT_0 \
  --generation-attributes VisaPin={PinVerificationKeyIndex=1}
```

#### 输出：

```
{
  "GenerationKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2",
  "GenerationKeyCheckValue": "7F2363",
  "EncryptionKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsuplneuyt",
  "EncryptionKeyCheckValue": "7CC9E2",
  "EncryptedPinBlock": "AC17DC148BDA645E",
  "PinData": {
    "VerificationValue": "5507"
  }
}
```



```
}
```

有关更多信息，请参阅《AWS 支付加密用户指南》中的[生成 PIN 数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GeneratePinData](#)中的。

## re-encrypt-data

以下代码示例演示了如何使用 re-encrypt-data。

### AWS CLI

使用不同的密钥重新加密数据

以下re-encrypt-data示例对使用 AES 对称密钥加密的密文进行解密，并使用每笔交易派生的唯一密钥 (DUKPT) 密钥对其进行重新加密。

```
aws payment-cryptography-data re-encrypt-data \
  --incoming-key-identifier arn:aws:payment-cryptography:us-
west-2:111122223333:key/hyv7ymboitd4vfy \
  --outgoing-key-identifier arn:aws:payment-cryptography:us-
west-2:111122223333:key/jl6ythkcvzesbxen \
  --cipher-text
4D2B0BDBA192D5AEFEAA5B3EC28E4A65383C313FFA25140101560F75FE1B99F27192A90980AB9334 \
  --incoming-encryption-attributes
"Dukpt={Mode=ECB,KeySerialNumber=012345678911111}" \
  --outgoing-encryption-attributes '{"Symmetric": {"Mode": "ECB"}}'
```

输出：

```
{
  "CipherText":
"F94959DA30EEFF0C035483C6067667CF6796E3C1AD28C2B61F9CFEB772A8DD41C0D6822931E0D3B1",
  "KeyArn": "arn:aws:payment-cryptography:us-west-2:111122223333:key/
jl6ythkcvzesbxen",
  "KeyCheckValue": "2E8CD9"
}
```

有关更多信息，请参阅《AWS 支付密码学用户指南》中的[加密和解密数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReEncryptData](#)中的。

## translate-pin-data

以下代码示例演示了如何使用 `translate-pin-data`。

### AWS CLI

#### 转换 PIN 数据

以下 `translate-pin-data` 示例使用 DUKPT 算法将 PIN 从使用 ISO 0 PIN 块的 PEK TDES 加密转换为使用 DUKPT 算法的 AES ISO 4 PIN 块。

```
aws payment-cryptography-data translate-pin-data \
  --encrypted-pin-block "AC17DC148BDA645E" \
  --incoming-translation-
attributes=IsoFormat0='{PrimaryAccountNumber=171234567890123}' \
  --incoming-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt \
  --outgoing-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/4pmyquwjs3yj4vwe \
  --outgoing-translation-attributes
IsoFormat4="{PrimaryAccountNumber=171234567890123}" \
  --outgoing-dukpt-attributes KeySerialNumber="FFFF9876543210E00008"
```

输出：

```
{
  "PinBlock": "1F4209C670E49F83E75CC72E81B787D9",
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt
  "KeyCheckValue": "7CC9E2"
}
```

有关更多信息，请参阅《AWS 支付加密用户指南》中的 [Translate PIN 数据](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TranslatePinData](#) 中的。

## verify-auth-request-cryptogram

以下代码示例演示了如何使用 `verify-auth-request-cryptogram`。

### AWS CLI

#### 验证身份验证请求

以下verify-auth-request-cryptogram示例验证授权请求密码 (ARQC)。

```
aws payment-cryptography-data verify-auth-request-cryptogram \  
  --auth-request-cryptogram F6E1BD1E6037FB3E \  
  --auth-response-attributes '{"ArpcMethod1": {"AuthResponseCode": "1111"}}' \  
  --key-identifier arn:aws:payment-cryptography:us-west-2:111122223333:key/  
pboipdfzd4mdklya \  
  --major-key-derivation-mode "EMV_OPTION_A" \  
  --session-key-derivation-attributes '{"EmvCommon":  
{"ApplicationTransactionCounter": "1234", "PanSequenceNumber":  
"01", "PrimaryAccountNumber": "471234567890123"}}' \  
  --transaction-data "123456789ABCDEF"
```

输出：

```
{  
  "AuthResponseValue": "D899B8C6FBF971AA",  
  "KeyArn": "arn:aws:payment-cryptography:us-west-2:111122223333:key/  
pboipdfzd4mdklya",  
  "KeyCheckValue": "985792"  
}
```

有关更多信息，请参阅《AWS 支付[密码学用户指南](#)》中的[验证身份验证请求 \(ARQC\) 密码](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[VerifyAuthRequestCryptogram](#)中的。

## verify-card-validation-data

以下代码示例演示了如何使用 verify-card-validation-data。

### AWS CLI

#### 验证 CVV

以下verify-card-validation-data示例验证了 PAN 的 CVV/CVV2。

```
aws payment-cryptography-data verify-card-validation-data \  
  --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/  
tqv5yij6wtxx64pi \  
  --primary-account-number=171234567890123 \  
  --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \  

```

```
--validation-data 801
```

输出：

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
  tqv5yij6wtxx64pi",
  "KeyCheckValue": "CADD1"
}
```

有关更多信息，请参阅《AWS 支付密码学用户指南》中的[验证银行卡数据](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[VerifyCardValidationData](#)中的。

## verify-mac

以下代码示例演示了如何使用 verify-mac。

### AWS CLI

#### 验证 MAC

以下 verify-mac 示例使用算法 HMAC\_SHA256 和 HMAC 加密密钥验证用于卡数据身份验证的基于哈希的消息身份验证码 (HMAC)。

```
aws payment-cryptography-data verify-mac \
  --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
  qnobl5lghrzunce6 \
  --message-data
  "3b343038383439303031303733393431353d32343038323236303030373030303f33" \
  --verification-attributes='Algorithm=HMAC_SHA256' \
  --mac ED87F26E961C6D0DDB78DA5038AA2BDDEA0DCE03E5B5E96BDDD494F4A7AA470C
```

输出：

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
  qnobl5lghrzunce6,
  "KeyCheckValue": "2976E7",
}
```

有关更多信息，请参阅《AWS 支付加密用户指南》中的[“验证 MAC”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[VerifyMac](#)中的。

## verify-pin-data

以下代码示例演示了如何使用 verify-pin-data。

### AWS CLI

#### 验证 PIN

以下verify-pin-data示例验证了 PAN 的 PIN。

```
aws payment-cryptography-data verify-pin-data \
  --verification-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2 \
  --encryption-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt \
  --primary-account-number 171234567890123 \
  --pin-block-format ISO_FORMAT_0 \
  --verification-attributes
  VisaPin="{PinVerificationKeyIndex=1,VerificationValue=5507}" \
  --encrypted-pin-block AC17DC148BDA645E
```

输出：

```
{
  "VerificationKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
  "VerificationKeyCheckValue": "7F2363",
  "EncryptionKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
  "EncryptionKeyCheckValue": "7CC9E2",
}
```

有关更多信息，请参阅《AWS 支付加密用户指南》中的[验证 PIN 数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[VerifyPinData](#)中的。

## 使用 Amazon Pinpoint 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon Pinpoint 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## create-app

以下代码示例演示了如何使用 create-app。

AWS CLI

示例 1：创建应用程序

以下 create-app 示例创建一个新的应用程序（项目）。

```
aws pinpoint create-app \  
  --create-application-request Name=ExampleCorp
```

输出：

```
{  
  "ApplicationResponse": {  
    "Arn": "arn:aws:mobiletargeting:us-  
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example",  
    "Id": "810c7aab86d42fb2b56c8c966example",  
    "Name": "ExampleCorp",  
    "tags": {}  
  }  
}
```

示例 2：创建带有标签的应用程序

以下 create-app 示例创建一个新的应用程序（项目），并将标签（键和值）与该应用程序关联。

```
aws pinpoint create-app \  
  --create-application-request Name=ExampleCorp,tags={"Stack"="Test"}
```

输出：

```
{  
  "ApplicationResponse": {  
    "Arn": "arn:aws:mobiletargeting:us-  
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example",  
    "Id": "810c7aab86d42fb2b56c8c966example",  
    "Name": "ExampleCorp",  
    "tags": {  
      "Stack": "Test"  
    }  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateApp](#)中的。

## create-sms-template

以下代码示例演示了如何使用 `create-sms-template`。

### AWS CLI

为通过 SMS 渠道发送的消息创建消息模板

以下`create-sms-template`示例创建了一个 SMS 消息模板。

```
aws pinpoint create-sms-template \  
  --template-name TestTemplate \  
  --sms-template-request file://myfile.json \  
  --region us-east-1
```

`myfile.json` 的内容：

```
{  
  "Body": "hello\n how are you?\n food is good",  
  "TemplateDescription": "Test SMS Template"  
}
```

输出：

```
{
  "CreateTemplateMessageBody": {
    "Arn": "arn:aws:mobiletargeting:us-east-1:AIDACKCEVSQ6C2EXAMPLE:templates/
TestTemplate/SMS",
    "Message": "Created",
    "RequestID": "8c36b17f-a0b0-400f-ac21-29e9b62a975d"
  }
}
```

有关更多信息，请参阅[亚马逊 Pinpoint 用户指南中的亚马逊 Pinpoint 消息模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSmsTemplate](#)中的。

## delete-app

以下代码示例演示了如何使用 delete-app。

### AWS CLI

#### 删除应用程序

以下 delete-app 示例删除一个应用程序（项目）。

```
aws pinpoint delete-app \
  --application-id 810c7aab86d42fb2b56c8c966example
```

输出：

```
{
  "ApplicationResponse": {
    "Arn": "arn:aws:mobiletargeting:us-
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example",
    "Id": "810c7aab86d42fb2b56c8c966example",
    "Name": "ExampleCorp",
    "tags": {}
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteApp](#)中的。



## get-apns-channel

以下代码示例演示了如何使用 `get-apns-channel`。

### AWS CLI

检索有关应用程序的 APnS 频道状态和设置的信息

以下`get-apns-channel`示例检索有关应用程序的 APNs 频道的状态和设置的信息。

```
aws pinpoint get-apns-channel \  
  --application-id 9ab1068eb0a6461c86cce7f27ce0efd7 \  
  --region us-east-1
```

输出：

```
{  
  "APNSChannelResponse": {  
    "ApplicationId": "9ab1068eb0a6461c86cce7f27ce0efd7",  
    "CreationDate": "2019-05-09T21:54:45.082Z",  
    "DefaultAuthenticationMethod": "CERTIFICATE",  
    "Enabled": true,  
    "HasCredential": true,  
    "HasTokenKey": false,  
    "Id": "apns",  
    "IsArchived": false,  
    "LastModifiedDate": "2019-05-09T22:04:01.067Z",  
    "Platform": "APNS",  
    "Version": 2  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetApnsChannel](#)中的。

## get-app

以下代码示例演示了如何使用 `get-app`。

### AWS CLI

检索有关应用程序（项目）的信息

以下`get-app`示例检索有关应用程序（项目）的信息。

```
aws pinpoint get-app \  
  --application-id 810c7aab86d42fb2b56c8c966example \  
  --region us-east-1
```

输出：

```
{  
  "ApplicationResponse": {  
    "Arn": "arn:aws:mobiletargeting:us-  
east-1:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example",  
    "Id": "810c7aab86d42fb2b56c8c966example",  
    "Name": "ExampleCorp",  
    "tags": {  
      "Year": "2019",  
      "Stack": "Production"  
    }  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetApp](#)中的。

## get-apps

以下代码示例演示了如何使用 get-apps。

### AWS CLI

检索所有应用程序的相关信息

以下get-apps示例检索有关您的所有应用程序（项目）的信息。

```
aws pinpoint get-apps
```

输出：

```
{  
  "ApplicationsResponse": {  
    "Item": [  
      {  
        "Arn": "arn:aws:mobiletargeting:us-  
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example",  
        "Id": "810c7aab86d42fb2b56c8c966example",
```

```

        "Name": "ExampleCorp",
        "tags": {
            "Year": "2019",
            "Stack": "Production"
        }
    },
    {
        "Arn": "arn:aws:mobiletargeting:us-
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/42d8c7eb0990a57ba1d5476a3example",
        "Id": "42d8c7eb0990a57ba1d5476a3example",
        "Name": "AnyCompany",
        "tags": {}
    },
    {
        "Arn": "arn:aws:mobiletargeting:us-
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/80f5c382b638ffe5ad12376bbexample",
        "Id": "80f5c382b638ffe5ad12376bbexample",
        "Name": "ExampleCorp_Test",
        "tags": {
            "Year": "2019",
            "Stack": "Test"
        }
    }
],
"NextToken":
"eyJJdcmVhdGlvbkRhdGUiOiIyMDE5LTA3LTE2VDE0jM40jUzLjkwM1oiLCJBY2NvdW50SWQiOiI1MTIzOTcxODM4Nz"
}
}

```

NextToken响应值的存在表明还有更多可用的输出。再次调用该命令并提供该值作为NextToken输入参数。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetApps](#)中的。

## get-campaign

以下代码示例演示了如何使用 get-campaign。

### AWS CLI

检索有关活动的状态、配置和其他设置的信息

以下get-campaign示例检索有关活动的状态、配置和其他设置的信息。

```
aws pinpoint get-campaign \
  --application-id 6e0b7591a90841d2b5d93fa11143e5a7 \
  --campaign-id a1e63c6cc0eb43ed826ffcc3cc90b30d \
  --region us-east-1
```

输出：

```
{
  "CampaignResponse": {
    "AdditionalTreatments": [],
    "ApplicationId": "6e0b7591a90841d2b5d93fa11143e5a7",
    "Arn": "arn:aws:mobiletargeting:us-
east-1:AIDACKCEVSQ6C2EXAMPLE:apps/6e0b7591a90841d2b5d93fa11143e5a7/campaigns/
a1e63c6cc0eb43ed826ffcc3cc90b30d",
    "CreationDate": "2019-10-08T18:40:16.581Z",
    "Description": " ",
    "HoldoutPercent": 0,
    "Id": "a1e63c6cc0eb43ed826ffcc3cc90b30d",
    "IsPaused": false,
    "LastModifiedDate": "2019-10-08T18:40:16.581Z",
    "Limits": {
      "Daily": 0,
      "MaximumDuration": 60,
      "MessagesPerSecond": 50,
      "Total": 0
    },
    "MessageConfiguration": {
      "EmailMessage": {
        "FromAddress": "sender@example.com",
        "HtmlBody": "<!DOCTYPE html>\n <html lang=\\"en\">\n <head>\n
<meta http-equiv=\\"Content-Type\" content=\\"text/html; charset=utf-8\" />\n</head>
\n<body>Hello</body>\n</html>",
        "Title": "PinpointDemo"
      }
    },
    "Name": "MyCampaign",
    "Schedule": {
      "IsLocalTime": false,
      "StartTime": "IMMEDIATE",
      "Timezone": "utc"
    },
    "SegmentId": "b66c9e42f71444b2aa2e0fffc1df28f60",
    "SegmentVersion": 1,
  }
}
```

```
    "State": {
      "CampaignStatus": "COMPLETED"
    },
    "tags": {},
    "TemplateConfiguration": {},
    "Version": 1
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCampaign](#)中的。

## get-campaigns

以下代码示例演示了如何使用 get-campaigns。

### AWS CLI

检索与应用程序关联的所有活动的状态、配置和其他设置的相关信息

以下get-campaigns示例检索与应用程序关联的所有活动的状态、配置和其他设置的相关信息。

```
aws pinpoint get-campaigns \
  --application-id 6e0b7591a90841d2b5d93fa11143e5a7 \
  --region us-east-1
```

输出：

```
{
  "CampaignsResponse": {
    "Item": [
      {
        "AdditionalTreatments": [],
        "ApplicationId": "6e0b7591a90841d2b5d93fa11143e5a7",
        "Arn": "arn:aws:mobiletargeting:us-east-1:AIDACKCEVSQ6C2EXAMPLE:apps/6e0b7591a90841d2b5d93fa11143e5a7/campaigns/7e1280344c8f4a9aa40a00b006fe44f1",
        "CreationDate": "2019-10-08T18:40:22.905Z",
        "Description": " ",
        "HoldoutPercent": 0,
        "Id": "7e1280344c8f4a9aa40a00b006fe44f1",
        "IsPaused": false,
        "LastModifiedDate": "2019-10-08T18:40:22.905Z",
        "Limits": {},

```

```

    "MessageConfiguration": {
      "EmailMessage": {
        "FromAddress": "sender@example.com",
        "HtmlBody": "<!DOCTYPE html>\n  <html lang=\n
\n  <head>\n  <meta http-equiv=\n\"Content-Type\n\" content=\n\"text/html;\n
charset=utf-8\n\" />\n</head>\n<body>Hello</body>\n</html>",
        "Title": "PinpointDemo Test"
      }
    },
    "Name": "MyCampaign1",
    "Schedule": {
      "IsLocalTime": false,
      "QuietTime": {},
      "StartTime": "IMMEDIATE",
      "Timezone": "UTC"
    },
    "SegmentId": "b66c9e42f71444b2aa2e0ffc1df28f60",
    "SegmentVersion": 1,
    "State": {
      "CampaignStatus": "COMPLETED"
    },
    "tags": {},
    "TemplateConfiguration": {},
    "Version": 1
  },
  {
    "AdditionalTreatments": [],
    "ApplicationId": "6e0b7591a90841d2b5d93fa11143e5a7",
    "Arn": "arn:aws:mobiletargeting:us-
east-1:AIDACKCEVSQ6C2EXAMPLE:apps/6e0b7591a90841d2b5d93fa11143e5a7/campaigns/
a1e63c6cc0eb43ed826ffcc3cc90b30d",
    "CreationDate": "2019-10-08T18:40:16.581Z",
    "Description": " ",
    "HoldoutPercent": 0,
    "Id": "a1e63c6cc0eb43ed826ffcc3cc90b30d",
    "IsPaused": false,
    "LastModifiedDate": "2019-10-08T18:40:16.581Z",
    "Limits": {
      "Daily": 0,
      "MaximumDuration": 60,
      "MessagesPerSecond": 50,
      "Total": 0
    },
    "MessageConfiguration": {

```

```

        "EmailMessage": {
            "FromAddress": "sender@example.com",
            "HtmlBody": "<!DOCTYPE html>\n    <html lang=\"en
\n    <head>\n    <meta http-equiv=\"Content-Type\" content=\"text/html;
charset=utf-8\" />\n</head>\n<body>Demo</body>\n</html>",
            "Title": "PinpointDemo"
        }
    },
    "Name": "MyCampaign2",
    "Schedule": {
        "IsLocalTime": false,
        "StartTime": "IMMEDIATE",
        "Timezone": "utc"
    },
    "SegmentId": "b66c9e42f71444b2aa2e0ffc1df28f60",
    "SegmentVersion": 1,
    "State": {
        "CampaignStatus": "COMPLETED"
    },
    "tags": {},
    "TemplateConfiguration": {},
    "Version": 1
    }
}
]
}
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCampaigns](#)中的。

## get-channels

以下代码示例演示了如何使用 get-channels。

### AWS CLI

检索有关应用程序每个频道的历史和状态的信息

以下get-channels示例检索有关应用程序每个频道的历史和状态的信息。

```

aws pinpoint get-channels \
  --application-id 6e0b7591a90841d2b5d93fa11143e5a7 \
  --region us-east-1

```

输出：

```
{
  "ChannelsResponse": {
    "Channels": {
      "GCM": {
        "ApplicationId": "6e0b7591a90841d2b5d93fa11143e5a7",
        "CreationDate": "2019-10-08T18:28:23.182Z",
        "Enabled": true,
        "HasCredential": true,
        "Id": "gcm",
        "IsArchived": false,
        "LastModifiedDate": "2019-10-08T18:28:23.182Z",
        "Version": 1
      },
      "SMS": {
        "ApplicationId": "6e0b7591a90841d2b5d93fa11143e5a7",
        "CreationDate": "2019-10-08T18:39:18.511Z",
        "Enabled": true,
        "Id": "sms",
        "IsArchived": false,
        "LastModifiedDate": "2019-10-08T18:39:18.511Z",
        "Version": 1
      },
      "EMAIL": {
        "ApplicationId": "6e0b7591a90841d2b5d93fa11143e5a7",
        "CreationDate": "2019-10-08T18:27:23.990Z",
        "Enabled": true,
        "Id": "email",
        "IsArchived": false,
        "LastModifiedDate": "2019-10-08T18:27:23.990Z",
        "Version": 1
      },
      "IN_APP": {
        "Enabled": true,
        "IsArchived": false,
        "Version": 0
      }
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetChannels](#)中的。



## get-email-channel

以下代码示例演示了如何使用 `get-email-channel`。

### AWS CLI

检索有关应用程序电子邮件渠道状态和设置的信息

以下`get-email-channel`示例检索应用程序电子邮件渠道的状态和设置。

```
aws pinpoint get-email-channel \  
  --application-id 6e0b7591a90841d2b5d93fa11143e5a7 \  
  --region us-east-1
```

输出：

```
{  
  "EmailChannelResponse": {  
    "ApplicationId": "6e0b7591a90841d2b5d93fa11143e5a7",  
    "CreationDate": "2019-10-08T18:27:23.990Z",  
    "Enabled": true,  
    "FromAddress": "sender@example.com",  
    "Id": "email",  
    "Identity": "arn:aws:ses:us-east-1:AIDACKCEVSQ6C2EXAMPLE:identity/  
sender@example.com",  
    "IsArchived": false,  
    "LastModifiedDate": "2019-10-08T18:27:23.990Z",  
    "MessagesPerSecond": 1,  
    "Platform": "EMAIL",  
    "RoleArn": "arn:aws:iam::AIDACKCEVSQ6C2EXAMPLE:role/pinpoint-events",  
    "Version": 1  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetEmailChannel](#)中的。

## get-endpoint

以下代码示例演示了如何使用 `get-endpoint`。

### AWS CLI

检索有关应用程序特定端点的设置和属性的信息

以下 `get-endpoint` 示例检索有关应用程序特定端点的设置和属性的信息。

```
aws pinpoint get-endpoint \  
  --application-id 611e3e3cdd47474c9c1399a505665b91 \  
  --endpoint-id testendpoint \  
  --region us-east-1
```

输出：

```
{  
  "EndpointResponse": {  
    "Address": "+11234567890",  
    "ApplicationId": "611e3e3cdd47474c9c1399a505665b91",  
    "Attributes": {},  
    "ChannelType": "SMS",  
    "CohortId": "63",  
    "CreationDate": "2019-01-28T23:55:11.534Z",  
    "EffectiveDate": "2021-08-06T00:04:51.763Z",  
    "EndpointStatus": "ACTIVE",  
    "Id": "testendpoint",  
    "Location": {  
      "Country": "USA"  
    },  
    "Metrics": {  
      "SmsDelivered": 1.0  
    },  
    "OptOut": "ALL",  
    "RequestId": "a204b1f2-7e26-48a7-9c80-b49a2143489d",  
    "User": {  
      "UserAttributes": {  
        "Age": [  
          "24"  
        ]  
      },  
      "UserId": "testuser"  
    }  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetEndpoint](#)中的。

## get-gcm-channel

以下代码示例演示了如何使用 `get-gcm-channel`。

### AWS CLI

检索有关应用程序的 GCM 频道状态和设置的信息

以下 `get-gcm-channel` 示例检索有关应用程序的 GCM 频道的状态和设置的信息。

```
aws pinpoint get-gcm-channel \  
  --application-id 6e0b7591a90841d2b5d93fa11143e5a7 \  
  --region us-east-1
```

输出：

```
{  
  "GCMChannelResponse": {  
    "ApplicationId": "6e0b7591a90841d2b5d93fa11143e5a7",  
    "CreationDate": "2019-10-08T18:28:23.182Z",  
    "Enabled": true,  
    "HasCredential": true,  
    "Id": "gcm",  
    "IsArchived": false,  
    "LastModifiedDate": "2019-10-08T18:28:23.182Z",  
    "Platform": "GCM",  
    "Version": 1  
  }  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetGcmChannel](#) 中的。

## get-sms-channel

以下代码示例演示了如何使用 `get-sms-channel`。

### AWS CLI

检索有关应用程序的短信渠道的状态和设置的信息

以下 `get-sms-channel` 示例检索应用程序的短信渠道的状态和设置。

```
aws pinpoint get-sms-channel \  
  --application-id 6e0b7591a90841d2b5d93fa11143e5a7 \  
  --region us-east-1
```

```
--application-id 6e0b7591a90841d2b5d93fa11143e5a7 \  
--region us-east-1
```

输出：

```
{  
  "SMSChannelResponse": {  
    "ApplicationId": "6e0b7591a90841d2b5d93fa11143e5a7",  
    "CreationDate": "2019-10-08T18:39:18.511Z",  
    "Enabled": true,  
    "Id": "sms",  
    "IsArchived": false,  
    "LastModifiedDate": "2019-10-08T18:39:18.511Z",  
    "Platform": "SMS",  
    "PromotionalMessagesPerSecond": 20,  
    "TransactionalMessagesPerSecond": 20,  
    "Version": 1  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSmsChannel](#)中的。

## get-sms-template

以下代码示例演示了如何使用 get-sms-template。

### AWS CLI

检索通过 SMS 渠道发送的消息的消息模板的内容和设置

以下get-sms-template示例检索 SMS 消息模板的内容和设置。

```
aws pinpoint get-sms-template \  
  --template-name TestTemplate \  
  --region us-east-1
```

输出：

```
{  
  "SMSTemplateResponse": {  
    "Arn": "arn:aws:mobiletargeting:us-east-1:AIDACKCEVSQ6C2EXAMPLE:templates/  
TestTemplate/SMS",
```

```
    "Body": "hello\n how are you?\n food is good",
    "CreationDate": "2023-06-20T21:37:30.124Z",
    "LastModifiedDate": "2023-06-20T21:37:30.124Z",
    "tags": {},
    "TemplateDescription": "Test SMS Template",
    "TemplateName": "TestTemplate",
    "TemplateType": "SMS",
    "Version": "1"
  }
}
```

有关更多信息，请参阅[亚马逊 Pinpoint 用户指南中的亚马逊 Pinpoint 消息模板](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetSmsTemplate](#) 中的。

## get-voice-channel

以下代码示例演示了如何使用 get-voice-channel。

### AWS CLI

检索有关应用程序语音通道状态和设置的信息

以下 get-voice-channel 示例检索应用程序语音通道的状态和设置。

```
aws pinpoint get-voice-channel \
  --application-id 6e0b7591a90841d2b5d93fa11143e5a7 \
  --region us-east-1
```

输出：

```
{
  "VoiceChannelResponse": {
    "ApplicationId": "6e0b7591a90841d2b5d93fa11143e5a7",
    "CreationDate": "2022-04-28T00:17:03.836Z",
    "Enabled": true,
    "Id": "voice",
    "IsArchived": false,
    "LastModifiedDate": "2022-04-28T00:17:03.836Z",
    "Platform": "VOICE",
    "Version": 1
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVoiceChannel](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

检索资源的标签列表

以下`list-tags-for-resource`示例检索与指定资源关联的所有标签（密钥名称和值）。

```
aws pinpoint list-tags-for-resource \
  --resource-arn arn:aws:mobiletargeting:us-
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example
```

输出：

```
{
  "TagsModel": {
    "tags": {
      "Year": "2019",
      "Stack": "Production"
    }
  }
}
```

有关更多信息，请参阅《亚马逊 Pinpoint 开发者指南》中的“标记亚马逊 Pinpoint 资源”\_\_。 <  
<https://docs.aws.amazon.com/pinpoint/latest/developerguide/tagging-resources.html>>

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## phone-number-validate

以下代码示例演示了如何使用 `phone-number-validate`。

### AWS CLI

检索有关电话号码的信息

以下内容`phone-number-validate`检索有关电话号码的信息。

```
aws pinpoint phone-number-validate \  
  --number-validate-request PhoneNumber="+12065550142" \  
  --region us-east-1
```

输出：

```
{  
  "NumberValidateResponse": {  
    "Carrier": "ExampleCorp Mobile",  
    "City": "Seattle",  
    "CleansedPhoneNumberE164": "+12065550142",  
    "CleansedPhoneNumberNational": "2065550142",  
    "Country": "United States",  
    "CountryCodeIso2": "US",  
    "CountryCodeNumeric": "1",  
    "OriginalPhoneNumber": "+12065550142",  
    "PhoneType": "MOBILE",  
    "PhoneTypeCode": 0,  
    "Timezone": "America/Los_Angeles",  
    "ZipCode": "98101"  
  }  
}
```

有关更多信息，请参阅《Amazon Pinpoint 用户指南》中的 [Amazon Pinpoint SMS 渠道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PhoneNumberValidate](#)中的。

## send-messages

以下代码示例演示了如何使用 send-messages。

### AWS CLI

使用应用程序的端点发送短信

以下 send-messages 示例通过端点为应用程序发送直接消息。

```
aws pinpoint send-messages \  
  --application-id 611e3e3cdd47474c9c1399a505665b91 \  
  --message-request file://myfile.json \  
  --region us-west-2
```

myfile.json 的内容：

```
{
  "MessageConfiguration": {
    "SMSMessage": {
      "Body": "hello, how are you?"
    }
  },
  "Endpoints": {
    "testendpoint": {}
  }
}
```

输出：

```
{
  "MessageResponse": {
    "ApplicationId": "611e3e3cdd47474c9c1399a505665b91",
    "EndpointResult": {
      "testendpoint": {
        "Address": "+12345678900",
        "DeliveryStatus": "SUCCESSFUL",
        "MessageId": "itnuqhai5alf1n6ahv3udc05n7hhddr6gb3lq6g0",
        "StatusCode": 200,
        "StatusMessage": "MessageId:
itnuqhai5alf1n6ahv3udc05n7hhddr6gb3lq6g0"
      }
    },
    "RequestId": "c7e23264-04b2-4a46-b800-d24923f74753"
  }
}
```

有关更多信息，请参阅《Amazon Pinpoint 用户指南》中的 [Amazon Pinpoint SMS 渠道](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SendMessages](#) 中的。

## send-users-messages

以下代码示例演示了如何使用 send-users-messages。

### AWS CLI

为应用程序的用户发送 SMS 消息



以下send-users-messages示例为应用程序的用户发送了一条私信。

```
aws pinpoint send-users-messages \  
  --application-id 611e3e3cdd47474c9c1399a505665b91 \  
  --send-users-message-request file://myfile.json \  
  --region us-west-2
```

myfile.json 的内容：

```
{  
  "MessageConfiguration": {  
    "SMSMessage": {  
      "Body": "hello, how are you?"  
    }  
  },  
  "Users": {  
    "testuser": {}  
  }  
}
```

输出：

```
{  
  "SendUsersMessageResponse": {  
    "ApplicationId": "611e3e3cdd47474c9c1399a505665b91",  
    "RequestId": "e0b12cf5-2359-11e9-bb0b-d5fb91876b25",  
    "Result": {  
      "testuser": {  
        "testuserendpoint": {  
          "DeliveryStatus": "SUCCESSFUL",  
          "MessageId": "7qu4hk5bqhda3i7i2n4pjf98qcu8b7p45ifsmo0",  
          "StatusCode": 200,  
          "StatusMessage": "MessageId:  
7qu4hk5bqhda3i7i2n4pjf98qcu8b7p45ifsmo0",  
          "Address": "+12345678900"  
        }  
      }  
    }  
  }  
}
```

有关更多信息，请参阅《Amazon Pinpoint 用户指南》中的 [Amazon Pinpoint SMS 渠道](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SendUsersMessages](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源添加标签

以下示例向资源添加两个标签（密钥名称和值）。

```
aws pinpoint list-tags-for-resource \  
  --resource-arn arn:aws:mobiletargeting:us-  
east-1:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example \  
  --tags-model tags={Stack=Production,Year=2019}
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊 Pinpoint 开发者指南》中的“标记亚马逊 Pinpoint 资源”\_\_。 <  
<https://docs.aws.amazon.com/pinpoint/latest/developerguide/tagging-resources.html>>

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

示例 1：从资源中移除标签

以下 untag-resource 示例从资源中删除指定的标签（密钥名称和值）。

```
aws pinpoint untag-resource \  
  --resource-arn arn:aws:mobiletargeting:us-  
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example \  
  --tag-keys Year
```

此命令不生成任何输出。

示例 2：从资源中移除多个标签

以下 `untag-resource` 示例从资源中删除指定的标签（密钥名称和值）。

```
aws pinpoint untag-resource \  
  --resource-arn arn:aws:mobiletargeting:us-  
east-1:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example \  
  --tag-keys Year Stack
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊 Pinpoint 开发者指南》中的“标记亚马逊 Pinpoint 资源”\_\_。 <  
<https://docs.aws.amazon.com/pinpoint/latest/developerguide/tagging-resources.html>>

• 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-sms-channel

以下代码示例演示了如何使用 `update-sms-channel`。

### AWS CLI

启用短信频道或更新应用程序短信频道的状态和设置。

以下 `update-sms-channel` 示例为应用程序的 SMS 频道启用 SMS 频道。

```
aws pinpoint update-sms-channel \  
  --application-id 611e3e3cdd47474c9c1399a505665b91 \  
  --sms-channel-request Enabled=true \  
  --region us-west-2
```

输出：

```
{  
  "SMSChannelResponse": {  
    "ApplicationId": "611e3e3cdd47474c9c1399a505665b91",  
    "CreationDate": "2019-01-28T23:25:25.224Z",  
    "Enabled": true,  
    "Id": "sms",  
    "IsArchived": false,  
    "LastModifiedDate": "2023-05-18T23:22:50.977Z",  
    "Platform": "SMS",  
    "PromotionalMessagesPerSecond": 20,  
    "TransactionalMessagesPerSecond": 20,
```

```
    "Version": 3
  }
}
```

有关更多信息，请参阅《Amazon Pinpoint 用户指南》中的 [Amazon Pinpoint SMS 渠道](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateSmsChannel](#) 中的。

## 使用 Amazon Polly 的示例 AWS CLI

以下代码示例向您展示了如何在 Amazon Polly 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### delete-lexicon

以下代码示例演示了如何使用 delete-lexicon。

AWS CLI

删除词典

以下 delete-lexicon 示例将删除指定的词典。

```
aws polly delete-lexicon \
  --name w3c
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Polly 开发者指南中的[使用 DeleteLexicon 操作](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteLexicon](#)中的。

## get-lexicon

以下代码示例演示了如何使用 get-lexicon。

### AWS CLI

#### 检索词典的内容

以下 get-lexicon 示例检索指定的发音词典内容。

```
aws polly get-lexicon \  
  --name w3c
```

输出：

```
{  
  "Lexicon": {  
    "Content": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<lexicon version=  
\"1.0\" \n      xmlns=    \"http://www.w3.org/2005/01/pronunciation-lexicon  
\" \n      xmlns:xsi= \"http://www.w3.org/2001/XMLSchema-instance\" \n      xsi:schemaLocation= \"http://www.w3.org/2005/01/pronunciation-lexicon \n      http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\" \n      alphabet= \"ipa\" \n      xml:lang= \"en-US\">\n  <lexeme>\n    <grapheme>W3C</  
grapheme>\n    <alias>World Wide Web Consortium</alias>\n  </lexeme>\n</lexicon>  
\n",  
    "Name": "w3c"  
  },  
  "LexiconAttributes": {  
    "Alphabet": "ipa",  
    "LanguageCode": "en-US",  
    "LastModified": 1603908910.99,  
    "LexiconArn": "arn:aws:polly:us-west-2:880185128111:lexicon/w3c",  
    "LexemesCount": 1,  
    "Size": 492  
  }  
}
```

有关更多信息，请参阅 Amazon Polly 开发者指南中的[使用 GetLexicon 操作](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLexicon](#)中的。

## get-speech-synthesis-task

以下代码示例演示了如何使用 `get-speech-synthesis-task`。

### AWS CLI

获取有关语音合成任务的信息

以下 `get-speech-synthesis-task` 示例检索有关指定语音合成任务的信息。

```
aws polly get-speech-synthesis-task \  
  --task-id 70b61c0f-57ce-4715-a247-cae8729dcce9
```

输出：

```
{  
  "SynthesisTask": {  
    "TaskId": "70b61c0f-57ce-4715-a247-cae8729dcce9",  
    "TaskStatus": "completed",  
    "OutputUri": "https://s3.us-west-2.amazonaws.com/my-s3-  
bucket/70b61c0f-57ce-4715-a247-cae8729dcce9.mp3",  
    "CreationTime": 1603911042.689,  
    "RequestCharacters": 1311,  
    "OutputFormat": "mp3",  
    "TextType": "text",  
    "VoiceId": "Joanna"  
  }  
}
```

有关更多信息，请参阅《Amazon Polly 开发人员指南》中的[创建长音频文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSpeechSynthesisTask](#)中的。

## list-lexicons

以下代码示例演示了如何使用 `list-lexicons`。

### AWS CLI

列出您的词典

以下 `list-lexicons` 示例列出了您的发音词典。

```
aws polly list-lexicons
```

输出：

```
{
  "Lexicons": [
    {
      "Name": "w3c",
      "Attributes": {
        "Alphabet": "ipa",
        "LanguageCode": "en-US",
        "LastModified": 1603908910.99,
        "LexiconArn": "arn:aws:polly:us-east-2:123456789012:lexicon/w3c",
        "LexemesCount": 1,
        "Size": 492
      }
    }
  ]
}
```

有关更多信息，请参阅 Amazon Polly 开发者指南中的[使用 ListLexicons 操作](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListLexicons](#)中的。

## list-speech-synthesis-tasks

以下代码示例演示了如何使用 `list-speech-synthesis-tasks`。

AWS CLI

列出您的语音合成任务

以下 `list-speech-synthesis-tasks` 示例列出了您的语音合成任务。

```
aws polly list-speech-synthesis-tasks
```

输出：

```
{
  "SynthesisTasks": [
```

```
{
  "TaskId": "70b61c0f-57ce-4715-a247-cae8729dcce9",
  "TaskStatus": "completed",
  "OutputUri": "https://s3.us-west-2.amazonaws.com/my-s3-
bucket/70b61c0f-57ce-4715-a247-cae8729dcce9.mp3",
  "CreationTime": 1603911042.689,
  "RequestCharacters": 1311,
  "OutputFormat": "mp3",
  "TextType": "text",
  "VoiceId": "Joanna"
}
]
```

有关更多信息，请参阅《Amazon Polly 开发人员指南》中的[创建长音频文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSpeechSynthesisTasks](#)中的。

## put-lexicon

以下代码示例演示了如何使用 put-lexicon。

### AWS CLI

#### 存储词典

以下 put-lexicon 示例存储指定的发音词典。example.pls 文件指定符合 W3C PLS 标准的词典。

```
aws polly put-lexicon \
  --name w3c \
  --content file://example.pls
```

#### example.pls 的内容

```
{
  <?xml version="1.0" encoding="UTF-8"?>
  <lexicon version="1.0"
    xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
      http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
```



```
    alphabet="ipa"  
    xml:lang="en-US">  
      <lexeme>  
        <grapheme>W3C</grapheme>  
        <alias>World Wide Web Consortium</alias>  
      </lexeme>  
    </lexicon>  
  }  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Polly 开发者指南中的[使用 PutLexicon 操作](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[PutLexicon](#)中的。

## start-speech-synthesis-task

以下代码示例演示了如何使用 start-speech-synthesis-task。

### AWS CLI

#### 合成文本

以下 start-speech-synthesis-task 示例合成了 text\_file.txt 中的文本，并将生成的 MP3 文件存储在指定的存储桶中。

```
aws polly start-speech-synthesis-task \  
  --output-format mp3 \  
  --output-s3-bucket-name my-s3-bucket \  
  --text file://text_file.txt \  
  --voice-id Joanna
```

输出：

```
{  
  "SynthesisTask": {  
    "TaskId": "70b61c0f-57ce-4715-a247-cae8729dcce9",  
    "TaskStatus": "scheduled",  
    "OutputUri": "https://s3.us-east-2.amazonaws.com/my-s3-  
bucket/70b61c0f-57ce-4715-a247-cae8729dcce9.mp3",  
    "CreationTime": 1603911042.689,  
    "RequestCharacters": 1311,  
    "OutputFormat": "mp3",
```

```
        "TextType": "text",
        "VoiceId": "Joanna"
    }
}
```

有关更多信息，请参阅《Amazon Polly 开发人员指南》中的[创建长音频文件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartSpeechSynthesisTask](#) 中的。

## AWS 价目表 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS 价目表。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **describe-services**

以下代码示例演示了如何使用 `describe-services`。

AWS CLI

检索服务元数据

此示例检索 Amazon EC2 服务代码的元数据。

命令:

```
aws pricing describe-services --service-code AmazonEC2 --format-version aws_v1 --max-items 1
```

输出：

```
{
  "Services": [
    {
      "ServiceCode": "AmazonEC2",
      "AttributeNames": [
        "volumeType",
        "maxIopsvolume",
        "instance",
        "instanceCapacity10xlarge",
        "locationType",
        "instanceFamily",
        "operatingSystem",
        "clockSpeed",
        "LeaseContractLength",
        "ecu",
        "networkPerformance",
        "instanceCapacity8xlarge",
        "group",
        "maxThroughputvolume",
        "gpuMemory",
        "ebsOptimized",
        "elasticGpuType",
        "maxVolumeSize",
        "gpu",
        "processorFeatures",
        "intelAvxAvailable",
        "instanceCapacity4xlarge",
        "servicecode",
        "groupDescription",
        "processorArchitecture",
        "physicalCores",
        "productFamily",
        "enhancedNetworkingSupported",
        "intelTurboAvailable",
        "memory",
        "dedicatedEbsThroughput",
        "vcpu",
        "OfferingClass",
        "instanceCapacityLarge",
        "capacitystatus",
        "termType",
        "storage",
```

```
        "intelAvx2Available",
        "storageMedia",
        "physicalProcessor",
        "provisioned",
        "servicename",
        "PurchaseOption",
        "instanceCapacity18xlarge",
        "instanceType",
        "tenancy",
        "usagetype",
        "normalizationSizeFactor",
        "instanceCapacity2xlarge",
        "instanceCapacity16xlarge",
        "maxIopsBurstPerformance",
        "instanceCapacity12xlarge",
        "instanceCapacity32xlarge",
        "instanceCapacityXlarge",
        "licenseModel",
        "currentGeneration",
        "preInstalledSw",
        "location",
        "instanceCapacity24xlarge",
        "instanceCapacity9xlarge",
        "instanceCapacityMedium",
        "operation"
    ]
}
],
"FormatVersion": "aws_v1"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeServices](#)中的。

## get-attribute-values

以下代码示例演示了如何使用 `get-attribute-values`。

### AWS CLI

#### 检索属性值列表

以下`get-attribute-values`示例检索给定属性的可用值列表。

```
aws pricing get-attribute-values \  
  --service-code AmazonEC2 \  
  --attribute-name volumeType \  
  --max-items 2
```

输出：

```
{  
  "NextToken": "eyJ0ZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ==",  
  "AttributeValues": [  
    {  
      "Value": "Cold HDD"  
    },  
    {  
      "Value": "General Purpose"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAttributeValues](#)中的。

## get-products

以下代码示例演示了如何使用 get-products。

### AWS CLI

#### 检索产品列表

此示例检索符合给定标准的产品列表。

命令：

```
aws pricing get-products --filters file://filters.json --format-version aws_v1 --  
max-results 1 --service-code AmazonEC2
```

filters.json：

```
[  
  {  
    "Type": "TERM_MATCH",  
    "Field": "ServiceCode",
```

```

    "Value": "AmazonEC2"
  },
  {
    "Type": "TERM_MATCH",
    "Field": "volumeType",
    "Value": "Provisioned IOPS"
  }
]

```

输出：

```

{
  "FormatVersion": "aws_v1",
  "NextToken": "WGDY7ko8fQXdlaUZVdasFQ==:RVSagyIFn770XQ0zdUIc09BY6ucBG9itXAZGZF/
zioUz0sUKh6PCcPwa0yPZRiMePb986TeoKYB9l55fw/
CyoMq5ymnGmT1Vj39Tl1jbbAlhcqnVfTmPIilx8Uy5bdDaBYy/e/20fw9Edzsykbs8LTBUbNbiDQ
+BBds5yeI9AQkUepruKk3aEahFPxJ55kx/zk",
  "PriceList": [
    "{ \"product\": { \"productFamily\": \"Storage\", \"attributes\": { \"storageMedia\":
\\\"SSD-backed\\\", \"maxThroughputVolume\": \"320 MB/sec\", \"volumeType\": \"Provisioned
IOPS\", \"maxIopsVolume\": \"20000\", \"serviceCode\": \"AmazonEC2\", \"usageType
\": \"APS1-EBS:VolumeUsage.piops\", \"locationType\": \"AWS Region\", \"location\":
\\\"Asia Pacific (Singapore)\\\", \"serviceName\": \"Amazon Elastic Compute Cloud\",
\\\"maxVolumeSize\": \"16 TiB\", \"operation\": \"\" }, \"sku\": \"3MKHN58N7RDDVGKJ\" },
\\\"serviceCode\": \"AmazonEC2\", \"terms\": { \"OnDemand\": { \"3MKHN58N7RDDVGKJ.JRTCKXETXF
\": { \"priceDimensions\": { \"3MKHN58N7RDDVGKJ.JRTCKXETXF.6YS6EN2CT7\": { \"unit\": \"GB-
Mo\", \"endRange\": \"Inf\", \"description\": \"$0.138 per GB-month of Provisioned
IOPS SSD (io1) provisioned storage - Asia Pacific (Singapore)\", \"appliesTo
\": [], \"rateCode\": \"3MKHN58N7RDDVGKJ.JRTCKXETXF.6YS6EN2CT7\", \"beginRange\":
\\\"0\", \"pricePerUnit\": { \"USD\": \"0.1380000000\" } } }, \"sku\": \"3MKHN58N7RDDVGKJ
\", \"effectiveDate\": \"2018-08-01T00:00:00Z\", \"offerTermCode\": \"JRTCKXETXF
\", \"termAttributes\": { } } } }, \"version\": \"20180808005701\", \"publicationDate\":
\\\"2018-08-08T00:57:01Z\\\" }"
  ]
}

```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetProducts](#) 中的。

## AWS Private CA 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS Private CA。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-certificate-authority-audit-report**

以下代码示例演示了如何使用 `create-certificate-authority-audit-report`。

AWS CLI

创建证书颁发机构审计报告

以下 `create-certificate-authority-audit-report` 命令为由 ARN 标识的私有 CA 创建审计报告。

```
aws acm-pca create-certificate-authority-audit-report --certificate-authority-arn arn:aws:acm-pca:us-east-1:accountid:certificate-authority/12345678-1234-1234-1234-123456789012 --s3-bucket-name your-bucket-name --audit-report-response-format JSON
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateCertificateAuthorityAuditReport](#) 中的。

### **create-certificate-authority**

以下代码示例演示了如何使用 `create-certificate-authority`。

AWS CLI

创建私有证书颁发机构

以下 `create-certificate-authority` 命令在您的 AWS 账户中创建私有证书颁发机构。

```
aws acm-pca create-certificate-authority --certificate-authority-configuration
file://C:\ca_config.txt --revocation-configuration file://C:\revoke_config.txt --
certificate-authority-type "SUBORDINATE" --idempotency-token 98256344
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCertificateAuthority](#)中的。

## delete-certificate-authority

以下代码示例演示了如何使用 delete-certificate-authority。

### AWS CLI

删除私有证书颁发机构

以下delete-certificate-authority命令删除由 ARN 标识的证书颁发机构。

```
aws acm-pca delete-certificate-authority --certificate-
authority-arn arn:aws:acm-pca:us-west-2:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCertificateAuthority](#)中的。

## describe-certificate-authority-audit-report

以下代码示例演示了如何使用 describe-certificate-authority-audit-report。

### AWS CLI

描述证书颁发机构的审计报告

以下describe-certificate-authority-audit-report命令列出了有关由 ARN 标识的 CA 的指定审计报告的信息。

```
aws acm-pca describe-certificate-authority-audit-report --certificate-
authority-arn arn:aws:acm-pca:us-west-2:123456789012:certificate-
authority/99999999-8888-7777-6666-555555555555 --audit-report-id
11111111-2222-3333-4444-555555555555
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCertificateAuthorityAuditReport](#)中的。



## describe-certificate-authority

以下代码示例演示了如何使用 describe-certificate-authority。

### AWS CLI

描述私有证书颁发机构

以下describe-certificate-authority命令列出了由 ARN 标识的私有 CA 的相关信息。

```
aws acm-pca describe-certificate-authority --certificate-
authority-arn arn:aws:acm-pca:us-west-2:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCertificateAuthority](#)中的。

## get-certificate-authority-certificate

以下代码示例演示了如何使用 get-certificate-authority-certificate。

### AWS CLI

检索证书颁发机构 (CA) 证书

以下get-certificate-authority-certificate命令检索 ARN 指定的私有 CA 的证书和证书链。

```
aws acm-pca get-certificate-authority-certificate --certificate-
authority-arn arn:aws:acm-pca:us-west-2:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012 --output text
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCertificateAuthorityCertificate](#)中的。

## get-certificate-authority-csr

以下代码示例演示了如何使用 get-certificate-authority-csr。

### AWS CLI

检索证书颁发机构的证书签名请求

以下 `get-certificate-authority-csr` 命令检索 ARN 指定的私有 CA 的 CSR。

```
aws acm-pca get-certificate-authority-csr --certificate-
authority-arn arn:aws:acm-pca:us-west-2:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012 --output text
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetCertificateAuthorityCsr](#) 中的。

## get-certificate

以下代码示例演示了如何使用 `get-certificate`。

### AWS CLI

检索已颁发的证书

以下 `get-certificate` 示例从指定的私有 CA 检索证书。

```
aws acm-pca get-certificate \
  --certificate-authority-arn arn:aws:acm-pca:us-west-2:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012 \
  --certificate-arn arn:aws:acm-pca:us-west-2:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012/
certificate/6707447683a9b7f4055627ffd55cebcc \
  --output text
```

输出：

```
-----BEGIN CERTIFICATE-----
MIIEDzCCAvegAwIBAgIRAJuJ8f6ZVYL7gG/rS3qvrZMwDQYJKoZIhvcNAQELBQAw
cTElMAkGA1UEBhMCVVMxEzARBgNVBAGMCl dhc2hpbmd0b24xEDA0BgNVBACMB1N1
...certificate body truncated for brevity...
tKCSglgZZrd4FdLw1EkGm+UVXnodwMtJEQyy3oTfZjURPIyyaqskTu/KSS7YDjK0
KQNy73D6Ltmd0EbAyq10XiDxqY41lvKHJ1eZrPaBmYNABxU=
-----END CERTIFICATE----- -----BEGIN CERTIFICATE-----
MIIDrzCCApegAwIBAgIRA0skdzLvcj1eShkoyEE693AwDQYJKoZIhvcNAQELBQAw
cTElMAkGA1UEBhMCVVMxEzARBgNVBAGMCl dhc2hpbmd0b24xEDA0BgNVBACMB1N1
...certificate body truncated for brevity...
kdRGB6P2hpxstDOUIwAoCbhoaWwfA4ybJznf+j0QhAziN1RdKQRR8n0DwPkt7H9w
dJ5nxsTk/fniJz86Ddtp6n8s82wYdkN3cVffeK72A9aTCOU=
-----END CERTIFICATE-----
```

输出的第一部分是证书本身。第二部分是链接到根 CA 证书的证书链。请注意，使用该 `--output text` 选项时，会在两个证书片段之间插入一个 TAB 字符（这是缩进文本的原因）。如果您打算使用此输出并使用其他工具解析证书，则可能需要删除该 TAB 字符，以便正确处理该字符。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetCertificate](#) 中的。

## **import-certificate-authority-certificate**

以下代码示例演示了如何使用 `import-certificate-authority-certificate`。

### AWS CLI

将您的证书颁发机构证书导入 ACM PCA

以下 `import-certificate-authority-certificate` 命令将 ARN 指定的 CA 的已签名私有 CA 证书导入 ACM PCA。

```
aws acm-pca import-certificate-authority-certificate --certificate-
authority-arn arn:aws:acm-pca:us-west-2:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012 --certificate file://C:\ca_cert.pem
--certificate-chain file://C:\ca_cert_chain.pem
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ImportCertificateAuthorityCertificate](#) 中的。

## **issue-certificate**

以下代码示例演示了如何使用 `issue-certificate`。

### AWS CLI

颁发私有证书

以下 `issue-certificate` 命令使用 ARN 指定的私有 CA 来颁发私有证书。

```
aws acm-pca issue-certificate --certificate-authority-arn arn:aws:acm-pca:us-
west-2:123456789012:certificate-authority/12345678-1234-1234-1234-123456789012
--csr file://C:\cert_1.csr --signing-algorithm "SHA256WITHRSA" --validity
Value=365,Type="DAYS" --idempotency-token 1234
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [IssueCertificate](#) 中的。

## list-certificate-authorities

以下代码示例演示了如何使用 `list-certificate-authorities`。

### AWS CLI

列出您的私有证书颁发机构

以下`list-certificate-authorities`命令列出了有关您账户中所有私有 CA 的信息。

```
aws acm-pca list-certificate-authorities --max-results 10
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCertificateAuthorities](#)中的。

## list-tags

以下代码示例演示了如何使用 `list-tags`。

### AWS CLI

列出您的证书颁发机构的标签

以下`list-tags`命令列出了与 ARN 指定的私有 CA 关联的标签。

```
aws acm-pca list-tags --certificate-authority-arn arn:aws:acm-pca:us-west-2:123456789012:certificate-authority/12345678-1234-1234-1234-123456789012 --max-results 10
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTags](#)中的。

## revoke-certificate

以下代码示例演示了如何使用 `revoke-certificate`。

### AWS CLI

吊销私有证书

以下`revoke-certificate`命令吊销由 ARN 标识的 CA 的私有证书。

```
aws acm-pca revoke-certificate --certificate-authority-arn arn:aws:acm-pca:us-west-2:1234567890:certificate-authority/12345678-1234-1234-1234-123456789012 --
```

```
certificate-serial 67:07:44:76:83:a9:b7:f4:05:56:27:ff:d5:5c:eb:cc --revocation-  
reason "KEY_COMPROMISE"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RevokeCertificate](#)中的。

## tag-certificate-authority

以下代码示例演示了如何使用 tag-certificate-authority。

### AWS CLI

将标签附加到私有证书颁发机构

以下tag-certificate-authority命令将一个或多个标签附加到您的私有 CA。

```
aws acm-pca tag-certificate-authority --certificate-authority-  
arn arn:aws:acm-pca:us-west-2:123456789012:certificate-  
authority/12345678-1234-1234-1234-123456789012 --tags Key=Admin,Value=Alice
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagCertificateAuthority](#)中的。

## untag-certificate-authority

以下代码示例演示了如何使用 untag-certificate-authority。

### AWS CLI

从您的私有证书颁发机构删除一个或多个标签

以下untag-certificate-authority命令从由 ARN 标识的私有 CA 中删除标签。

```
aws acm-pca untag-certificate-authority --certificate-authority-  
arn arn:aws:acm-pca:us-west-2:123456789012:certificate-  
authority/12345678-1234-1234-1234-123456789012 --tags Key=Purpose,Value=Website
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagCertificateAuthority](#)中的。

## update-certificate-authority

以下代码示例演示了如何使用 update-certificate-authority。

## AWS CLI

### 更新您的私有证书颁发机构的配置

以下update-certificate-authority命令更新由 ARN 标识的私有 CA 的状态和配置。

```
aws acm-pca update-certificate-authority --certificate-
authority-arn arn:aws:acm-pca:us-west-2:123456789012:certificate-
authority/12345678-1234-1234-1234-1232456789012 --revocation-configuration file://C:
\revoke_config.txt --status "DISABLED"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateCertificateAuthority](#)中的。

## AWS Proton 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Proton。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **cancel-service-instance-deployment**

以下代码示例演示了如何使用 cancel-service-instance-deployment。

### AWS CLI

#### 取消服务实例部署

以下cancel-service-instance-deployment示例取消了服务实例部署。

```
aws proton cancel-service-instance-deployment \
```

```
--service-instance-name "instance-one" \
--service-name "simple-svc"
```

输出：

```
{
  "serviceInstance": {
    "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
    "createdAt": "2021-04-02T21:29:59.962000+00:00",
    "deploymentStatus": "CANCELLING",
    "environmentName": "simple-env",
    "lastDeploymentAttemptedAt": "2021-04-02T21:45:15.406000+00:00",
    "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
    "name": "instance-one",
    "serviceName": "simple-svc",
    "spec": "proton: ServiceSpec\npipeline:\n
my_sample_pipeline_optional_input: abc\n my_sample_pipeline_required_input:
'123'\ninstances:\n- name: my-instance\n environment: MySimpleEnv
\n spec:\n my_sample_service_instance_optional_input: def\n
my_sample_service_instance_required_input: '456'\n- name: my-other-instance\n
environment: MySimpleEnv\n spec:\n my_sample_service_instance_required_input:
'789'\n",
    "templateMajorVersion": "1",
    "templateMinorVersion": "1",
    "templateName": "svc-simple"
  }
}
```

有关更多信息，请参阅《Pro AWS ton 管理员指南》中的“[更新服务实例](#)”或《Proton 用户指南》中的“[更新服务实例](#)”。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelServiceInstanceDeployment](#)中的。

## cancel-service-pipeline-deployment

以下代码示例演示了如何使用 cancel-service-pipeline-deployment。

### AWS CLI

#### 取消服务管道部署

以下cancel-service-pipeline-deployment示例取消了服务管道部署。

```
aws proton cancel-service-pipeline-deployment \  
  --service-name "simple-svc"
```

输出：

```
{  
  "pipeline": {  
    "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/pipeline",  
    "createdAt": "2021-04-02T21:29:59.962000+00:00",  
    "deploymentStatus": "CANCELLING",  
    "lastDeploymentAttemptedAt": "2021-04-02T22:02:45.095000+00:00",  
    "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",  
    "templateMajorVersion": "1",  
    "templateMinorVersion": "1",  
    "templateName": "svc-simple"  
  }  
}
```

有关更多信息，请参阅《Proton 管理员指南》中的 [“更新服务管道”](#) 或《Proton 用户指南》中的 [“更新服务管道”](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CancelServicePipelineDeployment](#) 中的。

## create-service

以下代码示例演示了如何使用 create-service。

### AWS CLI

#### 创建服务

以下 create-service 示例创建了一个带有服务管道的服务。

```
aws proton create-service \  
  --name "MySimpleService" \  
  --template-name "fargate-service" \  
  --template-major-version "1" \  
  --branch-name "mainline" \  
  --repository-connection-arn "arn:aws:codestar-connections:region-id:account-  
id:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" \  
  --repository-id "myorg/myapp" \  
  --spec file://spec.yaml
```



spec.yaml 的内容：

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_required_input: "hello"
  my_sample_pipeline_optional_input: "bye"

instances:
  - name: "acme-network-dev"
    environment: "ENV_NAME"
    spec:
      my_sample_service_instance_required_input: "hi"
      my_sample_service_instance_optional_input: "ho"
```

输出：

```
{
  "service": {
    "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
    "createdAt": "2020-11-18T19:50:27.460000+00:00",
    "lastModifiedAt": "2020-11-18T19:50:27.460000+00:00",
    "name": "MySimpleService",
    "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "repositoryId": "myorg/myapp",
    "status": "CREATE_IN_PROGRESS",
    "templateName": "fargate-service"
  }
}
```

有关更多信息，请参阅《AWS Proton 管理员指南》中的创建[服务](#)和《Proton 用户指南》中的[创建服务](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateService](#)中的。

## delete-service

以下代码示例演示了如何使用 delete-service。

### AWS CLI

#### 删除服务

以下delete-service示例删除服务。

```
aws proton delete-service \  
  --name "simple-svc"
```

输出：

```
{  
  "service": {  
    "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",  
    "branchName": "mainline",  
    "createdAt": "2020-11-28T22:40:50.512000+00:00",  
    "description": "Edit by updating description",  
    "lastModifiedAt": "2020-11-29T00:30:39.248000+00:00",  
    "name": "simple-svc",  
    "repositoryConnectionArn": "arn:aws:codestar-connections:region-  
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "repositoryId": "myorg/myapp",  
    "status": "DELETE_IN_PROGRESS",  
    "templateName": "fargate-service"  
  }  
}
```

有关更多信息，请参阅《AWS Proton 管理员指南》中的[删除服务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteService](#)中的。

## get-service-instance

以下代码示例演示了如何使用 get-service-instance。

### AWS CLI

获取服务实例详细信息

以下get-service-instance示例获取服务实例的详细数据。

```
aws proton get-service-instance \  
  --name "instance-one" \  
  --service-name "simple-svc"
```

输出：

```
{
  "serviceInstance": {
    "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
    "createdAt": "2020-11-28T22:40:50.512000+00:00",
    "deploymentStatus": "SUCCEEDED",
    "environmentName": "simple-env",
    "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
    "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
    "name": "instance-one",
    "serviceName": "simple-svc",
    "spec": "proton: ServiceSpec\npipeline:\n
my_sample_pipeline_optional_input: hello world\n
my_sample_pipeline_required_input: pipeline up\ninstances:\n- name: instance-one\n
environment: my-simple-env\n spec:\n   my_sample_service_instance_optional_input:
Ola\n   my_sample_service_instance_required_input: Ciao\n",
    "templateMajorVersion": "1",
    "templateMinorVersion": "0",
    "templateName": "svc-simple"
  }
}
```

有关更多信息，请参阅《Proton 管理员指南》中的[查看服务数据](#)或《Proton 用户指南》中的[查看服务数据](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetServiceInstance](#)中的。

## get-service

以下代码示例演示了如何使用 get-service。

### AWS CLI

#### 获取服务详情

以下 get-service 示例获取服务的详细数据。

```
aws proton get-service \
  --name "simple-svc"
```

输出：

```

{
  "service": {
    "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
    "branchName": "mainline",
    "createdAt": "2020-11-28T22:40:50.512000+00:00",
    "lastModifiedAt": "2020-11-28T22:44:51.207000+00:00",
    "name": "simple-svc",
    "pipeline": {
      "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
pipeline/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "createdAt": "2020-11-28T22:40:50.512000+00:00",
      "deploymentStatus": "SUCCEEDED",
      "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
      "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
      "spec": "proton: ServiceSpec\npipeline:\n
my_sample_pipeline_required_input: hello\n my_sample_pipeline_optional_input:
bye\ninstances:\n- name: instance-svc-simple\n environment: my-simple-
env\n spec:\n  my_sample_service_instance_required_input: hi\n
my_sample_service_instance_optional_input: ho\n",
      "templateMajorVersion": "1",
      "templateMinorVersion": "1",
      "templateName": "svc-simple"
    },
    "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "repositoryId": "myorg/myapp",
    "spec": "proton: ServiceSpec\npipeline:\n
my_sample_pipeline_required_input: hello\n my_sample_pipeline_optional_input:
bye\ninstances:\n- name: instance-svc-simple\n environment: my-simple-
env\n spec:\n  my_sample_service_instance_required_input: hi\n
my_sample_service_instance_optional_input: ho\n",
    "status": "ACTIVE",
    "templateName": "svc-simple"
  }
}

```

有关更多信息，请参阅《Pro AWS ton 管理员指南》中的[查看服务数据](#)或《Proton 用户指南》中的[查看服务数据](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetService](#)中的。

## list-service-instances

以下代码示例演示了如何使用 `list-service-instances`。

### AWS CLI

示例 1：列出所有服务实例

以下 `list-service-instances` 示例列出了服务实例。

```
aws proton list-service-instances
```

输出：

```
{
  "serviceInstances": [
    {
      "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-instance/instance-one",
      "createdAt": "2020-11-28T22:40:50.512000+00:00",
      "deploymentStatus": "SUCCEEDED",
      "environmentArn": "arn:aws:proton:region-id:123456789012:environment/simple-env",
      "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
      "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
      "name": "instance-one",
      "serviceName": "simple-svc",
      "templateMajorVersion": "1",
      "templateMinorVersion": "0",
      "templateName": "fargate-service"
    }
  ]
}
```

有关更多信息，请参阅《Proton 管理员指南》中的[查看服务实例数据](#)或《Proton 用户指南》中的[查看服务实例数据](#)。AWS

示例 2：列出指定的服务实例

以下 `get-service-instance` 示例获取了一个服务实例。

```
aws proton get-service-instance \
```

```
--name "instance-one" \  
--service-name "simple-svc"
```

输出：

```
{  
  "serviceInstance": {  
    "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-  
instance/instance-one",  
    "createdAt": "2020-11-28T22:40:50.512000+00:00",  
    "deploymentStatus": "SUCCEEDED",  
    "environmentName": "simple-env",  
    "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",  
    "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",  
    "name": "instance-one",  
    "serviceName": "simple-svc",  
    "spec": "proton: ServiceSpec\npipeline:\nmy_sample_pipeline_optional_input: hello world\nmy_sample_pipeline_required_input: pipeline up\ninstances:\n- name: instance-one\nenvironment: my-simple-env\n spec:\n  my_sample_service_instance_optional_input:  
0la\n  my_sample_service_instance_required_input: Ciao\n",  
    "templateMajorVersion": "1",  
    "templateMinorVersion": "0",  
    "templateName": "svc-simple"  
  }  
}
```

有关更多信息，请参阅《Pro AWS ton 管理员指南》中的[查看服务实例数据](#)或《Proton 用户指南》中的[查看服务实例数据](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListServiceInstances](#)中的。

## update-service-instance

以下代码示例演示了如何使用 update-service-instance。

### AWS CLI

将服务实例更新到新的次要版本

以下update-service-instance示例将服务实例更新为其服务模板的新次要版本，该模板添加了一个名为“my-other-instance”的新实例，其中包含新的必填输入。

```
aws proton update-service-instance \  
  --service-name "simple-svc" \  
  --spec "file:///service-spec.yaml" \  
  --template-major-version "1" \  
  --template-minor-version "1" \  
  --deployment-type "MINOR_VERSION" \  
  --name "instance-one"
```

service-spec.yaml 的内容：

```
proton: ServiceSpec  
pipeline:  
  my_sample_pipeline_optional_input: "abc"  
  my_sample_pipeline_required_input: "123"  
instances:  
  - name: "instance-one"  
    environment: "simple-env"  
    spec:  
      my_sample_service_instance_optional_input: "def"  
      my_sample_service_instance_required_input: "456"  
  - name: "my-other-instance"  
    environment: "simple-env"  
    spec:  
      my_sample_service_instance_required_input: "789"
```

输出：

```
{  
  "serviceInstance": {  
    "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-  
instance/instance-one",  
    "createdAt": "2021-04-02T21:29:59.962000+00:00",  
    "deploymentStatus": "IN_PROGRESS",  
    "environmentName": "arn:aws:proton:region-id:123456789012:environment/  
simple-env",  
    "lastDeploymentAttemptedAt": "2021-04-02T21:38:00.823000+00:00",  
    "lastDeploymentSucceededAt": "2021-04-02T21:29:59.962000+00:00",  
    "name": "instance-one",  
    "serviceName": "simple-svc",  
    "templateMajorVersion": "1",  
    "templateMinorVersion": "0",  
    "templateName": "svc-simple"
```

```
}
}
```

有关更多信息，请参阅《Pro AWS ton 管理员指南》中的“[更新服务实例](#)”或《Proton 用户指南》中的“[更新服务实例](#)”。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateServiceInstance](#)中的。

## update-service-pipeline

以下代码示例演示了如何使用 update-service-pipeline。

### AWS CLI

#### 更新服务管道

以下update-service-pipeline示例将服务管道更新为其服务模板的新次要版本。

```
aws proton update-service-pipeline \
  --service-name "simple-svc" \
  --spec "file://service-spec.yaml" \
  --template-major-version "1" \
  --template-minor-version "1" \
  --deployment-type "MINOR_VERSION"
```

输出：

```
{
  "pipeline": {
    "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/pipeline/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "createdAt": "2021-04-02T21:29:59.962000+00:00",
    "deploymentStatus": "IN_PROGRESS",
    "lastDeploymentAttemptedAt": "2021-04-02T21:39:28.991000+00:00",
    "lastDeploymentSucceededAt": "2021-04-02T21:29:59.962000+00:00",
    "spec": "proton: ServiceSpec\n\npipeline:\n
my_sample_pipeline_optional_input: \"abc\"\n my_sample_pipeline_required_input:
\"123\"\n\ninstances:\n - name: \"my-instance\"\n   environment: \"MySimpleEnv
\"\n   spec:\n     my_sample_service_instance_optional_input: \"def
\"\n     my_sample_service_instance_required_input: \"456\"\n - name:
\"my-other-instance\"\n   environment: \"MySimpleEnv\"\n   spec:\n
my_sample_service_instance_required_input: \"789\"\n\n",
```



```
    "templateMajorVersion": "1",
    "templateMinorVersion": "0",
    "templateName": "svc-simple"
  }
}
```

有关更多信息，请参阅《Proton 管理员指南》中的 [“更新服务管道”](#) 或《Proton 用户指南》中的 [“更新服务管道”](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateServicePipeline](#) 中的。

## update-service

以下代码示例演示了如何使用 update-service。

### AWS CLI

#### 更新服务

以下 update-service 示例编辑服务描述。

```
aws proton update-service \
  --name "MySimpleService" \
  --description "Edit by updating description"
```

输出：

```
{
  "service": {
    "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
    "branchName": "mainline",
    "createdAt": "2021-03-12T22:39:42.318000+00:00",
    "description": "Edit by updating description",
    "lastModifiedAt": "2021-03-12T22:44:21.975000+00:00",
    "name": "MySimpleService",
    "repositoryConnectionArn": "arn:aws:codestar-connections:region-id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "repositoryId": "myorg/myapp",
    "status": "ACTIVE",
    "templateName": "fargate-service"
  }
}
```

有关更多信息，请参阅《AWS Proton 管理员指南》中的[编辑服务](#)或《Proton 用户指南》中的[编辑服务](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateService](#)中的。

## 使用的 QLDB 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 QLDB 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **cancel-journal-kinesis-stream**

以下代码示例演示了如何使用 `cancel-journal-kinesis-stream`。

#### AWS CLI

##### 取消日记流

以下 `cancel-journal-kinesis-stream` 示例从账本中取消指定的日记账流。

```
aws qlldb cancel-journal-kinesis-stream \  
  --ledger-name myExampleLedger \  
  --stream-id 7ISckqwe4y25YyHLzYUFAf
```

##### 输出：

```
{
```

```
"StreamId": "7ISckqwe4y25YyHLzYUFAf"
}
```

有关更多信息，请参阅亚马逊 QLDB 开发者指南中的[从亚马逊 QLDB 流式传输日记数据](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CancelJournalKinesisStream](#)中的。

## create-ledger

以下代码示例演示了如何使用 create-ledger。

### AWS CLI

示例 1：创建具有默认属性的分类账

以下 create-ledger 示例使用名称 myExampleLedger 和权限模式 STANDARD 创建分类账。未指定删除保护和 AWS KMS 密钥的可选参数，因此它们分别默认为 true 和 AWS 拥有的 KMS 密钥。

```
aws qlldb create-ledger \  
  --name myExampleLedger \  
  --permissions-mode STANDARD
```

输出：

```
{  
  "State": "CREATING",  
  "Arn": "arn:aws:qlldb:us-west-2:123456789012:ledger/myExampleLedger",  
  "DeletionProtection": true,  
  "CreationDateTime": 1568839243.951,  
  "Name": "myExampleLedger",  
  "PermissionsMode": "STANDARD"  
}
```

示例 2：使用客户托管 KMS 密钥和指定标签创建禁用了删除保护的分类账

以下 create-ledger 示例使用名称 myExampleLedger2 和权限模式 STANDARD 创建分类账。删除保护功能已禁用，指定的客户托管 KMS 密钥用于静态加密，并将指定的标签附加到资源。

```
aws qlldb create-ledger \  
  --name myExampleLedger2
```

```
--name myExampleLedger2 \  
--permissions-mode STANDARD \  
--no-deletion-protection \  
--kms-key arn:aws:kms:us-west-2:123456789012:key/a1b2c3d4-5678-90ab-cdef-  
EXAMPLE111111 \  
--tags IsTest=true,Domain=Test
```

输出：

```
{  
  "Arn": "arn:aws:qldb:us-west-2:123456789012:ledger/myExampleLedger2",  
  "DeletionProtection": false,  
  "CreationDateTime": 1568839543.557,  
  "State": "CREATING",  
  "Name": "myExampleLedger2",  
  "PermissionsMode": "STANDARD",  
  "KmsKeyArn": "arn:aws:kms:us-west-2:123456789012:key/a1b2c3d4-5678-90ab-cdef-  
EXAMPLE111111"  
}
```

有关更多信息，请参阅《Amazon QLDB 开发人员指南》中的 [Amazon QLDB 分类账的基本操作](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLedger](#)中的。

## delete-ledger

以下代码示例演示了如何使用 delete-ledger。

AWS CLI

删除账本

以下delete-ledger示例删除了指定的账本。

```
aws qldb delete-ledger \  
  --name myExampleLedger
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon QLDB 开发人员指南》中的 [Amazon QLDB 分类账的基本操作](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLedger](#)中的。

## describe-journal-kinesis-stream

以下代码示例演示了如何使用 `describe-journal-kinesis-stream`。

### AWS CLI

#### 描述日记流

以下 `describe-journal-kinesis-stream` 示例显示了账本中指定日记账流的详细信息。

```
aws qlldb describe-journal-kinesis-stream \  
  --ledger-name myExampleLedger \  
  --stream-id 7ISCKqwe4y25YyHLzYUFaf
```

#### 输出：

```
{  
  "Stream": {  
    "LedgerName": "myExampleLedger",  
    "CreationTime": 1591221984.677,  
    "InclusiveStartTime": 1590710400.0,  
    "ExclusiveEndTime": 1590796799.0,  
    "RoleArn": "arn:aws:iam::123456789012:role/my-kinesis-stream-role",  
    "StreamId": "7ISCKqwe4y25YyHLzYUFaf",  
    "Arn": "arn:aws:qlldb:us-east-1:123456789012:stream/  
myExampleLedger/7ISCKqwe4y25YyHLzYUFaf",  
    "Status": "ACTIVE",  
    "KinesisConfiguration": {  
      "StreamArn": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-for-  
qlldb",  
      "AggregationEnabled": true  
    },  
    "StreamName": "myExampleLedger-stream"  
  }  
}
```

有关更多信息，请参阅亚马逊 QLDB 开发者指南中的[从亚马逊 QLDB 流式传输日记数据](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeJournalKinesisStream](#) 中的。

## describe-journal-s3-export

以下代码示例演示了如何使用 `describe-journal-s3-export`。

## AWS CLI

### 描述日志导出任务

以下 `describe-journal-s3-export` 示例显示了账本中指定导出任务的详细信息。

```
aws qlldb describe-journal-s3-export \  
  --name myExampleLedger \  
  --export-id ADR20NPKN5LINYGb4dp7yZ
```

输出：

```
{  
  "ExportDescription": {  
    "S3ExportConfiguration": {  
      "Bucket": "awsExampleBucket",  
      "Prefix": "ledgerexport1/",  
      "EncryptionConfiguration": {  
        "ObjectEncryptionType": "SSE_S3"  
      }  
    },  
    "RoleArn": "arn:aws:iam::123456789012:role/my-s3-export-role",  
    "Status": "COMPLETED",  
    "ExportCreationTime": 1568847801.418,  
    "InclusiveStartTime": 1568764800.0,  
    "ExclusiveEndTime": 1568847599.0,  
    "LedgerName": "myExampleLedger",  
    "ExportId": "ADR20NPKN5LINYGb4dp7yZ"  
  }  
}
```

有关更多信息，请参阅《[亚马逊 QLDB 开发者指南](#)》中的[在亚马逊 QLDB 中导出日记](#)。

- 有关 API 的详细信息，请参阅《[AWS CLI 命令参考](#)》中的 [DescribeJournalS3Export](#)。

## describe-ledger

以下代码示例演示了如何使用 `describe-ledger`。

## AWS CLI

### 描述账本

以下describe-ledger示例显示了指定账本的详细信息。

```
aws qlldb describe-ledger \  
  --name myExampleLedger
```

输出：

```
{  
  "CreationDateTime": 1568839243.951,  
  "Arn": "arn:aws:qlldb:us-west-2:123456789012:ledger/myExampleLedger",  
  "State": "ACTIVE",  
  "Name": "myExampleLedger",  
  "DeletionProtection": true,  
  "PermissionsMode": "STANDARD",  
  "EncryptionDescription": {  
    "KmsKeyArn": "arn:aws:kms:us-west-2:123456789012:key/a1b2c3d4-5678-90ab-  
cdef-EXAMPLE11111",  
    "EncryptionStatus": "ENABLED"  
  }  
}
```

有关更多信息，请参阅《Amazon QLDB 开发人员指南》中的 [Amazon QLDB 分类账的基本操作](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLedger](#)中的。

## export-journal-to-s3

以下代码示例演示了如何使用 export-journal-to-s3。

### AWS CLI

将日记块导出到 S3

以下export-journal-to-s3示例从名为的账本中为指定日期和时间范围内的日记账块创建导出任务myExampleLedger。导出任务会将数据块写入指定的 Amazon S3 存储桶。

```
aws qlldb export-journal-to-s3 \  
  --name myExampleLedger \  
  --inclusive-start-time 2019-09-18T00:00:00Z \  
  --exclusive-end-time 2019-09-18T22:59:59Z \  
  --role-arn arn:aws:iam::123456789012:role/my-s3-export-role \  
  --s3-bucket my-s3-bucket
```

```
--s3-export-configuration file://my-s3-export-config.json
```

my-s3-export-config.json 的内容：

```
{
  "Bucket": "awsExampleBucket",
  "Prefix": "ledgerexport1/",
  "EncryptionConfiguration": {
    "ObjectEncryptionType": "SSE_S3"
  }
}
```

输出：

```
{
  "ExportId": "ADR20NPKN5LINYGb4dp7yZ"
}
```

有关更多信息，请参阅《[亚马逊 QLDB 开发者指南](#)》中的[在亚马逊 QLDB 中导出日记](#)。

- 有关 API 的详细信息，请参阅《[AWS CLI 命令参考](#)》中的 [ExportJournalToS3](#)。

## get-block

以下代码示例演示了如何使用 get-block。

### AWS CLI

示例 1：使用输入文件获取日记块和校样以供验证

以下 get-block 示例请求来自指定账本的区块数据对象和证明。请求的是指定的摘要提示地址和区块地址。

```
aws qlldb get-block \
  --name vehicle-registration \
  --block-address file://myblockaddress.json \
  --digest-tip-address file://mydigesttipaddress.json
```

myblockaddress.json 的内容：

```
{
```



```
"IonText": "{strandId:\\"KmA3ZZca7vAIiJAK9S5Iw1\\",sequenceNo:100}"
}
```

mydigesttipaddress.json 的内容 :

```
{
  "IonText": "{strandId:\\"KmA3ZZca7vAIiJAK9S5Iw1\\",sequenceNo:123}"
}
```

输出 :

```
{
  "Block": {
    "IonText": "{blockAddress:{strandId:\\"KmA3ZZca7vAIiJAK9S5Iw1\\",sequenceNo:100},transactionId:\\"FnQeJBAicTX0Ah32ZnVtSX\\",blockTimestamp:2019-09-16T19:37:05.360Z,blockHash:{NoChM92yKRuJAb/jeLd1VnYn4DHiWI071ACfic9uHc=}},entriesHash:{105L0siKV14SdbuaYnH7uwXzUvqzIwUiRLXGbTyj/nY=}},previousBlockHash:{7kewBXhpdbC1cZKxhVmpoMHPUG0JtWQD0iY2LPfZkYA=}},entriesHashList:[{eRSwnmAM7WWANWdD5iG0yK+T4tDXyzUq6HZ/0fgLHos=}},{mHVex/yjHAWjFPpwhBuH2GKXmKJjK2FBa9faqoUVNtg=}},{y5cCB7p0AIUfsVQ1j0TqtE97b4b4oo1R0vnYyE5wWM=}},{TvTXygML1bMe6NvEZtGkX+KR+W/EJl4qD1mmV77KZQg=}}],transactionInfo:{statements:[{statement:\\"FROM VehicleRegistration AS r \\nWHERE r.VIN = '1N4AL11D75C109151'\\nINSERT INTO r.Owners.SecondaryOwners\\n  VALUE { 'PersonId' : 'CMvdR77XP8zAg1mmFDGTvt' }\\",startTime:2019-09-16T19:37:05.302Z,statementDigest:{jcgPX2vs0J0waum4qmDYtn1pCAT9xKNIZa+2k4R+mxA=}}}],documents:{JUJgkIcNbhS2goq8RqLuZ4:{tableName:\\"VehicleRegistration\\",tableId:\\"BFJKdXgzt9oF4wjMboxy4G\\",statements:[0]}}],revisions:[{blockAddress:{strandId:\\"KmA3ZZca7vAIiJAK9S5Iw1\\",sequenceNo:100},hash:{mHVex/yjHAWjFPpwhBuH2GKXmKJjK2FBa9faqoUVNtg=}},data:{VIN:\\"1N4AL11D75C109151\\",LicensePlateNumber:\\"LEWISR261LL\\",State:\\"WA\\",PendingPenaltyTicketAmount:90.25,ValidFromDate:2017-08-21,ValidToDate:2020-05-11,Owners:{PrimaryOwner:{PersonId:\\"BFJKdXhnLRT27sXBnojNGW\\"},SecondaryOwners:[{PersonId:\\"CMvdR77XP8zAg1mmFDGTvt\\"}]},City:\\"Everett\\",metadata:{id:\\"JUJgkIcNbhS2goq8RqLuZ4\\",version:3,txTime:2019-09-16T19:37:05.344Z,txId:\\"FnQeJBAicTX0Ah32ZnVtSX\\"}}}}],
  "Proof": {
    "IonText": "[{13+EXs69K1+rehlqyWLkt+oHDlw4Zi9pCLW/t/mgTPM=}},{{48CXG3ehPqsxCYd34EEa8Fso00RpWMA08010RJkF3Do=}},{{9UnwnKSQT0i3ge1JMVa+tMIqCEDaOPTkwxmyHSn8UPQ=}},{{3nW6Vryghk+7pd6wFcTlufgPM6qXHyTNECb1sCwcDaI=}},{{Irb5fNhBrNEQ1VPhzlnGT/ZQPadSmgfdtMYcwkN0xoI=}},{{+3CwpYG/ytf/
```

```

vq9GidpzSx6JJiLXt1hMQWNnq0y3jfY=}},{{NPx6cRhwsiy5m9UEWS5JTJrZoUd02jB0AA0myZAT
+qE=}}}"
  }
}

```

有关更多信息，请参阅《[亚马逊 QLDB 开发者指南](#)》中的 [Amazon QLDB 中的数据验证](#)。

示例 2：使用速记语法获取日记块和证明以供验证

以下 `get-block` 示例使用速记语法从指定账本中请求区块数据对象和证明。请求的是指定的摘要提示地址和区块地址。

```

aws qlldb get-block \
  --name vehicle-registration \
  --block-address 'IonText="{strandId:\\"KmA3ZZca7vAIiJAK9S5Iw1\\",sequenceNo:100}"' \
  --digest-tip-address 'IonText="{strandId:\\"KmA3ZZca7vAIiJAK9S5Iw1\\",sequenceNo:123}"'

```

输出：

```

{
  "Block": {
    "IonText": "{blockAddress:{strandId:\\"KmA3ZZca7vAIiJAK9S5Iw1\\",sequenceNo:100},transactionId:\\"FnQeJBAicTX0Ah32ZnVtSX\\",blockTimestamp:2019-09-16T19:37:05.360Z,blockHash:{{NoChM92yKRuJAb/jeLd1VnYn4DHiWI071ACfic9uHc=}},entriesHash:{{105L0siKV14SDbuaYnH7uwXzUvqzIwUiRLXGbTyj/nY=}},previousBlockHash:{{7kewBXhpdbClcZKxhVmpoMHPUG0JtWQD0iY2LPfZkYA=}},entriesHashList:{{eRSwnmAM7WWANWdD5iG0yK+T4tDXyzUq6HZ/0fgLHos=}},{{mHVex/yjHAWjFPpwhBuH2GKXmKJjK2FBa9faquUVNtg=}},{{y5cCB7p0AIUfsVQ1j0TqtE97b4b4oo1R0vnYyE5wWM=}},{{TvTXygML1bMe6NvEZtGkX+KR+W/EJl4qD1mmV77KZQg=}}}],transactionInfo:{statements:[{statement:\\"FROM VehicleRegistration AS r \\nWHERE r.VIN = '1N4AL11D75C109151'\\nINSERT INTO r.Owners.SecondaryOwners\\n  VALUE { 'PersonId' : 'CMVdR77XP8zAglmmFDGTvt' }\\n\",startTime:2019-09-16T19:37:05.302Z,statementDigest:{{jcgPX2vs0J0waum4qmDYtn1pCAT9xKNIZa+2k4R+mxA=}}}],documents:[JUJgkIcNbhS2goq8RqLuZ4:{tableName:\\"VehicleRegistration\\",tableId:\\"BFJKdXgz9oF4wjMboxy4G\\",statements:[0]}]},revisions:[{blockAddress:{strandId:\\"KmA3ZZca7vAIiJAK9S5Iw1\\",sequenceNo:100},hash:{{mHVex/yjHAWjFPpwhBuH2GKXmKJjK2FBa9faquUVNtg=}},data:{VIN:\\"1N4AL11D75C109151\\",LicensePlateNumber:\\"LEWISR261LL\\",State:\\"WA\\",PendingPenaltyTicketAmount:90.25,ValidFromDate:2017-08-21,ValidToDate:2020-05-11,Owners:

```

```
{PrimaryOwner:{PersonId:\"BFJKdXhnLRT27sXBnojNGW\"},SecondaryOwners:
[{{PersonId:\"CMVdR77XP8zAgImmFDGTvt\"}},{{PersonId:\"Everett\"}},{{id:
\"JUJgkIcNbhS2goq8RqLuZ4\",version:3,txTime:2019-09-16T19:37:05.344Z,txId:
\"FnQeJBAicTX0Ah32ZnVtSX\"}}]}]
},
  \"Proof\": {
    \"IonText\": \"[[{{13+EXs69K1+rehlqyWLkt+oHDlw4Zi9pCLW/t/mgTPM=}},
{{48CXG3ehPqsxCYd34EEa8Fso00RpWMA08010RJKf3Do=}},{{9UnwnKSQT0i3ge1JMVa
+tMIqCEDaOPTkwxmyHSn8UPQ=}},{{3nW6Vryghk+7pd6wFCtLufgPM6qXHyTNeCb1sCwcDaI=}},
{{Irb5fNhBrNEQ1VPhzlnGT/ZQPadSmgfdtMYcwkN0xoI=}},{{+3CwpYG/ytf/
vq9GidpzSx6JJiLXt1hMQWnNq0y3jfY=}},{{NPx6cRhwsiy5m9UEWS5JTJrZoUd02jB0AA0myZAT
+qE=}}]]\"
  }
}
```

有关更多信息，请参阅《亚马逊 [QLDB 开发者指南](#)》中的 [Amazon QLDB 中的数据验证](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetBlock](#) 中的。

## get-digest

以下代码示例演示了如何使用 get-digest。

### AWS CLI

#### 获取账本摘要

以下 get-digest 示例请求日记账中最新提交的区块中指定账本的摘要。

```
aws qlldb get-digest \
  --name vehicle-registration
```

输出：

```
{
  \"Digest\": \"6m6BMXobbJKpMhahwVthAEsN6awgnHK62Qq5McGP1Gk=\",
  \"DigestTipAddress\": {
    \"IonText\": \"{strandId:\"KmA3ZZca7vAIiJAK9S5Iw1\",sequenceNo:123}\"
  }
}
```

有关更多信息，请参阅《亚马逊 [QLDB 开发者指南](#)》中的 [Amazon QLDB 中的数据验证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDigest](#)中的。

## get-revision

以下代码示例演示了如何使用 get-revision。

### AWS CLI

示例 1：使用输入文件获取文档修订版和校样以供验证

以下get-revision示例请求来自指定账本的修订数据对象和证明。请求提供修订版的指定摘要提示地址、文档 ID 和区块地址。

```
aws qlldb get-revision \
  --name vehicle-registration \
  --block-address file://myblockaddress.json \
  --document-id JUJgkIcNbhS2goq8RqLuZ4 \
  --digest-tip-address file://mydigesttipaddress.json
```

myblockaddress.json 的内容：

```
{
  "IonText": "{strandId:\\"KmA3ZZca7vAIiJAK9S5Iw1\\",sequenceNo:100}"
}
```

mydigesttipaddress.json 的内容：

```
{
  "IonText": "{strandId:\\"KmA3ZZca7vAIiJAK9S5Iw1\\",sequenceNo:123}"
}
```

输出：

```
{
  "Revision": {
    "IonText": "{blockAddress:{strandId:\\"KmA3ZZca7vAIiJAK9S5Iw1\\",sequenceNo:100},hash:{mHVex/yjHAWjFPpwhBuH2GKXmKJjK2FBa9faqoUVNtg=}},data:
    {VIN:\\"1N4AL11D75C109151\\",LicensePlateNumber:\\"LEWISR261LL\\",State:\\"WA\\",PendingPenaltyTicketAmount:90.25,ValidFromDate:2017-08-21,ValidToDate:2020-05-11,Owners:
    {PrimaryOwner:{PersonId:\\"BFJKdXhnLRT27sXBnojNGW\\"},SecondaryOwners:
    [{PersonId:\\"CMVdR77XP8zAg1mmFDGTvt\\"}]},City:\\"Everett\\"},metadata:{id:
```

```
\ "JUJgkIcNbhS2goq8RqLuZ4\", version:3, txTime:2019-09-16T19:37:05.344Z, txId:
\FnQeJBAicTX0Ah32ZnVtSX\"}]"
  },
  "Proof": {
    "IonText": "[{{eRSwnmAM7WWANWDD5iG0yK+T4tDXyzUq6HZ/0fgLHos=}}, {{VV1rdaNuf
+yJZVG1msM6gr2T52QvB08Lg+KgpjcnWAU=}},
{{7kewBXhpdBc1cZKxhVmpoMhpUG0JtwQD0iY2LPfZkYA=}}, {{13+EXs69K1+reh1qyWLkt
+oHD1w4Zi9pCLW/t/mgTPM=}}, {{48CXG3ehPqsxCYd34EEa8Fso00RpWAA08010RJkf3Do=}},
{{9UnwnKSQT0i3ge1JMVa+tMIqCEDaOPTkwxmyHSn8UPQ=}}, {{3nW6Vryghk
+7pd6wFCtLufgPM6qXHyTNeCb1sCwcDaI=}}, {{Irb5fNhBrNEQ1VPhzlnGT/
ZQPadSmgfdtMYcwkN0xoI=}}, {{+3CwpYG/ytf/vq9GidpzSx6JJiLXt1hMQWNnq0y3jfy=}},
{{NPx6cRhwsiy5m9UEWS5JTJrZoUd02jB0AA0myZAT+qE=}}]"
  }
}
```

有关更多信息，请参阅《[亚马逊 QLDB 开发者指南](#)》中的 [Amazon QLDB 中的数据验证](#)。

示例 2：使用速记语法获取文档修订版和验证证明

以下 `get-revision` 示例使用速记语法请求来自指定账本的修订数据对象和证明。请求提供修订版的指定摘要提示地址、文档 ID 和区块地址。

```
aws qlldb get-revision \
  --name vehicle-registration \
  --block-address 'IonText="{strandId:\ "KmA3ZZca7vAIiJAK9S5Iw1\", sequenceNo:100}"' \
  --document-id JUJgkIcNbhS2goq8RqLuZ4 \
  --digest-tip-address 'IonText="{strandId:\ "KmA3ZZca7vAIiJAK9S5Iw1
\", sequenceNo:123}"'
```

输出：

```
{
  "Revision": {
    "IonText": "{blockAddress:{strandId:\ "KmA3ZZca7vAIiJAK9S5Iw1
\", sequenceNo:100}, hash:{{mHVex/yjHAWjFPpwhBuH2GKXmKJjK2FBa9faquUVNtg=}}, data:
{VIN:\ "1N4AL11D75C109151\", LicensePlateNumber:\ "LEWISR261LL\", State:\ "WA
\", PendingPenaltyTicketAmount:90.25, ValidFromDate:2017-08-21, ValidToDate:2020-05-11, Owners:
{PrimaryOwner:{PersonId:\ "BFJKdXhnLRT27sXBnojNGW\"}, SecondaryOwners:
[{{PersonId:\ "CMVdR77XP8zAg1mmFDGTvt\"}}]}, City:\ "Everett\"}, metadata:{id:
\"JUJgkIcNbhS2goq8RqLuZ4\", version:3, txTime:2019-09-16T19:37:05.344Z, txId:
\FnQeJBAicTX0Ah32ZnVtSX\"}]"
  },
```

```

    "Proof": {
      "IonText": "[{{eRSwnmAM7WWANWd5iG0yK+T4tDXyzUq6HZ/0fgLHos=}},{{VV1rdaNuf
+yJZVG1msM6gr2T52QvB08Lg+KgpjcnWAU=}},
{{7kewBXhpdBc1cZKxhVmpoMHPUG0JtWQD0iY2LPfZkYA=}},{{13+EXs69K1+rehlqyWLkt
+oHD1w4Zi9pCLW/t/mgTPM=}},{{48CXG3ehPqsxCYd34EEa8Fso00RpWWA08010RJKf3Do=}},
{{9UnwnKSQT0i3ge1JMVa+tMIqCEDa0PTkWxmyHSn8UPQ=}},{{3nW6Vryghk
+7pd6wFCtLufgPM6qXHyTNeCb1sCwcDaI=}},{{Irb5fNhBrNEQ1VPhz1nGT/
ZQPadSmgfdtMYcwkN0xoI=}},{{+3CWpYG/ytf/vq9GidpzSx6JJiLXt1hMQWNnq0y3jfY=}},
{{NPx6cRhwsiy5m9UEWS5JTJrZoUd02jB0AA0myZAT+qE=}}]"
    }
  }
}

```

有关更多信息，请参阅《亚马逊 [QLDB 开发者指南](#)》中的 [Amazon QLDB 中的数据验证](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetRevision](#) 中的。

## list-journal-kinesis-streams-for-ledger

以下代码示例演示了如何使用 `list-journal-kinesis-streams-for-ledger`。

### AWS CLI

列出分类账的日记账流

以下 `list-journal-kinesis-streams-for-ledger` 示例列出了指定账本的日记账流。

```

aws qlldb list-journal-kinesis-streams-for-ledger \
  --ledger-name myExampleLedger

```

输出：

```

{
  "Streams": [
    {
      "LedgerName": "myExampleLedger",
      "CreationTime": 1591221984.677,
      "InclusiveStartTime": 1590710400.0,
      "ExclusiveEndTime": 1590796799.0,
      "RoleArn": "arn:aws:iam::123456789012:role/my-kinesis-stream-role",
      "StreamId": "7ISckqwe4y25YyHLzYUFaf",
      "Arn": "arn:aws:qlldb:us-east-1:123456789012:stream/
myExampleLedger/7ISckqwe4y25YyHLzYUFaf",
    }
  ]
}

```

```
        "Status": "ACTIVE",
        "KinesisConfiguration": {
            "StreamArn": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
for-qlldb",
            "AggregationEnabled": true
        },
        "StreamName": "myExampleLedger-stream"
    }
]
}
```

有关更多信息，请参阅亚马逊 QLDB 开发者指南中的[从亚马逊 QLDB 流式传输日记数据](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListJournalKinesisStreamsForLedger](#)中的。

## list-journal-s3-exports-for-ledger

以下代码示例演示了如何使用 `list-journal-s3-exports-for-ledger`。

### AWS CLI

列出分类帐的日记账导出任务

以下 `list-journal-s3-exports-for-ledger` 示例列出了指定账本的日记账导出任务。

```
aws qlldb list-journal-s3-exports-for-ledger \
  --name myExampleLedger
```

输出：

```
{
  "JournalS3Exports": [
    {
      "LedgerName": "myExampleLedger",
      "ExclusiveEndTime": 1568847599.0,
      "ExportCreationTime": 1568847801.418,
      "S3ExportConfiguration": {
        "Bucket": "awsExampleBucket",
        "Prefix": "ledgerexport1/",
        "EncryptionConfiguration": {
          "ObjectEncryptionType": "SSE_S3"
        }
      }
    }
  ]
}
```

```
    },
    "ExportId": "ADR20NPKN5LINYGb4dp7yZ",
    "RoleArn": "arn:aws:iam::123456789012:role/qlldb-s3-export",
    "InclusiveStartTime": 1568764800.0,
    "Status": "IN_PROGRESS"
  }
]
}
```

有关更多信息，请参阅《[亚马逊 QLDB 开发者指南](#)》中的[在亚马逊 QLDB 中导出日记](#)。

- 有关 API 的详细信息，请参阅ExportsForLedger《[AWS CLI 命令参考](#)》中的[ListJournalS3](#)。

## list-journal-s3-exports

以下代码示例演示了如何使用 list-journal-s3-exports。

### AWS CLI

#### 列出日记账导出任务

以下list-journal-s3-exports示例列出了与当前 AWS 账户和区域关联的所有分类账的日记账导出任务。

```
aws qlldb list-journal-s3-exports
```

输出：

```
{
  "JournalS3Exports": [
    {
      "Status": "IN_PROGRESS",
      "LedgerName": "myExampleLedger",
      "S3ExportConfiguration": {
        "EncryptionConfiguration": {
          "ObjectEncryptionType": "SSE_S3"
        },
        "Bucket": "awsExampleBucket",
        "Prefix": "ledgerexport1/"
      },
      "RoleArn": "arn:aws:iam::123456789012:role/my-s3-export-role",
      "ExportCreationTime": 1568847801.418,
    }
  ]
}
```



```

    "ExportId": "ADR20NPKN5LINYGb4dp7yZ",
    "InclusiveStartTime": 1568764800.0,
    "ExclusiveEndTime": 1568847599.0
  },
  {
    "Status": "COMPLETED",
    "LedgerName": "myExampleLedger2",
    "S3ExportConfiguration": {
      "EncryptionConfiguration": {
        "ObjectEncryptionType": "SSE_S3"
      },
      "Bucket": "awsExampleBucket",
      "Prefix": "ledgerexport1/"
    },
    "RoleArn": "arn:aws:iam::123456789012:role/my-s3-export-role",
    "ExportCreationTime": 1568846847.638,
    "ExportId": "2pdvW8UQrjBAiYTMehEJDI",
    "InclusiveStartTime": 1568592000.0,
    "ExclusiveEndTime": 1568764800.0
  }
]
}

```

有关更多信息，请参阅《[亚马逊 QLDB 开发者指南](#)》中的[在亚马逊 QLDB 中导出日记](#)。

- 有关 API 的详细信息，请参阅《[AWS CLI 命令参考](#)》中的 [ListJournalS3Exports](#)。

## list-ledgers

以下代码示例演示了如何使用 list-ledgers。

### AWS CLI

列出您的可用分类账

以下list-ledgers示例列出了与当前 AWS 账户和区域关联的所有分类账。

```
aws qlldb list-ledgers
```

输出：

```
{
  "Ledgers": [
```

```
{
  "State": "ACTIVE",
  "CreationDateTime": 1568839243.951,
  "Name": "myExampleLedger"
},
{
  "State": "ACTIVE",
  "CreationDateTime": 1568839543.557,
  "Name": "myExampleLedger2"
}
]
```

有关更多信息，请参阅《Amazon QLDB 开发人员指南》中的 [Amazon QLDB 分类账的基本操作](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListLedgers](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出附在账本上的标签

以下`list-tags-for-resource`示例列出了附加到指定账本的所有标签。

```
aws qlldb list-tags-for-resource \
  --resource-arn arn:aws:qlldb:us-west-2:123456789012:ledger/myExampleLedger
```

输出：

```
{
  "Tags": {
    "IsTest": "true",
    "Domain": "Test"
  }
}
```

有关更多信息，请参阅亚马逊 Q LDB 开发者指南中的[为亚马逊 QLDB 资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## stream-journal-to-kinesis

以下代码示例演示了如何使用 stream-journal-to-kinesis。

### AWS CLI

#### 示例 1：使用输入文件将日记数据流式传输到 Kinesis Data Streams

以下 stream-journal-to-kinesis 示例从名为的账本创建指定日期和时间范围内的日记账数据流 myExampleLedger。该流将数据发送到指定的 Amazon Kinesis 数据流。

```
aws qlldb stream-journal-to-kinesis \  
  --ledger-name myExampleLedger \  
  --inclusive-start-time 2020-05-29T00:00:00Z \  
  --exclusive-end-time 2020-05-29T23:59:59Z \  
  --role-arn arn:aws:iam::123456789012:role/my-kinesis-stream-role \  
  --kinesis-configuration file://my-kinesis-config.json \  
  --stream-name myExampleLedger-stream
```

my-kinesis-config.json 的内容：

```
{  
  "StreamArn": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-for-qlldb",  
  "AggregationEnabled": true  
}
```

输出：

```
{  
  "StreamId": "7ISCKqwe4y25YyHLzYUFAf"  
}
```

有关更多信息，请参阅亚马逊 QLDB 开发者指南中的[从亚马逊 QLDB 流式传输日记数据](#)。

#### 示例 2：使用速记语法将日记数据流式传输到 Kinesis Data Streams

以下 stream-journal-to-kinesis 示例从名为的账本创建指定日期和时间范围内的日记账数据流 myExampleLedger。该流将数据发送到指定的 Amazon Kinesis 数据流。

```
aws qlldb stream-journal-to-kinesis \  
  --ledger-name myExampleLedger \  
  --inclusive-start-time 2020-05-29T00:00:00Z \  
  --exclusive-end-time 2020-05-29T23:59:59Z \  
  --role-arn arn:aws:iam::123456789012:role/my-kinesis-stream-role \  
  --kinesis-configuration file://my-kinesis-config.json \  
  --stream-name myExampleLedger-stream
```

```
--ledger-name myExampleLedger \  
--inclusive-start-time 2020-05-29T00:00:00Z \  
--exclusive-end-time 2020-05-29T23:59:59Z \  
--role-arn arn:aws:iam::123456789012:role/my-kinesis-stream-role \  
--stream-name myExampleLedger-stream \  
--kinesis-configuration StreamArn=arn:aws:kinesis:us-east-1:123456789012:stream/  
stream-for-qldb,AggregationEnabled=true
```

输出：

```
{  
  "StreamId": "7ISckqwe4y25YyHLzYUFAf"  
}
```

有关更多信息，请参阅亚马逊 QLDB 开发者指南中的[从亚马逊 QLDB 流式传输日记数据](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StreamJournalToKinesis](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为账本添加标签

以下tag-resource示例向指定的账本添加一组标签。

```
aws qldb tag-resource \  
  --resource-arn arn:aws:qldb:us-west-2:123456789012:ledger/myExampleLedger \  
  --tags IsTest=true,Domain=Test
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 QLDB 开发者指南中的[为亚马逊 QLDB 资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

## AWS CLI

### 要从资源中删除标签

以下`untag-resource`示例从指定账本中删除带有指定标签键的标签。

```
aws qlldb untag-resource \  
  --resource-arn arn:aws:qlldb:us-west-2:123456789012:ledger/myExampleLedger \  
  --tag-keys IsTest Domain
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 Q [LDB 开发者指南中的为亚马逊 QLDB 资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-ledger-permissions-mode

以下代码示例演示了如何使用 `update-ledger-permissions-mode`。

### AWS CLI

#### 示例 1：将账本的权限模式更新为标准

以下`update-ledger-permissions-mode`示例将STANDARD权限模式分配给指定的账本。

```
aws qlldb update-ledger-permissions-mode \  
  --name myExampleLedger \  
  --permissions-mode STANDARD
```

输出：

```
{  
  "Name": "myExampleLedger",  
  "Arn": "arn:aws:qlldb:us-west-2:123456789012:ledger/myExampleLedger",  
  "PermissionsMode": "STANDARD"  
}
```

#### 示例 2：将账本的权限模式更新为 ALLOW\_ALL

以下`update-ledger-permissions-mode`示例将ALLOW\_ALL权限模式分配给指定的账本。

```
aws qlldb update-ledger-permissions-mode \  
  --name myExampleLedger \  
  --permissions-mode ALLOW_ALL
```

输出：

```
{  
  "Name": "myExampleLedger",  
  "Arn": "arn:aws:qlldb:us-west-2:123456789012:ledger/myExampleLedger",  
  "PermissionsMode": "ALLOW_ALL"  
}
```

有关更多信息，请参阅《Amazon QLDB 开发人员指南》中的 [Amazon QLDB 分类账的基本操作](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateLedgerPermissionsMode](#) 中的。

## update-ledger

以下代码示例演示了如何使用 update-ledger。

### AWS CLI

示例 1：更新账本的删除保护属性

以下 update-ledger 示例更新了指定的账本以禁用删除保护功能。

```
aws qlldb update-ledger \  
  --name myExampleLedger \  
  --no-deletion-protection
```

输出：

```
{  
  "CreationDateTime": 1568839243.951,  
  "Arn": "arn:aws:qlldb:us-west-2:123456789012:ledger/myExampleLedger",  
  "DeletionProtection": false,  
  "Name": "myExampleLedger",  
  "State": "ACTIVE"  
}
```

示例 2：将账本的 AWS KMS 密钥更新为客户托管密钥

以下update-ledger示例将指定的账本更新为使用客户托管的 KMS 密钥进行静态加密。

```
aws qlldb update-ledger \  
  --name myExampleLedger \  
  --kms-key arn:aws:kms:us-west-2:123456789012:key/a1b2c3d4-5678-90ab-cdef-  
EXAMPLE11111
```

输出：

```
{  
  "CreationDateTime": 1568839243.951,  
  "Arn": "arn:aws:qlldb:us-west-2:123456789012:ledger/myExampleLedger",  
  "DeletionProtection": false,  
  "Name": "myExampleLedger",  
  "State": "ACTIVE",  
  "EncryptionDescription": {  
    "KmsKeyArn": "arn:aws:kms:us-west-2:123456789012:key/a1b2c3d4-5678-90ab-  
cdef-EXAMPLE11111",  
    "EncryptionStatus": "UPDATING"  
  }  
}
```

示例 3：将账本的 AWS KMS 密钥更新为 AWS 自有密钥

以下update-ledger示例将指定的账本更新为使用 AWS 拥有的 KMS 密钥进行静态加密。

```
aws qlldb update-ledger \  
  --name myExampleLedger \  
  --kms-key AWS_OWNED_KMS_KEY
```

输出：

```
{  
  "CreationDateTime": 1568839243.951,  
  "Arn": "arn:aws:qlldb:us-west-2:123456789012:ledger/myExampleLedger",  
  "DeletionProtection": false,  
  "Name": "myExampleLedger",  
  "State": "ACTIVE",  
  "EncryptionDescription": {  
    "KmsKeyArn": "AWS_OWNED_KMS_KEY",  
    "EncryptionStatus": "UPDATING"  
  }  
}
```

```
}  
}
```

有关更多信息，请参阅《Amazon QLDB 开发人员指南》中的 [Amazon QLDB 分类账的基本操作](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateLedger](#)中的。

## 使用 Amazon RDS 示例 AWS CLI

以下代码示例向您展示了如何在 Amazon RDS 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### add-option-to-option-group

以下代码示例演示了如何使用 add-option-to-option-group。

AWS CLI

向选项组添加选项

以下add-option-to-option-group示例将一个选项添加到指定的选项组。

```
aws rds add-option-to-option-group \  
  --option-group-name myoptiongroup \  
  --options OptionName=OEM,Port=5500,DBSecurityGroupMemberships=default \  
  --apply-immediately
```



输出：

```
{
  "OptionGroup": {
    "OptionGroupName": "myoptiongroup",
    "OptionGroupDescription": "Test Option Group",
    "EngineName": "oracle-ee",
    "MajorEngineVersion": "12.1",
    "Options": [
      {
        "OptionName": "Timezone",
        "OptionDescription": "Change time zone",
        "Persistent": true,
        "Permanent": false,
        "OptionSettings": [
          {
            "Name": "TIME_ZONE",
            "Value": "Australia/Sydney",
            "DefaultValue": "UTC",
            "Description": "Specifies the timezone the user wants to
change the system time to",
            "ApplyType": "DYNAMIC",
            "DataType": "STRING",
            "AllowedValues": "Africa/Cairo,Africa/Casablanca,Africa/
Harare,Africa/Lagos,Africa/Luanda,Africa/Monrovia,Africa/Nairobi,Africa/
Tripoli,Africa/Windhoek,America/Araguaina,America/Argentina/Buenos_Aires,America/
Asuncion,America/Bogota,America/Caracas,America/Chicago,America/Chihuahua,America/
Cuiaba,America/Denver,America/Detroit,America/Fortaleza,America/Godthab,America/
Guatemala,America/Halifax,America/Lima,America/Los_Angeles,America/Manaus,America/
Matamoros,America/Mexico_City,America/Monterrey,America/Montevideo,America/
New_York,America/Phoenix,America/Santiago,America/Sao_Paulo,America/Tijuana,America/
Toronto,Asia/Amman,Asia/Ashgabat,Asia/Baghdad,Asia/Baku,Asia/Bangkok,Asia/
Beirut,Asia/Calcutta,Asia/Damascus,Asia/Dhaka,Asia/Hong_Kong,Asia/Irkutsk,Asia/
Jakarta,Asia/Jerusalem,Asia/Kabul,Asia/Karachi,Asia/Kathmandu,Asia/Kolkata,Asia/
Krasnoyarsk,Asia/Magadan,Asia/Manila,Asia/Muscat,Asia/Novosibirsk,Asia/Rangoon,Asia/
Riyadh,Asia/Seoul,Asia/Shanghai,Asia/Singapore,Asia/Taipei,Asia/Tehran,Asia/
Tokyo,Asia/Ulaanbaatar,Asia/Vladivostok,Asia/Yakutsk,Asia/Yerevan,Atlantic/
Azores,Atlantic/Cape_Verde,Australia/Adelaide,Australia/Brisbane,Australia/
Darwin,Australia/Eucla,Australia/Hobart,Australia/Lord_Howe,Australia/
Perth,Australia/Sydney,Brazil/DeNoronha,Brazil/East,Canada/Newfoundland,Canada/
Saskatchewan,Etc/GMT-3,Europe/Amsterdam,Europe/Athens,Europe/Berlin,Europe/
Dublin,Europe/Helsinki,Europe/Kaliningrad,Europe/London,Europe/Madrid,Europe/
Moscow,Europe/Paris,Europe/Prague,Europe/Rome,Europe/Sarajevo,Pacific/Apia,Pacific/
Auckland,Pacific/Chatham,Pacific/Fiji,Pacific/Guam,Pacific/Honolulu,Pacific/
```

```

Kiritimati,Pacific/Marquesas,Pacific/Samoa,Pacific/Tongatapu,Pacific/Wake,US/
Alaska,US/Central,US/East-Indiana,US/Eastern,US/Pacific,UTC",
        "IsModifiable": true,
        "IsCollection": false
    }
],
"DBSecurityGroupMemberships": [],
"VpcSecurityGroupMemberships": []
},
{
    "OptionName": "OEM",
    "OptionDescription": "Oracle 12c EM Express",
    "Persistent": false,
    "Permanent": false,
    "Port": 5500,
    "OptionSettings": [],
    "DBSecurityGroupMemberships": [
        {
            "DBSecurityGroupName": "default",
            "Status": "authorized"
        }
    ],
    "VpcSecurityGroupMemberships": []
}
],
"AllowsVpcAndNonVpcInstanceMemberships": false,
"OptionGroupArn": "arn:aws:rds:us-east-1:123456789012:og:myoptiongroup"
}
}

```

有关更多信息，请参阅 Amazon RDS 用户指南中的[向选项组添加选项](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[AddOptionToOptionGroup](#)中的。

## add-role-to-db-cluster

以下代码示例演示了如何使用 `add-role-to-db-cluster`。

### AWS CLI

将 AWS 身份和访问管理 (IAM) Access Management 角色与数据库集群关联

以下 `add-role-to-db-cluster` 示例将角色与数据库集群相关联。

```
aws rds add-role-to-db-cluster \  
  --db-cluster-identifier mydbcluster \  
  --role-arn arn:aws:iam::123456789012:role/RDSLoadFromS3
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 Aurora 用户指南中的“[将 IAM 角色与亚马逊 Aurora MySQL 数据库集群关联](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddRoleToDbCluster](#)中的。

## add-role-to-db-instance

以下代码示例演示了如何使用 add-role-to-db-instance。

### AWS CLI

将 AWS 身份和访问管理 (IAM) Access Management 角色与数据库实例关联

以下add-role-to-db-instance示例将角色添加到名为的 Oracle 数据库实例test-instance。

```
aws rds add-role-to-db-instance \  
  --db-instance-identifier test-instance \  
  --feature-name S3_INTEGRATION \  
  --role-arn arn:aws:iam::111122223333:role/rds-s3-integration-role
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon RDS 用户指南中的 Amazon RDS Oracle 与 Amazon [S3 集成的先决条件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddRoleToDbInstance](#)中的。

## add-source-identifier-to-subscription

以下代码示例演示了如何使用 add-source-identifier-to-subscription。

### AWS CLI

向订阅添加来源标识符

以下add-source-identifier示例向现有订阅添加另一个源标识符。

```
aws rds add-source-identifier-to-subscription \  
  --subscription-name my-instance-events \  
  --source-identifier test-instance-repl
```

输出：

```
{  
  "EventSubscription": {  
    "SubscriptionCreationTime": "Tue Jul 31 23:22:01 UTC 2018",  
    "CustSubscriptionId": "my-instance-events",  
    "EventSubscriptionArn": "arn:aws:rds:us-east-1:123456789012:es:my-instance-  
events",  
    "Enabled": false,  
    "Status": "modifying",  
    "EventCategoriesList": [  
      "backup",  
      "recovery"  
    ],  
    "CustomerAwsId": "123456789012",  
    "SnsTopicArn": "arn:aws:sns:us-east-1:123456789012:interesting-events",  
    "SourceType": "db-instance",  
    "SourceIdsList": [  
      "test-instance",  
      "test-instance-repl"  
    ]  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddSourceIdentifierToSubscription](#)中的。

## add-tags-to-resource

以下代码示例演示了如何使用 add-tags-to-resource。

### AWS CLI

为资源添加标签

以下add-tags-to-resource示例为 RDS 数据库添加标签。

```
aws rds add-tags-to-resource \  
  --resource-name arn:aws:rds:us-east-1:123456789012:db:database-mysql \  
  --tags Key=Value
```

```
--tags "[{\"Key\": \"Name\"}, {\"Value\": \"MyDatabase\"}], {\"Key\": \"Environment\", \"Value\": \"test\"}]"
```

此命令不生成任何输出。

有关更多信息，请参阅 [Amazon RDS 用户指南中的为 Amazon RDS 资源添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AddTagsToResource](#) 中的。

## apply-pending-maintenance-action

以下代码示例演示了如何使用 apply-pending-maintenance-action。

### AWS CLI

应用待执行的维护操作

以下 apply-pending-maintenance-action 示例对数据库集群应用待处理的维护操作。

```
aws rds apply-pending-maintenance-action \  
  --resource-identifier arn:aws:rds:us-east-1:123456789012:cluster:my-db-cluster \  
  --apply-action system-update \  
  --opt-in-type immediate
```

输出：

```
{  
  "ResourcePendingMaintenanceActions": {  
    "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:cluster:my-db-cluster",  
    "PendingMaintenanceActionDetails": [  
      {  
        "Action": "system-update",  
        "OptInStatus": "immediate",  
        "CurrentApplyDate": "2021-01-23T01:07:36.100Z",  
        "Description": "Upgrade to Aurora PostgreSQL 3.3.2"  
      }  
    ]  
  }  
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的 [维护数据库实例](#) 和 [亚马逊 Aurora 用户指南中的维护亚马逊 Aurora 数据库集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ApplyPendingMaintenanceAction](#)中的。

## authorize-db-security-group-ingress

以下代码示例演示了如何使用 authorize-db-security-group-ingress。

### AWS CLI

将 AWS 身份和访问管理 (IAM) Access Management 角色与数据库实例关联

以下authorize-db-security-group-ingress示例使用 CIDR IP 范围 192.0.2.0/24 的入口规则配置默认安全组。

```
aws rds authorize-db-security-group-ingress \  
  --db-security-group-name default \  
  --cidrip 192.0.2.0/24
```

输出：

```
{  
  "DBSecurityGroup": {  
    "OwnerId": "123456789012",  
    "DBSecurityGroupName": "default",  
    "DBSecurityGroupDescription": "default",  
    "EC2SecurityGroups": [],  
    "IPRanges": [  
      {  
        "Status": "authorizing",  
        "CIDRIP": "192.0.2.0/24"  
      }  
    ],  
    "DBSecurityGroupArn": "arn:aws:rds:us-east-1:111122223333:secgrp:default"  
  }  
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[授权从 IP 范围访问数据库安全组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AuthorizeDbSecurityGroupIngress](#)中的。

## backtrack-db-cluster

以下代码示例演示了如何使用 backtrack-db-cluster。

## AWS CLI

### 回溯 Aurora 数据库集群

以下backtrack-db-cluster示例将指定的数据库集群示例集群回溯到 2018 年 3 月 19 日上午 10 点。

```
aws rds backtrack-db-cluster --db-cluster-identifier sample-cluster --backtrack-to 2018-03-19T10:00:00+00:00
```

此命令输出一个 JSON 块，用于确认对 RDS 资源的更改。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BacktrackDbCluster](#)中的。

## cancel-export-task

以下代码示例演示了如何使用 cancel-export-task。

### AWS CLI

#### 取消将快照导出到 Amazon S3

以下cancel-export-task示例取消正在进行的导出任务，该任务正在将快照导出到 Amazon S3。

```
aws rds cancel-export-task \
  --export-task-identifier my-s3-export-1
```

输出：

```
{
  "ExportTaskIdentifier": "my-s3-export-1",
  "SourceArn": "arn:aws:rds:us-east-1:123456789012:snapshot:publisher-final-snapshot",
  "SnapshotTime": "2019-03-24T20:01:09.815Z",
  "S3Bucket": "mybucket",
  "S3Prefix": "",
  "IamRoleArn": "arn:aws:iam::123456789012:role/service-role/export-snap-S3-role",
  "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/abcd0000-7bfd-4594-af38-aabbccddeeff",
  "Status": "CANCELING",
  "PercentProgress": 0,
  "TotalExtractedDataInGB": 0
}
```

```
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[取消快照导出任务](#)或 Amazon Aurora 用户指南中的[取消快照导出任务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CancelExportTask](#)中的。

## copy-db-cluster-parameter-group

以下代码示例演示了如何使用 `copy-db-cluster-parameter-group`。

### AWS CLI

#### 复制数据库集群参数组

以下 `copy-db-cluster-parameter-group` 示例创建了数据库集群参数组的副本。

```
aws rds copy-db-cluster-parameter-group \
  --source-db-cluster-parameter-group-identifier mydbclusterpg \
  --target-db-cluster-parameter-group-identifier mydbclusterpgcopy \
  --target-db-cluster-parameter-group-description "Copy of mydbclusterpg parameter
group"
```

输出：

```
{
  "DBClusterParameterGroup": {
    "DBClusterParameterGroupName": "mydbclusterpgcopy",
    "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterpgcopy",
    "DBParameterGroupFamily": "aurora-mysql5.7",
    "Description": "Copy of mydbclusterpg parameter group"
  }
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[复制数据库集群参数组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CopyDbClusterParameterGroup](#)中的。

## copy-db-cluster-snapshot

以下代码示例演示了如何使用 `copy-db-cluster-snapshot`。



## AWS CLI

### 复制数据库集群快照

以下copy-db-cluster-snapshot示例创建数据库集群快照的副本，包括其标签。

```
aws rds copy-db-cluster-snapshot \  
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-  
east-1:123456789012:cluster-snapshot:rds:myaurora-2019-06-04-09-16  
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy \  
  --copy-tags
```

输出：

```
{  
  "DBClusterSnapshot": {  
    "AvailabilityZones": [  
      "us-east-1a",  
      "us-east-1b",  
      "us-east-1e"  
    ],  
    "DBClusterSnapshotIdentifier": "myclustersnapshotcopy",  
    "DBClusterIdentifier": "myaurora",  
    "SnapshotCreateTime": "2019-06-04T09:16:42.649Z",  
    "Engine": "aurora-mysql",  
    "AllocatedStorage": 0,  
    "Status": "available",  
    "Port": 0,  
    "VpcId": "vpc-6594f31c",  
    "ClusterCreateTime": "2019-04-15T14:18:42.785Z",  
    "MasterUsername": "myadmin",  
    "EngineVersion": "5.7.mysql_aurora.2.04.2",  
    "LicenseModel": "aurora-mysql",  
    "SnapshotType": "manual",  
    "PercentProgress": 100,  
    "StorageEncrypted": true,  
    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE",  
    "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:123456789012:cluster-  
snapshot:myclustersnapshotcopy",  
    "IAMDatabaseAuthenticationEnabled": false  
  }  
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[复制快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CopyDbClusterSnapshot](#)中的。

## copy-db-parameter-group

以下代码示例演示了如何使用 copy-db-parameter-group。

### AWS CLI

#### 复制数据库集群参数组

以下copy-db-parameter-group示例创建了数据库参数组的副本。

```
aws rds copy-db-parameter-group \  
  --source-db-parameter-group-identifier mydbpg \  
  --target-db-parameter-group-identifier mydbpgcopy \  
  --target-db-parameter-group-description "Copy of mydbpg parameter group"
```

输出：

```
{  
  "DBParameterGroup": {  
    "DBParameterGroupName": "mydbpgcopy",  
    "DBParameterGroupArn": "arn:aws:rds:us-east-1:814387698303:pg:mydbpgcopy",  
    "DBParameterGroupFamily": "mysql5.7",  
    "Description": "Copy of mydbpg parameter group"  
  }  
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[复制数据库参数组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CopyDbParameterGroup](#)中的。

## copy-db-snapshot

以下代码示例演示了如何使用 copy-db-snapshot。

### AWS CLI

#### 复制数据库快照

以下copy-db-snapshot示例创建数据库快照的副本。

```
aws rds copy-db-snapshot \  
  --source-db-snapshot-identifier rds:database-mysql-2019-06-06-08-38  
  --target-db-snapshot-identifier mydbsnapshotcopy
```

输出：

```
{  
  "DBSnapshot": {  
    "VpcId": "vpc-6594f31c",  
    "Status": "creating",  
    "Encrypted": true,  
    "SourceDBSnapshotIdentifier": "arn:aws:rds:us-  
east-1:123456789012:snapshot:rds:database-mysql-2019-06-06-08-38",  
    "MasterUsername": "admin",  
    "Iops": 1000,  
    "Port": 3306,  
    "LicenseModel": "general-public-license",  
    "DBSnapshotArn": "arn:aws:rds:us-  
east-1:123456789012:snapshot:mydbsnapshotcopy",  
    "EngineVersion": "5.6.40",  
    "OptionGroupName": "default:mysql-5-6",  
    "ProcessorFeatures": [],  
    "Engine": "mysql",  
    "StorageType": "io1",  
    "DbiResourceId": "db-ZI7UJ5BLKMBYFGX7FDENCKADC4",  
    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE",  
    "SnapshotType": "manual",  
    "IAMDatabaseAuthenticationEnabled": false,  
    "SourceRegion": "us-east-1",  
    "DBInstanceIdentifier": "database-mysql",  
    "InstanceCreateTime": "2019-04-30T15:45:53.663Z",  
    "AvailabilityZone": "us-east-1f",  
    "PercentProgress": 0,  
    "AllocatedStorage": 100,  
    "DBSnapshotIdentifier": "mydbsnapshotcopy"  
  }  
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[复制快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CopyDbSnapshot](#)中的。

## copy-option-group

以下代码示例演示了如何使用 `copy-option-group`。

### AWS CLI

#### 复制选项组

以下 `copy-option-group` 示例创建了选项组的副本。

```
aws rds copy-option-group \  
  --source-option-group-identifier myoptiongroup \  
  --target-option-group-identifier new-option-group \  
  --target-option-group-description "My option group copy"
```

输出：

```
{  
  "OptionGroup": {  
    "Options": [],  
    "OptionGroupName": "new-option-group",  
    "MajorEngineVersion": "11.2",  
    "OptionGroupDescription": "My option group copy",  
    "AllowsVpcAndNonVpcInstanceMemberships": true,  
    "EngineName": "oracle-ee",  
    "OptionGroupArn": "arn:aws:rds:us-east-1:123456789012:og:new-option-group"  
  }  
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[制作选项组的副本](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CopyOptionGroup](#) 中的。

## create-blue-green-deployment

以下代码示例演示了如何使用 `create-blue-green-deployment`。

### AWS CLI

示例 1：为适用于 MySQL 的 RDS 数据库实例创建蓝/绿部署

以下 `create-blue-green-deployment` 示例为 MySQL 数据库实例创建蓝/绿部署。

```
aws rds create-blue-green-deployment \  
  --blue-green-deployment-name bgd-cli-test-instance \  
  --source arn:aws:rds:us-east-1:123456789012:db:my-db-instance \  
  --target-engine-version 8.0 \  
  --target-db-parameter-group-name mysql-80-group
```

输出：

```
{  
  "BlueGreenDeployment": {  
    "BlueGreenDeploymentIdentifier": "bgd-v53303651eexfake",  
    "BlueGreenDeploymentName": "bgd-cli-test-instance",  
    "Source": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance",  
    "SwitchoverDetails": [  
      {  
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-  
instance"  
      },  
      {  
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-  
instance-replica-1"  
      },  
      {  
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-  
instance-replica-2"  
      },  
      {  
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-  
instance-replica-3"  
      }  
    ],  
    "Tasks": [  
      {  
        "Name": "CREATING_READ_REPLICA_OF_SOURCE",  
        "Status": "PENDING"  
      },  
      {  
        "Name": "DB_ENGINE_VERSION_UPGRADE",  
        "Status": "PENDING"  
      },  
      {  
        "Name": "CONFIGURE_BACKUPS",  
        "Status": "PENDING"  
      }  
    ]  
  }  
}
```

```

    },
    {
      "Name": "CREATING_TOPOLOGY_OF_SOURCE",
      "Status": "PENDING"
    }
  ],
  "Status": "PROVISIONING",
  "CreateTime": "2022-02-25T21:18:51.183000+00:00"
}
}

```

有关更多信息，请参阅 Amazon RDS 用户指南中的[创建蓝/绿部署](#)。

示例 2：为 Aurora MySQL 数据库集群创建蓝/绿部署

以下 create-blue-green-deployment 示例为 Aurora MySQL 数据库集群创建蓝/绿部署。

```

aws rds create-blue-green-deployment \
  --blue-green-deployment-name my-blue-green-deployment \
  --source arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster \
  --target-engine-version 8.0 \
  --target-db-cluster-parameter-group-name ams-80-binlog-enabled \
  --target-db-parameter-group-name mysql-80-cluster-group

```

输出：

```

{
  "BlueGreenDeployment": {
    "BlueGreenDeploymentIdentifier": "bgd-wi89nwzglccsfake",
    "BlueGreenDeploymentName": "my-blue-green-deployment",
    "Source": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster",
    "SwitchoverDetails": [
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster",
        "Status": "PROVISIONING"
      },
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-1",
        "Status": "PROVISIONING"
      }
    ]
  }
}

```

```
    {
      "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-
cluster-2",
      "Status": "PROVISIONING"
    },
    {
      "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-
cluster-3",
      "Status": "PROVISIONING"
    },
    {
      "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-
excluded-member-endpoint",
      "Status": "PROVISIONING"
    },
    {
      "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-
reader-endpoint",
      "Status": "PROVISIONING"
    }
  ],
  "Tasks": [
    {
      "Name": "CREATING_READ_REPLICA_OF_SOURCE",
      "Status": "PENDING"
    },
    {
      "Name": "DB_ENGINE_VERSION_UPGRADE",
      "Status": "PENDING"
    },
    {
      "Name": "CREATE_DB_INSTANCES_FOR_CLUSTER",
      "Status": "PENDING"
    },
    {
      "Name": "CREATE_CUSTOM_ENDPOINTS",
      "Status": "PENDING"
    }
  ],
  "Status": "PROVISIONING",
  "CreateTime": "2022-02-25T21:12:00.288000+00:00"
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[创建蓝/绿部署](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateBlueGreenDeployment](#)中的。

## create-db-cluster-endpoint

以下代码示例演示了如何使用 create-db-cluster-endpoint。

### AWS CLI

#### 创建自定义数据库集群终端节点

以下 create-db-cluster-endpoint 示例创建自定义数据库集群终端节点并将其与指定的 Aurora 数据库集群关联。

```
aws rds create-db-cluster-endpoint \  
  --db-cluster-endpoint-identifier mycustomendpoint \  
  --endpoint-type reader \  
  --db-cluster-identifier mydbcluster \  
  --static-members dbinstance1 dbinstance2
```

输出：

```
{  
  "DBClusterEndpointIdentifier": "mycustomendpoint",  
  "DBClusterIdentifier": "mydbcluster",  
  "DBClusterEndpointResourceIdentifier": "cluster-endpoint-ANPAJ4AE5446DAEXAMPLE",  
  "Endpoint": "mycustomendpoint.cluster-custom-cnexample.us-  
east-1.rds.amazonaws.com",  
  "Status": "creating",  
  "EndpointType": "CUSTOM",  
  "CustomEndpointType": "READER",  
  "StaticMembers": [  
    "dbinstance1",  
    "dbinstance2"  
  ],  
  "ExcludedMembers": [],  
  "DBClusterEndpointArn": "arn:aws:rds:us-east-1:123456789012:cluster-  
endpoint:mycustomendpoint"  
}
```

有关更多信息，请参阅[亚马逊 Aurora 用户指南中的亚马逊 Aurora 连接管理](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDbClusterEndpoint](#)中的。

## create-db-cluster-parameter-group

以下代码示例演示了如何使用 create-db-cluster-parameter-group。

### AWS CLI

#### 创建数据库集群参数组

以下create-db-cluster-parameter-group示例创建数据库集群参数组。

```
aws rds create-db-cluster-parameter-group \  
  --db-cluster-parameter-group-name mydbclusterparametergroup \  
  --db-parameter-group-family aurora5.6 \  
  --description "My new cluster parameter group"
```

输出：

```
{  
  "DBClusterParameterGroup": {  
    "DBClusterParameterGroupName": "mydbclusterparametergroup",  
    "DBParameterGroupFamily": "aurora5.6",  
    "Description": "My new cluster parameter group",  
    "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-  
pg:mydbclusterparametergroup"  
  }  
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[创建数据库集群参数组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDbClusterParameterGroup](#)中的。

## create-db-cluster-snapshot

以下代码示例演示了如何使用 create-db-cluster-snapshot。

### AWS CLI

#### 创建数据库集群快照

以下create-db-cluster-snapshot示例创建数据库集群快照。

```
aws rds create-db-cluster-snapshot \  
  --db-cluster-identifier mydbcluster \  
  --db-cluster-snapshot-identifier mydbclustersnapshot
```

输出：

```
{  
  "DBClusterSnapshot": {  
    "AvailabilityZones": [  
      "us-east-1a",  
      "us-east-1b",  
      "us-east-1e"  
    ],  
    "DBClusterSnapshotIdentifier": "mydbclustersnapshot",  
    "DBClusterIdentifier": "mydbcluster",  
    "SnapshotCreateTime": "2019-06-18T21:21:00.469Z",  
    "Engine": "aurora-mysql",  
    "AllocatedStorage": 1,  
    "Status": "creating",  
    "Port": 0,  
    "VpcId": "vpc-6594f31c",  
    "ClusterCreateTime": "2019-04-15T14:18:42.785Z",  
    "MasterUsername": "myadmin",  
    "EngineVersion": "5.7.mysql_aurora.2.04.2",  
    "LicenseModel": "aurora-mysql",  
    "SnapshotType": "manual",  
    "PercentProgress": 0,  
    "StorageEncrypted": true,  
    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE",  
    "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:123456789012:cluster-  
snapshot:mydbclustersnapshot",  
    "IAMDatabaseAuthenticationEnabled": false  
  }  
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[创建数据库集群快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateDbClusterSnapshot](#)中的。

## create-db-cluster

以下代码示例演示了如何使用 create-db-cluster。

## AWS CLI

### 示例 1：创建兼容 MySQL 5.7 的数据库集群

以下 `create-db-cluster` 示例使用默认引擎版本创建兼容 MySQL 5.7 的数据库集群。将示例密码 `secret99` 替换为安全密码。当您使用控制台创建数据库集群时，Amazon RDS 会自动为您的数据库集群创建写入数据库实例。但是，当您使用 AWS CLI 创建数据库集群时，必须使用 `create-db-instance` AWS CLI 命令为数据库集群显式创建写入数据库实例。

```
aws rds create-db-cluster \  
  --db-cluster-identifier sample-cluster \  
  --engine aurora-mysql \  
  --engine-version 5.7 \  
  --master-username admin \  
  --master-user-password secret99 \  
  --db-subnet-group-name default \  
  --vpc-security-group-ids sg-0b9130572daf3dc16
```

输出：

```
{  
  "DBCluster": {  
    "DBSubnetGroup": "default",  
    "VpcSecurityGroups": [  
      {  
        "VpcSecurityGroupId": "sg-0b9130572daf3dc16",  
        "Status": "active"  
      }  
    ],  
    "AllocatedStorage": 1,  
    "AssociatedRoles": [],  
    "PreferredBackupWindow": "09:12-09:42",  
    "ClusterCreateTime": "2023-02-27T23:21:33.048Z",  
    "DeletionProtection": false,  
    "IAMDatabaseAuthenticationEnabled": false,  
    "ReadReplicaIdentifiers": [],  
    "EngineMode": "provisioned",  
    "Engine": "aurora-mysql",  
    "StorageEncrypted": false,  
    "MultiAZ": false,  
    "PreferredMaintenanceWindow": "mon:04:31-mon:05:01",  
    "HttpEndpointEnabled": false,  
  }  
}
```

```

    "BackupRetentionPeriod": 1,
    "DbClusterResourceId": "cluster-ANPAJ4AE5446DAEXAMPLE",
    "DBClusterIdentifier": "sample-cluster",
    "AvailabilityZones": [
        "us-east-1a",
        "us-east-1b",
        "us-east-1e"
    ],
    "MasterUsername": "master",
    "EngineVersion": "5.7.mysql_aurora.2.11.1",
    "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",
    "DBClusterMembers": [],
    "Port": 3306,
    "Status": "creating",
    "Endpoint": "sample-cluster.cluster-cnpxample.us-east-1.rds.amazonaws.com",
    "DBClusterParameterGroup": "default.aurora-mysql5.7",
    "HostedZoneId": "Z2R2ITUGPM61AM",
    "ReaderEndpoint": "sample-cluster.cluster-ro-cnpxample.us-east-1.rds.amazonaws.com",
    "CopyTagsToSnapshot": false
  }
}

```

## 示例 2：创建与 PostgreSQL 兼容的数据库集群

以下 `create-db-cluster` 示例使用默认引擎版本创建与 PostgreSQL 兼容的数据库集群。将示例密码 `secret99` 替换为安全密码。当您使用控制台创建数据库集群时，Amazon RDS 会自动为您的数据库集群创建写入数据库实例。但是，当您使用 AWS CLI 创建数据库集群时，必须使用 `CL create-db-instance AWS I` 命令为数据库集群显式创建写入器数据库实例。

```

aws rds create-db-cluster \
  --db-cluster-identifier sample-pg-cluster \
  --engine aurora-postgresql \
  --master-username master \
  --master-user-password secret99 \
  --db-subnet-group-name default \
  --vpc-security-group-ids sg-0b9130572daf3dc16

```

输出：

```

{
  "DBCluster": {

```

```
    "Endpoint": "sample-pg-cluster.cluster-cnpxample.us-
east-1.rds.amazonaws.com",
    "HttpEndpointEnabled": false,
    "DBClusterMembers": [],
    "EngineMode": "provisioned",
    "CopyTagsToSnapshot": false,
    "HostedZoneId": "Z2R2ITUGPM61AM",
    "IAMDatabaseAuthenticationEnabled": false,
    "AllocatedStorage": 1,
    "VpcSecurityGroups": [
      {
        "VpcSecurityGroupId": "sg-0b9130572daf3dc16",
        "Status": "active"
      }
    ],
    "DeletionProtection": false,
    "StorageEncrypted": false,
    "BackupRetentionPeriod": 1,
    "PreferredBackupWindow": "09:56-10:26",
    "ClusterCreateTime": "2023-02-27T23:26:08.371Z",
    "DBClusterParameterGroup": "default.aurora-postgresql13",
    "EngineVersion": "13.7",
    "Engine": "aurora-postgresql",
    "Status": "creating",
    "DBClusterIdentifier": "sample-pg-cluster",
    "MultiAZ": false,
    "Port": 5432,
    "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-pg-
cluster",
    "AssociatedRoles": [],
    "DbClusterResourceId": "cluster-ANPAJ4AE5446DAEXAMPLE",
    "PreferredMaintenanceWindow": "wed:03:33-wed:04:03",
    "ReaderEndpoint": "sample-pg-cluster.cluster-ro-cnpxample.us-
east-1.rds.amazonaws.com",
    "MasterUsername": "master",
    "AvailabilityZones": [
      "us-east-1a",
      "us-east-1b",
      "us-east-1c"
    ],
    "ReadReplicaIdentifiers": [],
    "DBSubnetGroup": "default"
  }
```

```
}
```

有关更多信息，请参阅[亚马逊 Aurora 用户指南中的创建亚马逊 Aurora 数据库集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDbCluster](#)中的。

## create-db-instance-read-replica

以下代码示例演示了如何使用 `create-db-instance-read-replica`。

### AWS CLI

#### 创建数据库实例只读副本

此示例为名为的现有数据库实例创建只读副本`test-instance`。只读副本名为`test-instance-repl`。

```
aws rds create-db-instance-read-replica \  
  --db-instance-identifier test-instance-repl \  
  --source-db-instance-identifier test-instance
```

输出：

```
{  
  "DBInstance": {  
    "IAMDatabaseAuthenticationEnabled": false,  
    "MonitoringInterval": 0,  
    "DBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance-repl",  
    "ReadReplicaSourceDBInstanceIdentifier": "test-instance",  
    "DBInstanceIdentifier": "test-instance-repl",  
    ...some output truncated...  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDbInstanceReadReplica](#)中的。

## create-db-instance

以下代码示例演示了如何使用 `create-db-instance`。

## AWS CLI

### 创建数据库实例

以下 `create-db-instance` 示例使用所需的选项启动新的数据库实例。

```
aws rds create-db-instance \  
  --db-instance-identifier test-mysql-instance \  
  --db-instance-class db.t3.micro \  
  --engine mysql \  
  --master-username admin \  
  --master-user-password secret99 \  
  --allocated-storage 20
```

输出：

```
{  
  "DBInstance": {  
    "DBInstanceIdentifier": "test-mysql-instance",  
    "DBInstanceClass": "db.t3.micro",  
    "Engine": "mysql",  
    "DBInstanceStatus": "creating",  
    "MasterUsername": "admin",  
    "AllocatedStorage": 20,  
    "PreferredBackupWindow": "12:55-13:25",  
    "BackupRetentionPeriod": 1,  
    "DBSecurityGroups": [],  
    "VpcSecurityGroups": [  
      {  
        "VpcSecurityGroupId": "sg-12345abc",  
        "Status": "active"  
      }  
    ],  
    "DBParameterGroups": [  
      {  
        "DBParameterGroupName": "default.mysql5.7",  
        "ParameterApplyStatus": "in-sync"  
      }  
    ],  
    "DBSubnetGroup": {  
      "DBSubnetGroupName": "default",  
      "DBSubnetGroupDescription": "default",  
      "VpcId": "vpc-2ff2ff2f",
```

```
"SubnetGroupStatus": "Complete",
"Subnets": [
  {
    "SubnetIdentifier": "subnet-#####",
    "SubnetAvailabilityZone": {
      "Name": "us-west-2c"
    },
    "SubnetStatus": "Active"
  },
  {
    "SubnetIdentifier": "subnet-#####",
    "SubnetAvailabilityZone": {
      "Name": "us-west-2d"
    },
    "SubnetStatus": "Active"
  },
  {
    "SubnetIdentifier": "subnet-#####",
    "SubnetAvailabilityZone": {
      "Name": "us-west-2a"
    },
    "SubnetStatus": "Active"
  },
  {
    "SubnetIdentifier": "subnet-#####",
    "SubnetAvailabilityZone": {
      "Name": "us-west-2b"
    },
    "SubnetStatus": "Active"
  }
]
},
"PreferredMaintenanceWindow": "sun:08:07-sun:08:37",
"PendingModifiedValues": {
  "MasterUserPassword": "*****"
},
"MultiAZ": false,
"EngineVersion": "5.7.22",
"AutoMinorVersionUpgrade": true,
"ReadReplicaDBInstanceIdentifiers": [],
"LicenseModel": "general-public-license",
"OptionGroupMemberships": [
  {
    "OptionGroupName": "default:mysql-5-7",
```



```

        "Status": "in-sync"
      }
    ],
    "PubliclyAccessible": true,
    "StorageType": "gp2",
    "DbInstancePort": 0,
    "StorageEncrypted": false,
    "DbiResourceId": "db-5555EXAMPLE444444444EXAMPLE",
    "CACertificateIdentifier": "rds-ca-2019",
    "DomainMemberships": [],
    "CopyTagsToSnapshot": false,
    "MonitoringInterval": 0,
    "DBInstanceArn": "arn:aws:rds:us-west-2:123456789012:db:test-mysql-
instance",
    "IAMDatabaseAuthenticationEnabled": false,
    "PerformanceInsightsEnabled": false,
    "DeletionProtection": false,
    "AssociatedRoles": []
  }
}

```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[创建 Amazon RDS 数据库实例](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的[CreateDBInstance](#)。

## create-db-parameter-group

以下代码示例演示了如何使用 create-db-parameter-group。

### AWS CLI

#### 创建数据库参数组

以下 create-db-parameter-group 示例创建一个数据库参数组。

```

aws rds create-db-parameter-group \
  --db-parameter-group-name mydbparametergroup \
  --db-parameter-group-family MySQL5.6 \
  --description "My new parameter group"

```

输出：

```
{
```

```

    "DBParameterGroup": {
      "DBParameterGroupName": "mydbparametergroup",
      "DBParameterGroupFamily": "mysql5.6",
      "Description": "My new parameter group",
      "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:mydbparametergroup"
    }
  }
}

```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[创建数据库参数组](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考ParameterGroup》中的[CreateDB](#)。

## create-db-proxy-endpoint

以下代码示例演示了如何使用 create-db-proxy-endpoint。

### AWS CLI

为 RDS 数据库创建数据库代理终端节点

以下create-db-proxy-endpoint示例创建了一个数据库代理终端节点。

```

aws rds create-db-proxy-endpoint \
  --db-proxy-name proxyExample \
  --db-proxy-endpoint-name "proxyep1" \
  --vpc-subnet-ids subnetgroup1 subnetgroup2

```

输出：

```

{
  "DBProxyEndpoint": {
    "DBProxyEndpointName": "proxyep1",
    "DBProxyEndpointArn": "arn:aws:rds:us-east-1:123456789012:db-proxy-
endpoint:prx-endpoint-0123a01b12345c0ab",
    "DBProxyName": "proxyExample",
    "Status": "creating",
    "VpcId": "vpc-1234567",
    "VpcSecurityGroupIds": [
      "sg-1234",
      "sg-5678"
    ],
    "VpcSubnetIds": [

```

```

        "subnetgroup1",
        "subnetgroup2"
    ],
    "Endpoint": "proxyep1.endpoint.proxy-ab0cd1efghij.us-
east-1.rds.amazonaws.com",
    "CreateDate": "2023-04-05T16:09:33.452000+00:00",
    "TargetRole": "READ_WRITE",
    "IsDefault": false
  }
}

```

有关更多信息，请参阅 Amazon RDS 用户指南中的[创建代理终端节点](#)和 Amazon Aurora 用户指南中的[创建代理终端节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateDbProxyEndpoint](#)中的。

## create-db-proxy

以下代码示例演示了如何使用 create-db-proxy。

### AWS CLI

为 RDS 数据库创建数据库代理

以下 create-db-proxy 示例创建了一个数据库代理。

```

aws rds create-db-proxy \
  --db-proxy-name proxyExample \
  --engine-family MYSQL \
  --auth
  Description="proxydescription1",AuthScheme="SECRETS",SecretArn="arn:aws:secretsmanager:us-
west-2:123456789123:secret:secretName-1234f",IAMAuth="DISABLED",ClientPasswordAuthType="MYSQL"
  \
  --role-arn arn:aws:iam::123456789123:role/ProxyRole \
  --vpc-subnet-ids subnetgroup1 subnetgroup2

```

输出：

```

{
  "DBProxy": {
    "DBProxyName": "proxyExample",
    "DBProxyArn": "arn:aws:rds:us-east-1:123456789012:db-
proxy:prx-0123a01b12345c0ab",

```

```

    "EngineFamily": "MYSQL",
    "VpcId": "vpc-1234567",
    "VpcSecuritytGroupIds": [
        "sg-1234",
        "sg-5678",
        "sg-9101"
    ],
    "VpcSubnetIds": [
        "subnetgroup1",
        "subnetgroup2"
    ],
    "Auth": "[
        {
            "Description": "proxydescription1",
            "AuthScheme": "SECRETS",
            "SecretArn": "arn:aws:secretsmanager:us-
west-2:123456789123:secret:proxysecret1-Abcd1e",
            "IAMAuth": "DISABLED"
        }
    ]",
    "RoleArn": "arn:aws:iam::12345678912:role/ProxyRole",
    "Endpoint": "proxyExample.proxy-ab0cd1efghij.us-east-1.rds.amazonaws.com",
    "RequireTLS": false,
    "IdleClientTimeout": 1800,
    "DebuggingLogging": false,
    "CreateDate": "2023-04-05T16:09:33.452000+00:00",
    "UpdatedDate": "2023-04-13T01:49:38.568000+00:00"
}
}

```

有关更多信息，请参阅 Amazon [RDS 用户指南中的创建 RDS 代理](#)和亚马逊 Aurora 用户指南中的 [创建 RDS 代理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDbProxy](#)中的。

## create-db-security-group

以下代码示例演示了如何使用 create-db-security-group。

### AWS CLI

#### 创建 Amazon RDS 数据库安全组

以下`create-db-security-group`命令创建一个新的 Amazon RDS 数据库安全组：

```
aws rds create-db-security-group --db-security-group-name mysecgroup --db-security-group-description "My Test Security Group"
```

在示例中，新的数据库安全组被命名`mysecgroup`并带有描述。

输出：

```
{
  "DBSecurityGroup": {
    "OwnerId": "123456789012",
    "DBSecurityGroupName": "mysecgroup",
    "DBSecurityGroupDescription": "My Test Security Group",
    "VpcId": "vpc-a1b2c3d4",
    "EC2SecurityGroups": [],
    "IPRanges": [],
    "DBSecurityGroupArn": "arn:aws:rds:us-west-2:123456789012:secgrp:mysecgroup"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDbSecurityGroup](#)中的。

## create-db-shard-group

以下代码示例演示了如何使用 `create-db-shard-group`。

### AWS CLI

示例 1：创建 Aurora PostgreSQL 主数据库集群

以下`create-db-cluster`示例创建了一个与 Aurora Serverless v2 和 Aurora Limitless 数据库兼容的 Aurora PostgreSQL SQL 主数据库集群。

```
aws rds create-db-cluster \  
  --db-cluster-identifier my-sv2-cluster \  
  --engine aurora-postgresql \  
  --engine-version 15.2-limitless \  
  --storage-type aurora-iopt1 \  
  --serverless-v2-scaling-configuration MinCapacity=2,MaxCapacity=16 \  
  --enable-limitless-database \  
  --enable-iam-authentication
```

```
--master-username myuser \  
--master-user-password mypassword \  
--enable-cloudwatch-logs-exports postgresql
```

输出：

```
{  
  "DBCluster": {  
    "AllocatedStorage": 1,  
    "AvailabilityZones": [  
      "us-east-2b",  
      "us-east-2c",  
      "us-east-2a"  
    ],  
    "BackupRetentionPeriod": 1,  
    "DBClusterIdentifier": "my-sv2-cluster",  
    "DBClusterParameterGroup": "default.aurora-postgresql15",  
    "DBSubnetGroup": "default",  
    "Status": "creating",  
    "Endpoint": "my-sv2-cluster.cluster-cekyexample.us-  
east-2.rds.amazonaws.com",  
    "ReaderEndpoint": "my-sv2-cluster.cluster-ro-cekyexample.us-  
east-2.rds.amazonaws.com",  
    "MultiAZ": false,  
    "Engine": "aurora-postgresql",  
    "EngineVersion": "15.2-limitless",  
    "Port": 5432,  
    "MasterUsername": "myuser",  
    "PreferredBackupWindow": "06:05-06:35",  
    "PreferredMaintenanceWindow": "mon:08:25-mon:08:55",  
    "ReadReplicaIdentifiers": [],  
    "DBClusterMembers": [],  
    "VpcSecurityGroups": [  
      {  
        "VpcSecurityGroupId": "sg-#####",  
        "Status": "active"  
      }  
    ],  
    "HostedZoneId": "Z2XHWR1EXAMPLE",  
    "StorageEncrypted": false,  
    "DbClusterResourceId": "cluster-XYEDT6ML6FHIXH4Q2J1EXAMPLE",  
    "DBClusterArn": "arn:aws:rds:us-east-2:123456789012:cluster:my-sv2-cluster",  
    "AssociatedRoles": [],
```

```

    "IAMDatabaseAuthenticationEnabled": false,
    "ClusterCreateTime": "2024-02-19T16:24:07.771000+00:00",
    "EnabledCloudwatchLogsExports": [
      "postgresql"
    ],
    "EngineMode": "provisioned",
    "DeletionProtection": false,
    "HttpEndpointEnabled": false,
    "CopyTagsToSnapshot": false,
    "CrossAccountClone": false,
    "DomainMemberships": [],
    "TagList": [],
    "StorageType": "aurora-iopt1",
    "AutoMinorVersionUpgrade": true,
    "ServerlessV2ScalingConfiguration": {
      "MinCapacity": 2.0,
      "MaxCapacity": 16.0
    },
    "NetworkType": "IPV4",
    "IOOptimizedNextAllowedModificationTime":
"2024-03-21T16:24:07.781000+00:00",
    "LimitlessDatabase": {
      "Status": "not-in-use",
      "MinRequiredACU": 96.0
    }
  }
}

```

## 示例 2：创建主（写入器）数据库实例

以下 `create-db-instance` 示例创建了一个 Aurora Serverless v2 主（写入器）数据库实例。当您使用控制台创建数据库集群时，Amazon RDS 会自动为您的数据库集群创建写入数据库实例。但是，当您使用 AWS CLI 创建数据库集群时，必须使用 `CL create-db-instance AWS I` 命令为数据库集群显式创建写入器数据库实例。

```

aws rds create-db-instance \
  --db-instance-identifier my-sv2-instance \
  --db-cluster-identifier my-sv2-cluster \
  --engine aurora-postgresql \
  --db-instance-class db.serverless

```

输出：

```
{
  "DBInstance": {
    "DBInstanceIdentifier": "my-sv2-instance",
    "DBInstanceClass": "db.serverless",
    "Engine": "aurora-postgresql",
    "DBInstanceStatus": "creating",
    "MasterUsername": "myuser",
    "AllocatedStorage": 1,
    "PreferredBackupWindow": "06:05-06:35",
    "BackupRetentionPeriod": 1,
    "DBSecurityGroups": [],
    "VpcSecurityGroups": [
      {
        "VpcSecurityGroupId": "sg-#####",
        "Status": "active"
      }
    ],
    "DBParameterGroups": [
      {
        "DBParameterGroupName": "default.aurora-postgresql15",
        "ParameterApplyStatus": "in-sync"
      }
    ],
    "DBSubnetGroup": {
      "DBSubnetGroupName": "default",
      "DBSubnetGroupDescription": "default",
      "VpcId": "vpc-#####",
      "SubnetGroupStatus": "Complete",
      "Subnets": [
        {
          "SubnetIdentifier": "subnet-#####",
          "SubnetAvailabilityZone": {
            "Name": "us-east-2c"
          },
          "SubnetOutpost": {},
          "SubnetStatus": "Active"
        },
        {
          "SubnetIdentifier": "subnet-#####",
          "SubnetAvailabilityZone": {
            "Name": "us-east-2a"
          },
          "SubnetOutpost": {}
        }
      ]
    }
  }
}
```



```
        "SubnetStatus": "Active"
      },
      {
        "SubnetIdentifier": "subnet-#####",
        "SubnetAvailabilityZone": {
          "Name": "us-east-2b"
        },
        "SubnetOutpost": {},
        "SubnetStatus": "Active"
      }
    ]
  },
  "PreferredMaintenanceWindow": "fri:09:01-fri:09:31",
  "PendingModifiedValues": {
    "PendingCloudwatchLogsExports": {
      "LogTypesToEnable": [
        "postgresql"
      ]
    }
  },
  "MultiAZ": false,
  "EngineVersion": "15.2-limitless",
  "AutoMinorVersionUpgrade": true,
  "ReadReplicaDBInstanceIdentifiers": [],
  "LicenseModel": "postgresql-license",
  "OptionGroupMemberships": [
    {
      "OptionGroupName": "default:aurora-postgresql-15",
      "Status": "in-sync"
    }
  ],
  "PubliclyAccessible": false,
  "StorageType": "aurora-iopt1",
  "DbInstancePort": 0,
  "DBClusterIdentifier": "my-sv2-cluster",
  "StorageEncrypted": false,
  "DbiResourceId": "db-BIQTE3B3K3RM7M74SK5EXAMPLE",
  "CACertificateIdentifier": "rds-ca-rsa2048-g1",
  "DomainMemberships": [],
  "CopyTagsToSnapshot": false,
  "MonitoringInterval": 0,
  "PromotionTier": 1,
  "DBInstanceArn": "arn:aws:rds:us-east-2:123456789012:db:my-sv2-instance",
  "IAMDatabaseAuthenticationEnabled": false,
```

```

    "PerformanceInsightsEnabled": false,
    "DeletionProtection": false,
    "AssociatedRoles": [],
    "TagList": [],
    "CustomerOwnedIpEnabled": false,
    "BackupTarget": "region",
    "NetworkType": "IPV4",
    "StorageThroughput": 0,
    "CertificateDetails": {
      "CAIdentifier": "rds-ca-rsa2048-g1"
    },
    "DedicatedLogVolume": false
  }
}

```

### 示例 3：创建数据库分片组

以下 `create-db-shard-group` 示例在您的 Aurora PostgreSQL 主数据库集群中创建了一个数据库分片组。

```

aws rds create-db-shard-group \
  --db-shard-group-identifier my-db-shard-group \
  --db-cluster-identifier my-sv2-cluster \
  --max-acu 768

```

输出：

```

{
  "DBShardGroupResourceId": "shardgroup-a6e3a0226aa243e2ac6c7a1234567890",
  "DBShardGroupIdentifier": "my-db-shard-group",
  "DBClusterIdentifier": "my-sv2-cluster",
  "MaxACU": 768.0,
  "ComputeRedundancy": 0,
  "Status": "creating",
  "PubliclyAccessible": false,
  "Endpoint": "my-sv2-cluster.limitless-cekyexample.us-east-2.rds.amazonaws.com"
}

```

有关更多信息，请参阅亚马逊 Aurora 用户指南中的 [使用 Aurora Serverless v2](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDbShardGroup](#) 中的。

## create-db-snapshot

以下代码示例演示了如何使用 create-db-snapshot。

### AWS CLI

#### 创建数据库快照

以下 create-db-snapshot 示例创建数据库快照。

```
aws rds create-db-snapshot \  
  --db-instance-identifier database-mysql \  
  --db-snapshot-identifier mydbsnapshot
```

输出：

```
{  
  "DBSnapshot": {  
    "DBSnapshotIdentifier": "mydbsnapshot",  
    "DBInstanceIdentifier": "database-mysql",  
    "Engine": "mysql",  
    "AllocatedStorage": 100,  
    "Status": "creating",  
    "Port": 3306,  
    "AvailabilityZone": "us-east-1b",  
    "VpcId": "vpc-6594f31c",  
    "InstanceCreateTime": "2019-04-30T15:45:53.663Z",  
    "MasterUsername": "admin",  
    "EngineVersion": "5.6.40",  
    "LicenseModel": "general-public-license",  
    "SnapshotType": "manual",  
    "Iops": 1000,  
    "OptionGroupName": "default:mysql-5-6",  
    "PercentProgress": 0,  
    "StorageType": "io1",  
    "Encrypted": true,  
    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE",  
    "DBSnapshotArn": "arn:aws:rds:us-east-1:123456789012:snapshot:mydbsnapshot",  
    "IAMDatabaseAuthenticationEnabled": false,  
    "ProcessorFeatures": [],  
    "DbiResourceId": "db-AKIAIOSFODNN7EXAMPLE"  
  }  
}
```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[创建数据库快照](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [CreateDBSnapshot](#)。

## create-db-subnet-group

以下代码示例演示了如何使用 create-db-subnet-group。

### AWS CLI

#### 创建数据库子网组

以下 create-db-subnet-group 示例 mysubnetgroup 使用现有子网创建名为的数据库子网组。

```
aws rds create-db-subnet-group \  
  --db-subnet-group-name mysubnetgroup \  
  --db-subnet-group-description "test DB subnet group" \  
  --subnet-ids  
  ["subnet-0a1dc4e1a6f123456","subnet-070dd7ecb3aaaaaaaa","subnet-00f5b198bc0abcdef"]'
```

输出：

```
{  
  "DBSubnetGroup": {  
    "DBSubnetGroupName": "mysubnetgroup",  
    "DBSubnetGroupDescription": "test DB subnet group",  
    "VpcId": "vpc-0f08e7610a1b2c3d4",  
    "SubnetGroupStatus": "Complete",  
    "Subnets": [  
      {  
        "SubnetIdentifier": "subnet-070dd7ecb3aaaaaaaa",  
        "SubnetAvailabilityZone": {  
          "Name": "us-west-2b"  
        },  
        "SubnetStatus": "Active"  
      },  
      {  
        "SubnetIdentifier": "subnet-00f5b198bc0abcdef",  
        "SubnetAvailabilityZone": {  
          "Name": "us-west-2d"  
        },  
        "SubnetStatus": "Active"  
      },  
    ],  
  },  
}
```

```

    {
      "SubnetIdentifier": "subnet-0a1dc4e1a6f123456",
      "SubnetAvailabilityZone": {
        "Name": "us-west-2b"
      },
      "SubnetStatus": "Active"
    }
  ],
  "DBSubnetGroupArn": "arn:aws:rds:us-
west-2:0123456789012:subgrp:mysubnetgroup"
}
}

```

有关更多信息，请参阅 Amazon RDS 用户指南 [中的在 VPC 中创建数据库实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDbSubnetGroup](#) 中的。

## create-event-subscription

以下代码示例演示了如何使用 create-event-subscription。

### AWS CLI

#### 创建活动订阅

以下 create-event-subscription 示例为当前 AWS 账户中的数据库实例创建备份和恢复事件的订阅。通知将发送到由指定的 Amazon 简单通知服务主题 --sns-topic-arn。

```

aws rds create-event-subscription \
  --subscription-name my-instance-events \
  --source-type db-instance \
  --event-categories ["backup","recovery"] \
  --sns-topic-arn arn:aws:sns:us-east-1:123456789012:interesting-events

```

输出：

```

{
  "EventSubscription": {
    "Status": "creating",
    "CustSubscriptionId": "my-instance-events",
    "SubscriptionCreationTime": "Tue Jul 31 23:22:01 UTC 2018",
    "EventCategoriesList": [
      "backup",

```

```
        "recovery"
      ],
      "SnsTopicArn": "arn:aws:sns:us-east-1:123456789012:interesting-events",
      "CustomerAwsId": "123456789012",
      "EventSubscriptionArn": "arn:aws:rds:us-east-1:123456789012:es:my-instance-
events",
      "SourceType": "db-instance",
      "Enabled": true
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateEventSubscription](#)中的。

## create-global-cluster

以下代码示例演示了如何使用 create-global-cluster。

### AWS CLI

#### 创建全局数据库集群

以下create-global-cluster示例创建了一个新的兼容 Aurora MySQL 的全局数据库集群。

```
aws rds create-global-cluster \  
  --global-cluster-identifier myglobalcluster \  
  --engine aurora-mysql
```

输出：

```
{  
  "GlobalCluster": {  
    "GlobalClusterIdentifier": "myglobalcluster",  
    "GlobalClusterResourceId": "cluster-f0e523bfe07aabb",  
    "GlobalClusterArn": "arn:aws:rds::123456789012:global-  
cluster:myglobalcluster",  
    "Status": "available",  
    "Engine": "aurora-mysql",  
    "EngineVersion": "5.7.mysql_aurora.2.07.2",  
    "StorageEncrypted": false,  
    "DeletionProtection": false,  
    "GlobalClusterMembers": []  
  }  
}
```

```
}
```

有关更多信息，请参阅亚马逊 Aurora 用户指南中的创建 Aurora [全球数据库](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateGlobalCluster](#)中的。

## create-option-group

以下代码示例演示了如何使用 create-option-group。

### AWS CLI

#### 创建 Amazon RDS 选项组

以下create-option-group命令为Oracle Enterprise Edition版本创建新的 Amazon RDS 选项组11.2`， is named ``MyOptionGroup并包含描述。

```
aws rds create-option-group \  
  --option-group-name MyOptionGroup \  
  --engine-name oracle-ee \  
  --major-engine-version 11.2 \  
  --option-group-description "Oracle Database Manager Database Control"
```

输出：

```
{  
  "OptionGroup": {  
    "OptionGroupName": "myoptiongroup",  
    "OptionGroupDescription": "Oracle Database Manager Database Control",  
    "EngineName": "oracle-ee",  
    "MajorEngineVersion": "11.2",  
    "Options": [],  
    "AllowsVpcAndNonVpcInstanceMemberships": true,  
    "OptionGroupArn": "arn:aws:rds:us-west-2:123456789012:og:myoptiongroup"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateOptionGroup](#)中的。

## delete-blue-green-deployment

以下代码示例演示了如何使用 delete-blue-green-deployment。

## AWS CLI

示例 1：在绿色环境中删除 RDS for MySQL 数据库实例的资源

以下delete-blue-green-deployment示例在绿色环境中删除 RDS for MySQL 数据库实例的资源。

```
aws rds delete-blue-green-deployment \  
  --blue-green-deployment-identifier bgd-v53303651eexfake \  
  --delete-target
```

输出：

```
{  
  "BlueGreenDeployment": {  
    "BlueGreenDeploymentIdentifier": "bgd-v53303651eexfake",  
    "BlueGreenDeploymentName": "bgd-cli-test-instance",  
    "Source": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance",  
    "Target": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-green-rkfbpe",  
    "SwitchoverDetails": [  
      {  
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance",  
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-green-rkfbpe",  
        "Status": "AVAILABLE"  
      },  
      {  
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-replica-1",  
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-replica-1-green-j382ha",  
        "Status": "AVAILABLE"  
      },  
      {  
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-replica-2",  
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-replica-2-green-ejv4ao",  
        "Status": "AVAILABLE"  
      },  
      {
```



```

        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-3",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-3-green-vlpz3t",
        "Status": "AVAILABLE"
    }
],
"Tasks": [
    {
        "Name": "CREATING_READ_REPLICA_OF_SOURCE",
        "Status": "COMPLETED"
    },
    {
        "Name": "DB_ENGINE_VERSION_UPGRADE",
        "Status": "COMPLETED"
    },
    {
        "Name": "CONFIGURE_BACKUPS",
        "Status": "COMPLETED"
    },
    {
        "Name": "CREATING_TOPOLOGY_OF_SOURCE",
        "Status": "COMPLETED"
    }
],
"Status": "DELETING",
"CreateTime": "2022-02-25T21:18:51.183000+00:00",
>DeleteTime": "2022-02-25T22:25:31.331000+00:00"
}
}

```

有关更多信息，请参阅 Amazon RDS 用户指南中的[删除蓝/绿部署](#)。

示例 2：在绿色环境中删除 Aurora MySQL 数据库集群的资源

以下 delete-blue-green-deployment 示例删除了 Aurora MySQL 数据库集群在绿色环境中的资源。

```

aws rds delete-blue-green-deployment \
  --blue-green-deployment-identifier bgd-wi89nwzglccsfake \
  --delete-target

```

输出：

```
{
  "BlueGreenDeployment": {
    "BlueGreenDeploymentIdentifier": "bgd-wi89nwzglccsfake",
    "BlueGreenDeploymentName": "my-blue-green-deployment",
    "Source": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-
cluster",
    "Target": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-
cluster-green-3rnukl",
    "SwitchoverDetails": [
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster:my-
aurora-mysql-cluster",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster:my-
aurora-mysql-cluster-green-3rnukl",
        "Status": "AVAILABLE"
      },
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-
mysql-cluster-1",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-
mysql-cluster-1-green-gpmaxf",
        "Status": "AVAILABLE"
      },
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-
mysql-cluster-2",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-
mysql-cluster-2-green-j2oajq",
        "Status": "AVAILABLE"
      },
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-
mysql-cluster-3",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-
mysql-cluster-3-green-mkxies",
        "Status": "AVAILABLE"
      },
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster-
endpoint:my-excluded-member-endpoint",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster-
endpoint:my-excluded-member-endpoint-green-4sqjrq",
        "Status": "AVAILABLE"
      }
    ]
  }
}
```

```

    },
    {
      "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster-
endpoint:my-reader-endpoint",
      "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster-
endpoint:my-reader-endpoint-green-gwzlg",
      "Status": "AVAILABLE"
    }
  ],
  "Tasks": [
    {
      "Name": "CREATING_READ_REPLICA_OF_SOURCE",
      "Status": "COMPLETED"
    },
    {
      "Name": "DB_ENGINE_VERSION_UPGRADE",
      "Status": "COMPLETED"
    },
    {
      "Name": "CREATE_DB_INSTANCES_FOR_CLUSTER",
      "Status": "COMPLETED"
    },
    {
      "Name": "CREATE_CUSTOM_ENDPOINTS",
      "Status": "COMPLETED"
    }
  ],
  "Status": "DELETING",
  "CreateTime": "2022-02-25T21:12:00.288000+00:00",
  "DeleteTime": "2022-02-25T22:29:11.336000+00:00"
}
}

```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[删除蓝/绿部署](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteBlueGreenDeployment](#)中的。

## delete-db-cluster-endpoint

以下代码示例演示了如何使用 delete-db-cluster-endpoint。

### AWS CLI

#### 删除自定义数据库集群终端节点

以下delete-db-cluster-endpoint示例删除了指定的自定义数据库集群终端节点。

```
aws rds delete-db-cluster-endpoint \  
  --db-cluster-endpoint-identifier mycustomendpoint
```

输出：

```
{  
  "DBClusterEndpointIdentifier": "mycustomendpoint",  
  "DBClusterIdentifier": "mydbcluster",  
  "DBClusterEndpointResourceIdentifier": "cluster-endpoint-ANPAJ4AE5446DAEXAMPLE",  
  "Endpoint": "mycustomendpoint.cluster-custom-cnpeexample.us-  
east-1.rds.amazonaws.com",  
  "Status": "deleting",  
  "EndpointType": "CUSTOM",  
  "CustomEndpointType": "READER",  
  "StaticMembers": [  
    "dbinstance1",  
    "dbinstance2",  
    "dbinstance3"  
  ],  
  "ExcludedMembers": [],  
  "DBClusterEndpointArn": "arn:aws:rds:us-east-1:123456789012:cluster-  
endpoint:mycustomendpoint"  
}
```

有关更多信息，请参阅[亚马逊 Aurora 用户指南中的亚马逊 Aurora 连接管理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDbClusterEndpoint](#)中的。

## delete-db-cluster-parameter-group

以下代码示例演示了如何使用 delete-db-cluster-parameter-group。

### AWS CLI

删除数据库集群参数组

以下delete-db-cluster-parameter-group示例删除了指定的数据库集群参数组。

```
aws rds delete-db-cluster-parameter-group \  
  --parameter-group-name myparametergroup
```

```
--db-cluster-parameter-group-name mydbclusterparametergroup
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Aurora 用户指南中的使用[数据库参数组](#)和[数据库集群参数组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteDbClusterParameterGroup](#)中的。

## delete-db-cluster-snapshot

以下代码示例演示了如何使用 delete-db-cluster-snapshot。

### AWS CLI

#### 删除数据库集群快照

以下 delete-db-cluster-snapshot 示例删除了指定的数据库集群快照。

```
aws rds delete-db-cluster-snapshot \  
  --db-cluster-snapshot-identifier mydbclustersnapshot
```

输出：

```
{  
  "DBClusterSnapshot": {  
    "AvailabilityZones": [  
      "us-east-1a",  
      "us-east-1b",  
      "us-east-1e"  
    ],  
    "DBClusterSnapshotIdentifier": "mydbclustersnapshot",  
    "DBClusterIdentifier": "mydbcluster",  
    "SnapshotCreateTime": "2019-06-18T21:21:00.469Z",  
    "Engine": "aurora-mysql",  
    "AllocatedStorage": 0,  
    "Status": "available",  
    "Port": 0,  
    "VpcId": "vpc-6594f31c",  
    "ClusterCreateTime": "2019-04-15T14:18:42.785Z",  
    "MasterUsername": "myadmin",  
    "EngineVersion": "5.7.mysql_aurora.2.04.2",  
    "LicenseModel": "aurora-mysql",
```

```
    "SnapshotType": "manual",
    "PercentProgress": 100,
    "StorageEncrypted": true,
    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE",
    "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:123456789012:cluster-
snapshot:mydbclustersnapshot",
    "IAMDatabaseAuthenticationEnabled": false
  }
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[删除快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDbClusterSnapshot](#)中的。

## delete-db-cluster

以下代码示例演示了如何使用 delete-db-cluster。

### AWS CLI

示例 1：删除数据库集群中的数据库实例

以下delete-db-instance示例删除数据库集群中的最后一个数据库实例。如果数据库集群包含未处于删除状态的数据库实例，则无法将其删除。删除数据库集群中的数据库实例时，您无法拍摄最终快照。

```
aws rds delete-db-instance \
  --db-instance-identifier database-3
```

输出：

```
{
  "DBInstance": {
    "DBInstanceIdentifier": "database-3",
    "DBInstanceClass": "db.r4.large",
    "Engine": "aurora-postgresql",
    "DBInstanceStatus": "deleting",

    ...output omitted...
  }
}
```

有关更多信息，请参阅 [Amazon Aurora 用户指南中的删除 Aurora 数据库集群中的数据库实例](#)。

## 示例 2：删除数据库集群

以下 `delete-db-cluster` 示例删除名为 `mycluster` 的数据库集群并拍摄名为 `mycluster-final-snapshot` 的最终快照。拍摄快照时，数据库集群的状态为可用。要跟踪删除进度，请使用 `describe-db-clusters` CLI 命令。

```
aws rds delete-db-cluster \
  --db-cluster-identifier mycluster \
  --no-skip-final-snapshot \
  --final-db-snapshot-identifier mycluster-final-snapshot
```

输出：

```
{
  "DBCluster": {
    "AllocatedStorage": 20,
    "AvailabilityZones": [
      "eu-central-1b",
      "eu-central-1c",
      "eu-central-1a"
    ],
    "BackupRetentionPeriod": 7,
    "DBClusterIdentifier": "mycluster",
    "DBClusterParameterGroup": "default.aurora-postgresql10",
    "DBSubnetGroup": "default-vpc-aa11bb22",
    "Status": "available",

    ...output omitted...
  }
}
```

有关更多信息，请参阅 [Amazon Aurora 用户指南中的带有单个数据库实例的 Aurora 集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteDbCluster](#) 中的。

## delete-db-instance-automated-backup

以下代码示例演示了如何使用 `delete-db-instance-automated-backup`。

## AWS CLI

从区域中删除已复制的自动备份

以下delete-db-instance-automated-backup示例删除了具有指定 Amazon 资源名称 (ARN) 的自动备份。

```
aws rds delete-db-instance-automated-backup \
  --db-instance-automated-backups-arn "arn:aws:rds:us-west-2:123456789012:auto-
  backup:ab-jkib2gfgq5rv7replzadausbrktni2bn4example"
```

输出：

```
{
  "DBInstanceAutomatedBackup": {
    "DBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:new-orcl-db",
    "DbiResourceId": "db-JKIB2GFQ5RV7REPLZA4EXAMPLE",
    "Region": "us-east-1",
    "DBInstanceIdentifier": "new-orcl-db",
    "RestoreWindow": {},
    "AllocatedStorage": 20,
    "Status": "deleting",
    "Port": 1521,
    "AvailabilityZone": "us-east-1b",
    "VpcId": "vpc-#####",
    "InstanceCreateTime": "2020-12-04T15:28:31Z",
    "MasterUsername": "admin",
    "Engine": "oracle-se2",
    "EngineVersion": "12.1.0.2.v21",
    "LicenseModel": "bring-your-own-license",
    "OptionGroupName": "default:oracle-se2-12-1",
    "Encrypted": false,
    "StorageType": "gp2",
    "IAMDatabaseAuthenticationEnabled": false,
    "BackupRetentionPeriod": 7,
    "DBInstanceAutomatedBackupsArn": "arn:aws:rds:us-west-2:123456789012:auto-
    backup:ab-jkib2gfgq5rv7replzadausbrktni2bn4example"
  }
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[删除复制的备份](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDbInstanceAutomatedBackup](#)中的。



## delete-db-instance

以下代码示例演示了如何使用 delete-db-instance。

### AWS CLI

#### 删除数据库实例

以下 delete-db-instance 示例在创建名为 test-instance-final-snap 的最终数据库快照后删除指定的数据库实例。

```
aws rds delete-db-instance \  
  --db-instance-identifier test-instance \  
  --final-db-snapshot-identifier test-instance-final-snap
```

输出：

```
{  
  "DBInstance": {  
    "DBInstanceIdentifier": "test-instance",  
    "DBInstanceStatus": "deleting",  
    ...some output truncated...  
  }  
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DeleteDBInstance](#)。

## delete-db-parameter-group

以下代码示例演示了如何使用 delete-db-parameter-group。

### AWS CLI

#### 删除数据库参数组

以下 command 示例删除一个数据库参数组。

```
aws rds delete-db-parameter-group \  
  --db-parameter-group-name mydbparametergroup
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon RDS 用户指南》中的[使用数据库参数组](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》ParameterGroup 中的 [deletedB](#)。

## delete-db-proxy-endpoint

以下代码示例演示了如何使用 delete-db-proxy-endpoint。

### AWS CLI

删除 RDS 数据库的数据库代理终端节点

以下 delete-db-proxy-endpoint 示例删除目标数据库的数据库代理终端节点。

```
aws rds delete-db-proxy-endpoint \  
  --db-proxy-endpoint-name proxyEP1
```

输出：

```
{  
  "DBProxyEndpoint":  
    {  
      "DBProxyEndpointName": "proxyEP1",  
      "DBProxyEndpointArn": "arn:aws:rds:us-east-1:123456789012:db-proxy-  
endpoint:prx-endpoint-0123a01b12345c0ab",  
      "DBProxyName": "proxyExample",  
      "Status": "deleting",  
      "VpcId": "vpc-1234567",  
      "VpcSecurityGroupIds": [  
        "sg-1234",  
        "sg-5678"  
      ],  
      "VpcSubnetIds": [  
        "subnetgroup1",  
        "subnetgroup2"  
      ],  
      "Endpoint": "proxyEP1.endpoint.proxy-ab0cd1efghij.us-  
east-1.rds.amazonaws.com",  
      "CreateDate": "2023-04-13T01:49:38.568000+00:00",  
      "TargetRole": "READ_ONLY",  
      "IsDefault": false  
    }  
}
```

```
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[删除代理终端节点](#)和 Amazon Aurora 用户指南中的[删除代理终端节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteDbProxyEndpoint](#)中的。

## delete-db-proxy

以下代码示例演示了如何使用 delete-db-proxy。

### AWS CLI

删除 RDS 数据库的数据库代理

以下 delete-db-proxy 示例删除数据库代理。

```
aws rds delete-db-proxy \  
  --db-proxy-name proxyExample
```

输出：

```
{  
  "DBProxy":  
  {  
    "DBProxyName": "proxyExample",  
    "DBProxyArn": "arn:aws:rds:us-east-1:123456789012:db-  
proxy:prx-0123a01b12345c0ab",  
    "Status": "deleting",  
    "EngineFamily": "PostgreSQL",  
    "VpcId": "vpc-1234567",  
    "VpcSecurityGroupIds": [  
      "sg-1234",  
      "sg-5678"  
    ],  
    "VpcSubnetIds": [  
      "subnetgroup1",  
      "subnetgroup2"  
    ],  
    "Auth": "[  
      {  
        "Description": "proxydescription`"
```

```
        "AuthScheme": "SECRETS",
        "SecretArn": "arn:aws:secretsmanager:us-
west-2:123456789123:secret:proxysecret1-Abcd1e",
        "IAMAuth": "DISABLED"
    } ],
    "RoleArn": "arn:aws:iam::12345678912:role/ProxyPostgreSQLRole",
    "Endpoint": "proxyExample.proxy-ab0cd1efghij.us-
east-1.rds.amazonaws.com",
    "RequireTLS": false,
    "IdleClientTimeout": 1800,
    "DebuggingLogging": false,
    "CreateDate": "2023-04-05T16:09:33.452000+00:00",
    "UpdateDate": "2023-04-13T01:49:38.568000+00:00"
}
}
```

有关更多信息，请参阅 Amazon [RDS](#) 用户指南中的[删除 RDS 代理](#)和亚马逊 Aurora 用户指南中的[删除 RDS 代理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDbProxy](#)中的。

## delete-db-security-group

以下代码示例演示了如何使用 delete-db-security-group。

### AWS CLI

#### 删除数据库安全组

以下delete-db-security-group示例删除名为的数据库安全组mysecuritygroup。

```
aws rds delete-db-security-group \
  --db-security-group-name mysecuritygroup
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon RDS 用户指南中的[使用数据库安全组 \( EC2-Classical 平台 \)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDbSecurityGroup](#)中的。

## delete-db-shard-group

以下代码示例演示了如何使用 delete-db-shard-group。

## AWS CLI

### 示例 1：删除数据库分片组失败

以下delete-db-shard-group示例显示了在删除所有数据库和架构之前尝试删除数据库分片组时发生的错误。

```
aws rds delete-db-shard-group \  
  --db-shard-group-identifier limitless-test-shard-grp
```

#### 输出：

```
An error occurred (InvalidDBShardGroupState) when calling the DeleteDBShardGroup  
operation: Unable to delete the DB shard group limitless-test-db-shard-group.  
Delete all of your Limitless Database databases and schemas, then try again.
```

### 示例 2：成功删除数据库分片组

以下delete-db-shard-group示例在您删除所有数据库和架构（包括架构）后删除数据库分片组。public

```
aws rds delete-db-shard-group \  
  --db-shard-group-identifier limitless-test-shard-grp
```

#### 输出：

```
{  
  "DBShardGroupResourceId": "shardgroup-7bb446329da94788b3f957746example",  
  "DBShardGroupIdentifier": "limitless-test-shard-grp",  
  "DBClusterIdentifier": "limitless-test-cluster",  
  "MaxACU": 768.0,  
  "ComputeRedundancy": 0,  
  "Status": "deleting",  
  "PubliclyAccessible": true,  
  "Endpoint": "limitless-test-cluster.limitless-cekyexample.us-  
east-2.rds.amazonaws.com"  
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的删除 Aurora [数据库集群和数据库实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDbShardGroup](#)中的。

## delete-db-snapshot

以下代码示例演示了如何使用 delete-db-snapshot。

### AWS CLI

#### 删除数据库快照

以下delete-db-snapshot示例删除了指定的数据库快照。

```
aws rds delete-db-snapshot \  
  --db-snapshot-identifier mydbsnapshot
```

输出：

```
{  
  "DBSnapshot": {  
    "DBSnapshotIdentifier": "mydbsnapshot",  
    "DBInstanceIdentifier": "database-mysql",  
    "SnapshotCreateTime": "2019-06-18T22:08:40.702Z",  
    "Engine": "mysql",  
    "AllocatedStorage": 100,  
    "Status": "deleted",  
    "Port": 3306,  
    "AvailabilityZone": "us-east-1b",  
    "VpcId": "vpc-6594f31c",  
    "InstanceCreateTime": "2019-04-30T15:45:53.663Z",  
    "MasterUsername": "admin",  
    "EngineVersion": "5.6.40",  
    "LicenseModel": "general-public-license",  
    "SnapshotType": "manual",  
    "Iops": 1000,  
    "OptionGroupName": "default:mysql-5-6",  
    "PercentProgress": 100,  
    "StorageType": "io1",  
    "Encrypted": true,  
    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE",  
    "DBSnapshotArn": "arn:aws:rds:us-east-1:123456789012:snapshot:mydbsnapshot",  
    "IAMDatabaseAuthenticationEnabled": false,  
    "ProcessorFeatures": [],  
    "DbiResourceId": "db-AKIAIOSFODNN7EXAMPLE"  
  }  
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[删除快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDbSnapshot](#)中的。

## delete-db-subnet-group

以下代码示例演示了如何使用 delete-db-subnet-group。

### AWS CLI

删除数据库子网组

以下delete-db-subnet-group示例删除名为的数据库子网组mysubnetgroup。

```
aws rds delete-db-subnet-group --db-subnet-group-name mysubnetgroup
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon RDS 用户指南中的[在 VPC 中使用数据库实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDbSubnetGroup](#)中的。

## delete-event-subscription

以下代码示例演示了如何使用 delete-event-subscription。

### AWS CLI

删除活动订阅

以下delete-event-subscription示例删除了指定的事件订阅。

```
aws rds delete-event-subscription --subscription-name my-instance-events
```

输出：

```
{
  "EventSubscription": {
    "EventSubscriptionArn": "arn:aws:rds:us-east-1:123456789012:es:my-instance-events",
    "CustomerAwsId": "123456789012",
    "Enabled": false,
    "SourceIdsList": [
```

```
        "test-instance"
    ],
    "SourceType": "db-instance",
    "EventCategoriesList": [
        "backup",
        "recovery"
    ],
    "SubscriptionCreationTime": "2018-07-31 23:22:01.893",
    "CustSubscriptionId": "my-instance-events",
    "SnsTopicArn": "arn:aws:sns:us-east-1:123456789012:interesting-events",
    "Status": "deleting"
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteEventSubscription](#)中的。

## delete-global-cluster

以下代码示例演示了如何使用 delete-global-cluster。

### AWS CLI

#### 删除全局数据库集群

以下delete-global-cluster示例删除了与 Aurora MySQL 兼容的全局数据库集群。输出显示了您要删除的集群，但后续describe-global-clusters命令并未列出该数据库集群。

```
aws rds delete-global-cluster \
    --global-cluster-identifier myglobalcluster
```

输出：

```
{
  "GlobalCluster": {
    "GlobalClusterIdentifier": "myglobalcluster",
    "GlobalClusterResourceId": "cluster-f0e523bfe07aabb",
    "GlobalClusterArn": "arn:aws:rds::123456789012:global-
cluster:myglobalcluster",
    "Status": "available",
    "Engine": "aurora-mysql",
    "EngineVersion": "5.7.mysql_aurora.2.07.2",
    "StorageEncrypted": false,
```



```
        "DeletionProtection": false,  
        "GlobalClusterMembers": []  
    }  
}
```

有关更多信息，请参阅亚马逊 Aurora 用户指南中的删除 Aurora [全球数据库](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteGlobalCluster](#)中的。

## delete-option-group

以下代码示例演示了如何使用 delete-option-group。

### AWS CLI

#### 删除选项组

以下delete-option-group示例删除了指定的选项组。

```
aws rds delete-option-group \  
  --option-group-name myoptiongroup
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon RDS 用户指南中的[删除选项组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteOptionGroup](#)中的。

## deregister-db-proxy-targets

以下代码示例演示了如何使用 deregister-db-proxy-targets。

### AWS CLI

#### 从数据库目标组中取消注册数据库代理目标

以下deregister-db-proxy-targets示例删除了代理proxyExample与其目标之间的关联。

```
aws rds deregister-db-proxy-targets \  
  --db-proxy-name proxyExample \  
  --db-instance-identifiers database-1
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon [RDS](#) 用户指南中的[删除 RDS 代理](#)和亚马逊 Aurora 用户指南中的[删除 RDS 代理](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeregisterDbProxyTargets](#)中的。

## describe-account-attributes

以下代码示例演示了如何使用 describe-account-attributes。

### AWS CLI

描述账户属性

以下 describe-account-attributes 示例检索当前 AWS 账户的属性。

```
aws rds describe-account-attributes
```

输出：

```
{
  "AccountQuotas": [
    {
      "Max": 40,
      "Used": 4,
      "AccountQuotaName": "DBInstances"
    },
    {
      "Max": 40,
      "Used": 0,
      "AccountQuotaName": "ReservedDBInstances"
    },
    {
      "Max": 100000,
      "Used": 40,
      "AccountQuotaName": "AllocatedStorage"
    },
    {
      "Max": 25,
      "Used": 0,
      "AccountQuotaName": "DBSecurityGroups"
    },
    {
      "Max": 20,
```

```
    "Used": 0,  
    "AccountQuotaName": "AuthorizationsPerDBSecurityGroup"  
  },  
  {  
    "Max": 50,  
    "Used": 1,  
    "AccountQuotaName": "DBParameterGroups"  
  },  
  {  
    "Max": 100,  
    "Used": 3,  
    "AccountQuotaName": "ManualSnapshots"  
  },  
  {  
    "Max": 20,  
    "Used": 0,  
    "AccountQuotaName": "EventSubscriptions"  
  },  
  {  
    "Max": 50,  
    "Used": 1,  
    "AccountQuotaName": "DBSubnetGroups"  
  },  
  {  
    "Max": 20,  
    "Used": 1,  
    "AccountQuotaName": "OptionGroups"  
  },  
  {  
    "Max": 20,  
    "Used": 6,  
    "AccountQuotaName": "SubnetsPerDBSubnetGroup"  
  },  
  {  
    "Max": 5,  
    "Used": 0,  
    "AccountQuotaName": "ReadReplicasPerMaster"  
  },  
  {  
    "Max": 40,  
    "Used": 1,  
    "AccountQuotaName": "DBClusters"  
  },  
  {
```

```

        "Max": 50,
        "Used": 0,
        "AccountQuotaName": "DBClusterParameterGroups"
    },
    {
        "Max": 5,
        "Used": 0,
        "AccountQuotaName": "DBClusterRoles"
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAccountAttributes](#)中的。

## describe-blue-green-deployments

以下代码示例演示了如何使用 describe-blue-green-deployments。

### AWS CLI

示例 1：描述 RDS 数据库实例在创建完成后的蓝/绿部署

以下describe-blue-green-deployment示例在创建完成后检索蓝/绿部署的详细信息。

```

aws rds describe-blue-green-deployments \
  --blue-green-deployment-identifier bgd-v53303651eexfake

```

输出：

```

{
  "BlueGreenDeployments": [
    {
      "BlueGreenDeploymentIdentifier": "bgd-v53303651eexfake",
      "BlueGreenDeploymentName": "bgd-cli-test-instance",
      "Source": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance",
      "Target": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-green-
rkfbpe",
      "SwitchoverDetails": [
        {
          "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance",
          "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-green-rkfbpe",

```

```
        "Status": "AVAILABLE"
      },
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-1",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-1-green-j382ha",
        "Status": "AVAILABLE"
      },
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-2",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-2-green-ejv4ao",
        "Status": "AVAILABLE"
      },
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-3",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-3-green-vlpz3t",
        "Status": "AVAILABLE"
      }
    ],
    "Tasks": [
      {
        "Name": "CREATING_READ_REPLICA_OF_SOURCE",
        "Status": "COMPLETED"
      },
      {
        "Name": "DB_ENGINE_VERSION_UPGRADE",
        "Status": "COMPLETED"
      },
      {
        "Name": "CONFIGURE_BACKUPS",
        "Status": "COMPLETED"
      },
      {
        "Name": "CREATING_TOPOLOGY_OF_SOURCE",
        "Status": "COMPLETED"
      }
    ],
    "Status": "AVAILABLE",
    "CreateTime": "2022-02-25T21:18:51.183000+00:00"
```

```
    }
  ]
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[查看蓝/绿部署](#)。

示例 2：描述 Aurora MySQL 数据库集群的蓝/绿部署

以下 describe-blue-green-deployment 示例检索蓝/绿部署的详细信息。

```
aws rds describe-blue-green-deployments \
  --blue-green-deployment-identifier bgd-wi89nwzglccsfake
```

输出：

```
{
  "BlueGreenDeployments": [
    {
      "BlueGreenDeploymentIdentifier": "bgd-wi89nwzglccsfake",
      "BlueGreenDeploymentName": "my-blue-green-deployment",
      "Source": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster",
      "Target": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster-green-3rnukl",
      "SwitchoverDetails": [
        {
          "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster",
          "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster-green-3rnukl",
          "Status": "AVAILABLE"
        },
        {
          "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-1",
          "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-1-green-gpmaxf",
          "Status": "AVAILABLE"
        },
        {
          "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-2",
```

```

        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-2-green-j2oajq",
        "Status": "AVAILABLE"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-3",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-3-green-mkxies",
        "Status": "AVAILABLE"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-excluded-member-endpoint",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-excluded-member-endpoint-green-4sqjrq",
        "Status": "AVAILABLE"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-reader-endpoint",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-reader-endpoint-green-gwwzlg",
        "Status": "AVAILABLE"
    }
],
"Tasks": [
    {
        "Name": "CREATING_READ_REPLICA_OF_SOURCE",
        "Status": "COMPLETED"
    },
    {
        "Name": "DB_ENGINE_VERSION_UPGRADE",
        "Status": "COMPLETED"
    },
    {
        "Name": "CREATE_DB_INSTANCES_FOR_CLUSTER",
        "Status": "COMPLETED"
    },
    {
        "Name": "CREATE_CUSTOM_ENDPOINTS",
        "Status": "COMPLETED"
    }
],

```

```

        "Status": "AVAILABLE",
        "CreateTime": "2022-02-25T21:12:00.288000+00:00"
    }
]
}

```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[查看蓝/绿部署](#)。

### 示例 3：描述切换后的 Aurora MySQL 集群的蓝/绿部署

在绿色环境升级为生产环境之后，以下describe-blue-green-deployment示例检索有关蓝/绿部署的详细信息。

```

aws rds describe-blue-green-deployments \
  --blue-green-deployment-identifier bgd-wi89nwzglccsfake

```

输出：

```

{
  "BlueGreenDeployments": [
    {
      "BlueGreenDeploymentIdentifier": "bgd-wi89nwzglccsfake",
      "BlueGreenDeploymentName": "my-blue-green-deployment",
      "Source": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster-old1",
      "Target": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster",
      "SwitchoverDetails": [
        {
          "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster-old1",
          "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster",
          "Status": "SWITCHOVER_COMPLETED"
        },
        {
          "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-1-old1",
          "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-1",
          "Status": "SWITCHOVER_COMPLETED"
        }
      ]
    }
  ]
}

```



```
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-2-old1",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-2",
        "Status": "SWITCHOVER_COMPLETED"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-3-old1",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-3",
        "Status": "SWITCHOVER_COMPLETED"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-excluded-member-endpoint-old1",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-excluded-member-endpoint",
        "Status": "SWITCHOVER_COMPLETED"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-reader-endpoint-old1",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-reader-endpoint",
        "Status": "SWITCHOVER_COMPLETED"
    }
],
"Tasks": [
    {
        "Name": "CREATING_READ_REPLICA_OF_SOURCE",
        "Status": "COMPLETED"
    },
    {
        "Name": "DB_ENGINE_VERSION_UPGRADE",
        "Status": "COMPLETED"
    },
    {
        "Name": "CREATE_DB_INSTANCES_FOR_CLUSTER",
        "Status": "COMPLETED"
    },
    {
        "Name": "CREATE_CUSTOM_ENDPOINTS",
        "Status": "COMPLETED"
    }
]
```

```

    }
  ],
  "Status": "SWITCHOVER_COMPLETED",
  "CreateTime": "2022-02-25T22:38:49.522000+00:00"
}
]
}

```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[查看蓝/绿部署](#)。

#### 示例 4：描述蓝/绿组合部署

以下 describe-blue-green-deployment 示例检索蓝/绿组合部署的详细信息。

```
aws rds describe-blue-green-deployments
```

输出：

```

{
  "BlueGreenDeployments": [
    {
      "BlueGreenDeploymentIdentifier": "bgd-wi89nwzgfakelccs",
      "BlueGreenDeploymentName": "my-blue-green-deployment",
      "Source": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster",
      "Target": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster-green-3rnukl",
      "SwitchoverDetails": [
        {
          "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster",
          "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster-green-3rnukl",
          "Status": "AVAILABLE"
        },
        {
          "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-1",
          "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-1-green-gpmaxf",
          "Status": "AVAILABLE"
        }
      ]
    }
  ]
}

```

```
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-2",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-2-green-j2oajq",
        "Status": "AVAILABLE"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-3",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-3-green-mkxies",
        "Status": "AVAILABLE"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-excluded-member-endpoint",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-excluded-member-endpoint-green-4sqjrq",
        "Status": "AVAILABLE"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-reader-endpoint",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-reader-endpoint-green-gwwzlg",
        "Status": "AVAILABLE"
    }
],
"Tasks": [
    {
        "Name": "CREATING_READ_REPLICA_OF_SOURCE",
        "Status": "COMPLETED"
    },
    {
        "Name": "DB_ENGINE_VERSION_UPGRADE",
        "Status": "COMPLETED"
    },
    {
        "Name": "CREATE_DB_INSTANCES_FOR_CLUSTER",
        "Status": "COMPLETED"
    },
    {
        "Name": "CREATE_CUSTOM_ENDPOINTS",
        "Status": "COMPLETED"
    }
]
```

```
    }
  ],
  "Status": "AVAILABLE",
  "CreateTime": "2022-02-25T21:12:00.288000+00:00"
},
{
  "BlueGreenDeploymentIdentifier": "bgd-v5330365fake1eex",
  "BlueGreenDeploymentName": "bgd-cli-test-instance",
  "Source": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-old1",
  "Target": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance",
  "SwitchoverDetails": [
    {
      "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-old1",
      "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance",
      "Status": "SWITCHOVER_COMPLETED"
    },
    {
      "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-1-old1",
      "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-1",
      "Status": "SWITCHOVER_COMPLETED"
    },
    {
      "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-2-old1",
      "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-2",
      "Status": "SWITCHOVER_COMPLETED"
    },
    {
      "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-3-old1",
      "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-
instance-replica-3",
      "Status": "SWITCHOVER_COMPLETED"
    }
  ],
  "Tasks": [
    {
      "Name": "CREATING_READ_REPLICA_OF_SOURCE",
      "Status": "COMPLETED"
    }
  ]
}
```

```
    },
    {
      "Name": "DB_ENGINE_VERSION_UPGRADE",
      "Status": "COMPLETED"
    },
    {
      "Name": "CONFIGURE_BACKUPS",
      "Status": "COMPLETED"
    },
    {
      "Name": "CREATING_TOPOLOGY_OF_SOURCE",
      "Status": "COMPLETED"
    }
  ],
  "Status": "SWITCHOVER_COMPLETED",
  "CreateTime": "2022-02-25T22:33:22.225000+00:00"
}
]
```

有关更多信息，请参阅 [Amazon RDS 用户指南中的查看蓝/绿部署](#) 和 Amazon Aurora 用户指南中的 [查看蓝/绿部署](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeBlueGreenDeployments](#) 中的。

## describe-certificates

以下代码示例演示了如何使用 describe-certificates。

### AWS CLI

#### 描述证书

以下 describe-certificates 示例检索与用户默认区域关联的证书的详细信息。

```
aws rds describe-certificates
```

输出：

```
{
  "Certificates": [
    {
```

```

    "CertificateIdentifier": "rds-ca-ecc384-g1",
    "CertificateType": "CA",
    "Thumbprint": "2ee3dcc06e50192559b13929e73484354f23387d",
    "ValidFrom": "2021-05-24T22:06:59+00:00",
    "ValidTill": "2121-05-24T23:06:59+00:00",
    "CertificateArn": "arn:aws:rds:us-west-2::cert:rds-ca-ecc384-g1",
    "CustomerOverride": false
  },
  {
    "CertificateIdentifier": "rds-ca-rsa4096-g1",
    "CertificateType": "CA",
    "Thumbprint": "19da4f2af579a8ae1f6a0fa77aa5befd874b4cab",
    "ValidFrom": "2021-05-24T22:03:20+00:00",
    "ValidTill": "2121-05-24T23:03:20+00:00",
    "CertificateArn": "arn:aws:rds:us-west-2::cert:rds-ca-rsa4096-g1",
    "CustomerOverride": false
  },
  {
    "CertificateIdentifier": "rds-ca-rsa2048-g1",
    "CertificateType": "CA",
    "Thumbprint": "7c40cb42714b6fdb2b296f9bbd0e8bb364436a76",
    "ValidFrom": "2021-05-24T21:59:00+00:00",
    "ValidTill": "2061-05-24T22:59:00+00:00",
    "CertificateArn": "arn:aws:rds:us-west-2::cert:rds-ca-rsa2048-g1",
    "CustomerOverride": true,
    "CustomerOverrideValidTill": "2061-05-24T22:59:00+00:00"
  },
  {
    "CertificateIdentifier": "rds-ca-2019",
    "CertificateType": "CA",
    "Thumbprint": "d40ddb29e3750dfffa671c3140bbf5f478d1c8096",
    "ValidFrom": "2019-08-22T17:08:50+00:00",
    "ValidTill": "2024-08-22T17:08:50+00:00",
    "CertificateArn": "arn:aws:rds:us-west-2::cert:rds-ca-2019",
    "CustomerOverride": false
  }
],
"DefaultCertificateForNewLaunches": "rds-ca-rsa2048-g1"
}

```

有关更多信息，请参阅 Amazon RD [S 用户指南中的使用 SSL/TLS 加密与数据库实例的连接](#)和 Amazon Aurora 用户指南 [中的使用 SSL/TLS 加密与数据库集群的连接](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeCertificates](#) 中的。

## describe-db-cluster-backtracks

以下代码示例演示了如何使用 `describe-db-cluster-backtracks`。

### AWS CLI

描述数据库集群的回溯

以下 `describe-db-cluster-backtracks` 示例检索有关指定数据库集群的详细信息。

```
aws rds describe-db-cluster-backtracks \
  --db-cluster-identifier mydbcluster
```

输出：

```
{
  "DBClusterBacktracks": [
    {
      "DBClusterIdentifier": "mydbcluster",
      "BacktrackIdentifier": "2f5f5294-0dd2-44c9-9f50-EXAMPLE",
      "BacktrackTo": "2021-02-12T04:59:22Z",
      "BacktrackedFrom": "2021-02-12T14:37:31.640Z",
      "BacktrackRequestCreationTime": "2021-02-12T14:36:18.819Z",
      "Status": "COMPLETED"
    },
    {
      "DBClusterIdentifier": "mydbcluster",
      "BacktrackIdentifier": "3c7a6421-af2a-4ea3-ae95-EXAMPLE",
      "BacktrackTo": "2021-02-11T22:53:46Z",
      "BacktrackedFrom": "2021-02-12T00:09:27.006Z",
      "BacktrackRequestCreationTime": "2021-02-12T00:07:53.487Z",
      "Status": "COMPLETED"
    }
  ]
}
```

有关更多信息，请参阅亚马逊 Aurora 用户指南中的回溯 Aurora [数据库集群](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDbClusterBacktracks](#) 中的。

## describe-db-cluster-endpoints

以下代码示例演示了如何使用 `describe-db-cluster-endpoints`。

## AWS CLI

### 示例 1：描述数据库集群终端节点

以下 `describe-db-cluster-endpoints` 示例检索数据库集群终端节点的详细信息。最常见的 Aurora 集群有两个终端节点。一个端点有类型 `WRITER`。您可以将此端点用于所有 SQL 语句。另一个端点有类型 `READER`。此端点只能用于 `SELECT` 和其他只读 SQL 语句。

```
aws rds describe-db-cluster-endpoints
```

输出：

```
{
  "DBClusterEndpoints": [
    {
      "DBClusterIdentifier": "my-database-1",
      "Endpoint": "my-database-1.cluster-cnpexample.us-east-1.rds.amazonaws.com",
      "Status": "creating",
      "EndpointType": "WRITER"
    },
    {
      "DBClusterIdentifier": "my-database-1",
      "Endpoint": "my-database-1.cluster-ro-cnpexample.us-east-1.rds.amazonaws.com",
      "Status": "creating",
      "EndpointType": "READER"
    },
    {
      "DBClusterIdentifier": "mydbcluster",
      "Endpoint": "mydbcluster.cluster-cnpexample.us-east-1.rds.amazonaws.com",
      "Status": "available",
      "EndpointType": "WRITER"
    },
    {
      "DBClusterIdentifier": "mydbcluster",
      "Endpoint": "mydbcluster.cluster-ro-cnpexample.us-east-1.rds.amazonaws.com",
      "Status": "available",
      "EndpointType": "READER"
    }
  ]
}
```



```
}
```

## 示例 2：描述单个数据库集群的数据库集群终端节点

以下describe-db-cluster-endpoints示例检索单个指定数据库集群的数据库集群终端节点的详细信息。Aurora 无服务器集群只有一个类型为的终端节点。WRITER

```
aws rds describe-db-cluster-endpoints \  
  --db-cluster-identifier serverless-cluster
```

输出：

```
{  
  "DBClusterEndpoints": [  
    {  
      "Status": "available",  
      "Endpoint": "serverless-cluster.cluster-cnpxample.us-  
east-1.rds.amazonaws.com",  
      "DBClusterIdentifier": "serverless-cluster",  
      "EndpointType": "WRITER"  
    }  
  ]  
}
```

有关更多信息，请参阅[亚马逊 Aurora 用户指南中的亚马逊 Aurora 连接管理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbClusterEndpoints](#)中的。

## describe-db-cluster-parameter-groups

以下代码示例演示了如何使用 describe-db-cluster-parameter-groups。

AWS CLI

描述数据库集群参数组

以下describe-db-cluster-parameter-groups示例检索数据库集群参数组的详细信息。

```
aws rds describe-db-cluster-parameter-groups
```

输出：

```
{
  "DBClusterParameterGroups": [
    {
      "DBClusterParameterGroupName": "default.aurora-mysql5.7",
      "DBParameterGroupFamily": "aurora-mysql5.7",
      "Description": "Default cluster parameter group for aurora-mysql5.7",
      "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:cluster-pg:default.aurora-mysql5.7"
    },
    {
      "DBClusterParameterGroupName": "default.aurora-postgresql9.6",
      "DBParameterGroupFamily": "aurora-postgresql9.6",
      "Description": "Default cluster parameter group for aurora-
postgresql9.6",
      "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:cluster-pg:default.aurora-postgresql9.6"
    },
    {
      "DBClusterParameterGroupName": "default.aurora5.6",
      "DBParameterGroupFamily": "aurora5.6",
      "Description": "Default cluster parameter group for aurora5.6",
      "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:cluster-pg:default.aurora5.6"
    },
    {
      "DBClusterParameterGroupName": "mydbclusterpg",
      "DBParameterGroupFamily": "aurora-mysql5.7",
      "Description": "My DB cluster parameter group",
      "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:cluster-pg:mydbclusterpg"
    },
    {
      "DBClusterParameterGroupName": "mydbclusterpgcopy",
      "DBParameterGroupFamily": "aurora-mysql5.7",
      "Description": "Copy of mydbclusterpg parameter group",
      "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:cluster-pg:mydbclusterpgcopy"
    }
  ]
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的使用[数据库参数组](#)和[数据库集群参数组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbClusterParameterGroups](#)中的。

## describe-db-cluster-parameters

以下代码示例演示了如何使用 describe-db-cluster-parameters。

### AWS CLI

示例 1：描述数据库集群参数组中的参数

以下describe-db-cluster-parameters示例检索有关数据库集群参数组中参数的详细信息。

```
aws rds describe-db-cluster-parameters \  
  --db-cluster-parameter-group-name mydbclusterpg
```

输出：

```
{  
  "Parameters": [  
    {  
      "ParameterName": "allow-suspicious-udfs",  
      "Description": "Controls whether user-defined functions that have only  
an xxx symbol for the main function can be loaded",  
      "Source": "engine-default",  
      "ApplyType": "static",  
      "DataType": "boolean",  
      "AllowedValues": "0,1",  
      "IsModifiable": false,  
      "ApplyMethod": "pending-reboot",  
      "SupportedEngineModes": [  
        "provisioned"  
      ]  
    },  
    {  
      "ParameterName": "aurora_lab_mode",  
      "ParameterValue": "0",  
      "Description": "Enables new features in the Aurora engine.",  
      "Source": "engine-default",  
      "ApplyType": "static",  
      "DataType": "boolean",  
      "AllowedValues": "0,1",  
      "IsModifiable": true,  
      "ApplyMethod": "pending-reboot",
```

```

        "SupportedEngineModes": [
            "provisioned"
        ]
    },
    ...some output truncated...
]
}

```

示例 2：仅列出数据库集群参数组中的参数名称

以下describe-db-cluster-parameters示例仅检索数据库集群参数组中参数的名称。

```

aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name default.aurora-mysql5.7 \
  --query 'Parameters[].{ParameterName:ParameterName}'

```

输出：

```

[
  {
    "ParameterName": "allow-suspicious-udfs"
  },
  {
    "ParameterName": "aurora_binlog_read_buffer_size"
  },
  {
    "ParameterName": "aurora_binlog_replication_max_yield_seconds"
  },
  {
    "ParameterName": "aurora_binlog_use_large_read_buffer"
  },
  {
    "ParameterName": "aurora_lab_mode"
  },
  ...some output truncated...
]

```

示例 3：仅描述数据库集群参数组中可修改的参数

以下describe-db-cluster-parameters示例仅检索可在数据库集群参数组中修改的参数的名称。

```
aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name default.aurora-mysql5.7 \
  --query 'Parameters[].{ParameterName:ParameterName,IsModifiable:IsModifiable} |
  [?IsModifiable == `true`]'
```

输出：

```
[
  {
    "ParameterName": "aurora_binlog_read_buffer_size",
    "IsModifiable": true
  },
  {
    "ParameterName": "aurora_binlog_replication_max_yield_seconds",
    "IsModifiable": true
  },
  {
    "ParameterName": "aurora_binlog_use_large_read_buffer",
    "IsModifiable": true
  },
  {
    "ParameterName": "aurora_lab_mode",
    "IsModifiable": true
  },
  ...some output truncated...
]
```

示例 4：仅描述数据库集群参数组中可修改的布尔参数

以下describe-db-cluster-parameters示例仅检索可在数据库集群参数组中修改且数据类型为布尔值的参数的名称。

```
aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name default.aurora-mysql5.7 \
  --query 'Parameters[.
  {ParameterName:ParameterName,DataType:DataType,IsModifiable:IsModifiable} | [?
  DataType == `boolean`] | [?IsModifiable == `true`]'
```

输出：

```
[
  {
    "DataType": "boolean",
    "ParameterName": "aurora_binlog_use_large_read_buffer",
    "IsModifiable": true
  },
  {
    "DataType": "boolean",
    "ParameterName": "aurora_lab_mode",
    "IsModifiable": true
  },
  {
    "DataType": "boolean",
    "ParameterName": "autocommit",
    "IsModifiable": true
  },
  {
    "DataType": "boolean",
    "ParameterName": "automatic_sp_privileges",
    "IsModifiable": true
  },
  ...some output truncated...
]
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的使用[数据库参数组和数据库集群参数组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeDbClusterParameters](#)中的。

## describe-db-cluster-snapshot-attributes

以下代码示例演示了如何使用 describe-db-cluster-snapshot-attributes。

### AWS CLI

描述数据库集群快照的属性名称和值

以下 describe-db-cluster-snapshot-attributes 示例检索指定数据库集群快照的属性名称和值的详细信息。

```
aws rds describe-db-cluster-snapshot-attributes \
  --db-cluster-snapshot-identifier myclustersnapshot
```

输出：

```
{
  "DBClusterSnapshotAttributesResult": {
    "DBClusterSnapshotIdentifier": "myclustersnapshot",
    "DBClusterSnapshotAttributes": [
      {
        "AttributeName": "restore",
        "AttributeValues": [
          "123456789012"
        ]
      }
    ]
  }
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[共享数据库集群快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeDbClusterSnapshotAttributes](#)中的。

## describe-db-cluster-snapshots

以下代码示例演示了如何使用 describe-db-cluster-snapshots。

### AWS CLI

描述数据库集群的数据库集群快照

以下 describe-db-cluster-snapshots 示例检索指定数据库集群的数据库集群快照的详细信息。

```
aws rds describe-db-cluster-snapshots \
  --db-cluster-identifier mydbcluster
```

输出：

```
{
  "DBClusterSnapshots": [
    {
      "AvailabilityZones": [
        "us-east-1a",
        "us-east-1b",
        "us-east-1e"
      ]
    }
  ]
}
```

```

    ],
    "DBClusterSnapshotIdentifier": "myclustersnapshotcopy",
    "DBClusterIdentifier": "mydbcluster",
    "SnapshotCreateTime": "2019-06-04T09:16:42.649Z",
    "Engine": "aurora-mysql",
    "AllocatedStorage": 0,
    "Status": "available",
    "Port": 0,
    "VpcId": "vpc-6594f31c",
    "ClusterCreateTime": "2019-04-15T14:18:42.785Z",
    "MasterUsername": "myadmin",
    "EngineVersion": "5.7.mysql_aurora.2.04.2",
    "LicenseModel": "aurora-mysql",
    "SnapshotType": "manual",
    "PercentProgress": 100,
    "StorageEncrypted": true,
    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
AKIAIOSFODNN7EXAMPLE",
    "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:814387698303:cluster-
snapshot:myclustersnapshotcopy",
    "IAMDatabaseAuthenticationEnabled": false
  },
  {
    "AvailabilityZones": [
      "us-east-1a",
      "us-east-1b",
      "us-east-1e"
    ],
    "DBClusterSnapshotIdentifier": "rds:mydbcluster-2019-06-20-09-16",
    "DBClusterIdentifier": "mydbcluster",
    "SnapshotCreateTime": "2019-06-20T09:16:26.569Z",
    "Engine": "aurora-mysql",
    "AllocatedStorage": 0,
    "Status": "available",
    "Port": 0,
    "VpcId": "vpc-6594f31c",
    "ClusterCreateTime": "2019-04-15T14:18:42.785Z",
    "MasterUsername": "myadmin",
    "EngineVersion": "5.7.mysql_aurora.2.04.2",
    "LicenseModel": "aurora-mysql",
    "SnapshotType": "automated",
    "PercentProgress": 100,
    "StorageEncrypted": true,

```



```
        "KmsKeyId": "arn:aws:kms:us-east-1:814387698303:key/
AKIAIOSFODNN7EXAMPLE",
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:123456789012:cluster-
snapshot:rds:mydbcluster-2019-06-20-09-16",
        "IAMDatabaseAuthenticationEnabled": false
    }
]
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[创建数据库集群快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeDbClusterSnapshots](#)中的。

## describe-db-clusters

以下代码示例演示了如何使用 describe-db-clusters。

### AWS CLI

#### 示例 1：描述数据库集群

以下 describe-db-clusters 示例检索指定数据库集群的详细信息。

```
aws rds describe-db-clusters \
  --db-cluster-identifier mydbcluster
```

输出：

```
{
  "DBClusters": [
    {
      "AllocatedStorage": 1,
      "AvailabilityZones": [
        "us-east-1a",
        "us-east-1b",
        "us-east-1e"
      ],
      "BackupRetentionPeriod": 1,
      "DatabaseName": "mydbcluster",
      "DBClusterIdentifier": "mydbcluster",
      "DBClusterParameterGroup": "default.aurora-mysql5.7",
      "DBSubnetGroup": "default",
      "Status": "available",
```

```
    "EarliestRestorableTime": "2019-06-19T09:16:28.210Z",
    "Endpoint": "mydbcluster.cluster-cnpxexample.us-
east-1.rds.amazonaws.com",
    "ReaderEndpoint": "mydbcluster.cluster-ro-cnpxexample.us-
east-1.rds.amazonaws.com",
    "MultiAZ": true,
    "Engine": "aurora-mysql",
    "EngineVersion": "5.7.mysql_aurora.2.04.2",
    "LatestRestorableTime": "2019-06-20T22:38:14.908Z",
    "Port": 3306,
    "MasterUsername": "myadmin",
    "PreferredBackupWindow": "09:09-09:39",
    "PreferredMaintenanceWindow": "sat:04:09-sat:04:39",
    "ReadReplicaIdentifiers": [],
    "DBClusterMembers": [
      {
        "DBInstanceIdentifier": "dbinstance3",
        "IsClusterWriter": false,
        "DBClusterParameterGroupStatus": "in-sync",
        "PromotionTier": 1
      },
      {
        "DBInstanceIdentifier": "dbinstance1",
        "IsClusterWriter": false,
        "DBClusterParameterGroupStatus": "in-sync",
        "PromotionTier": 1
      },
      {
        "DBInstanceIdentifier": "dbinstance2",
        "IsClusterWriter": false,
        "DBClusterParameterGroupStatus": "in-sync",
        "PromotionTier": 1
      },
      {
        "DBInstanceIdentifier": "mydbcluster",
        "IsClusterWriter": false,
        "DBClusterParameterGroupStatus": "in-sync",
        "PromotionTier": 1
      },
      {
        "DBInstanceIdentifier": "mydbcluster-us-east-1b",
        "IsClusterWriter": false,
        "DBClusterParameterGroupStatus": "in-sync",
        "PromotionTier": 1
      }
    ]
  }
}
```

```

    },
    {
      "DBInstanceIdentifier": "mydbcluster",
      "IsClusterWriter": true,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    }
  ],
  "VpcSecurityGroups": [
    {
      "VpcSecurityGroupId": "sg-0b9130572daf3dc16",
      "Status": "active"
    }
  ],
  "HostedZoneId": "Z2R2ITUGPM61AM",
  "StorageEncrypted": true,
  "KmsKeyId": "arn:aws:kms:us-east-1:814387698303:key/
AKIAIOSFODNN7EXAMPLE",
  "DbClusterResourceId": "cluster-AKIAIOSFODNN7EXAMPLE",
  "DBClusterArn": "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster",
  "AssociatedRoles": [],
  "IAMDatabaseAuthenticationEnabled": false,
  "ClusterCreateTime": "2019-04-15T14:18:42.785Z",
  "EngineMode": "provisioned",
  "DeletionProtection": false,
  "HttpEndpointEnabled": false
}
]
}

```

示例 2：列出所有数据库集群的某些属性

以下describe-db-clusters示例仅检索当前 AWS 区域中所有数据库集群的DBClusterIdentifierEndpoint、和ReaderEndpoint属性。

```

aws rds describe-db-clusters \
  --query 'DBClusters[].[
{DBClusterIdentifier:DBClusterIdentifier,Endpoint:Endpoint,ReaderEndpoint:ReaderEndpoint}]'

```

输出：

```
[
```

```

    {
      "Endpoint": "cluster-57-2020-05-01-2270.cluster-cnpxample.us-
east-1.rds.amazonaws.com",
      "ReaderEndpoint": "cluster-57-2020-05-01-2270.cluster-ro-cnpxample.us-
east-1.rds.amazonaws.com",
      "DBClusterIdentifier": "cluster-57-2020-05-01-2270"
    },
    {
      "Endpoint": "cluster-57-2020-05-01-4615.cluster-cnpxample.us-
east-1.rds.amazonaws.com",
      "ReaderEndpoint": "cluster-57-2020-05-01-4615.cluster-ro-cnpxample.us-
east-1.rds.amazonaws.com",
      "DBClusterIdentifier": "cluster-57-2020-05-01-4615"
    },
    {
      "Endpoint": "pg2-cluster.cluster-cnpxample.us-east-1.rds.amazonaws.com",
      "ReaderEndpoint": "pg2-cluster.cluster-ro-cnpxample.us-
east-1.rds.amazonaws.com",
      "DBClusterIdentifier": "pg2-cluster"
    },
    ...output omitted...
  }
]

```

### 示例 3：列出具有特定属性的数据库集群

以下describe-db-clusters示例仅检索使用数据库引擎的数据库集群的DBClusterIdentifieraurora-postgresql和Engine属性。

```

aws rds describe-db-clusters \
  --query 'DBClusters[].{DBClusterIdentifier:DBClusterIdentifier,Engine:Engine} |
[?Engine == `aurora-postgresql`]'

```

输出：

```

[
  {
    "Engine": "aurora-postgresql",
    "DBClusterIdentifier": "pg2-cluster"
  }
]

```

有关更多信息，请参阅[亚马逊 Aurora 用户指南中的亚马逊 Aurora 数据库集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbClusters](#)中的。

## describe-db-engine-versions

以下代码示例演示了如何使用 describe-db-engine-versions。

### AWS CLI

描述 MySQL 数据库引擎的数据库引擎版本

以下 describe-db-engine-versions 示例显示了有关指定数据库引擎的每个数据库引擎版本的详细信息。

```
aws rds describe-db-engine-versions \  
  --engine mysql
```

输出：

```
{  
  "DBEngineVersions": [  
    {  
      "Engine": "mysql",  
      "EngineVersion": "5.5.46",  
      "DBParameterGroupFamily": "mysql5.5",  
      "DBEngineDescription": "MySQL Community Edition",  
      "DBEngineVersionDescription": "MySQL 5.5.46",  
      "ValidUpgradeTarget": [  
        {  
          "Engine": "mysql",  
          "EngineVersion": "5.5.53",  
          "Description": "MySQL 5.5.53",  
          "AutoUpgrade": false,  
          "IsMajorVersionUpgrade": false  
        },  
        {  
          "Engine": "mysql",  
          "EngineVersion": "5.5.54",  
          "Description": "MySQL 5.5.54",  
          "AutoUpgrade": false,  
          "IsMajorVersionUpgrade": false  
        }  
      ]  
    }  
  ]  
}
```

```

    {
      "Engine": "mysql",
      "EngineVersion": "5.5.57",
      "Description": "MySQL 5.5.57",
      "AutoUpgrade": false,
      "IsMajorVersionUpgrade": false
    },
    ...some output truncated...
  ]
}

```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[什么是 Amazon Relational Database Service \( Amazon RDS \) ?](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》EngineVersions 中的 [describedB](#)。

## describe-db-instance-automated-backups

以下代码示例演示了如何使用 describe-db-instance-automated-backups。

### AWS CLI

描述数据库实例的自动备份

以下 describe-db-instance-automated-backups 示例显示了有关指定数据库实例自动备份的详细信息。详细信息包括在其他 AWS 区域复制的自动备份。

```
aws rds describe-db-instance-automated-backups \
  --db-instance-identifier new-orcl-db
```

输出：

```

{
  "DBInstanceAutomatedBackups": [
    {
      "DBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:new-orcl-db",
      "DbiResourceId": "db-JKIB2GFQ5RV7REPLZA4EXAMPLE",
      "Region": "us-east-1",
      "DBInstanceIdentifier": "new-orcl-db",
      "RestoreWindow": {
        "EarliestTime": "2020-12-07T21:05:20.939Z",
        "LatestTime": "2020-12-07T21:05:20.939Z"
      }
    }
  ]
}

```

```

    },
    "AllocatedStorage": 20,
    "Status": "replicating",
    "Port": 1521,
    "InstanceCreateTime": "2020-12-04T15:28:31Z",
    "MasterUsername": "admin",
    "Engine": "oracle-se2",
    "EngineVersion": "12.1.0.2.v21",
    "LicenseModel": "bring-your-own-license",
    "OptionGroupName": "default:oracle-se2-12-1",
    "Encrypted": false,
    "StorageType": "gp2",
    "IAMDatabaseAuthenticationEnabled": false,
    "BackupRetentionPeriod": 14,
    "DBInstanceAutomatedBackupsArn": "arn:aws:rds:us-
west-2:123456789012:auto-backup:ab-jkib2gfg5rv7replzadausbrktni2bn4example"
  }
]
}

```

有关更多信息，请参阅 Amazon RDS 用户指南中的[查找有关复制备份的信息](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeDbInstanceAutomatedBackups](#)中的。

## describe-db-instances

以下代码示例演示了如何使用 describe-db-instances。

### AWS CLI

#### 描述数据库实例

以下 describe-db-instances 示例将检索有关指定数据库实例的详细信息。

```
aws rds describe-db-instances \
  --db-instance-identifier mydbinstancecf
```

输出：

```
{
  "DBInstances": [
    {
      "DBInstanceIdentifier": "mydbinstancecf",
```

```

        "DBInstanceClass": "db.t3.small",
        "Engine": "mysql",
        "DBInstanceStatus": "available",
        "MasterUsername": "masterawsuser",
        "Endpoint": {
            "Address": "mydbinstancecf.abcxample.us-east-1.rds.amazonaws.com",
            "Port": 3306,
            "HostedZoneId": "Z2R2ITUGPM61AM"
        },
        ...some output truncated...
    }
]
}

```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DescribeDBInstances](#)。

## describe-db-log-files

以下代码示例演示了如何使用 describe-db-log-files。

### AWS CLI

描述数据库实例的日志文件

以下describe-db-log-files示例检索有关指定数据库实例的日志文件的详细信息。

```
aws rds describe-db-log-files -\
    -db-instance-identifier test-instance
```

输出：

```

{
  "DescribeDBLogFiles": [
    {
      "Size": 0,
      "LastWritten": 1533060000000,
      "LogFileName": "error/mysql-error-running.log"
    },
    {
      "Size": 2683,
      "LastWritten": 1532994300000,
      "LogFileName": "error/mysql-error-running.log.0"
    }
  ]
}

```



```
    },
    {
      "Size": 107,
      "LastWritten": 1533057300000,
      "LogFileName": "error/mysql-error-running.log.18"
    },
    {
      "Size": 13105,
      "LastWritten": 1532991000000,
      "LogFileName": "error/mysql-error-running.log.23"
    },
    {
      "Size": 0,
      "LastWritten": 1533061200000,
      "LogFileName": "error/mysql-error.log"
    },
    {
      "Size": 3519,
      "LastWritten": 1532989252000,
      "LogFileName": "mysqlUpgrade"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbLogFiles](#)中的。

## describe-db-parameter-groups

以下代码示例演示了如何使用 describe-db-parameter-groups。

### AWS CLI

#### 描述数据库参数组

以下 describe-db-parameter-groups 示例将检索有关数据库参数组的详细信息。

```
aws rds describe-db-parameter-groups
```

输出：

```
{
  "DBParameterGroups": [
```

```

    {
      "DBParameterGroupName": "default.aurora-mysql5.7",
      "DBParameterGroupFamily": "aurora-mysql5.7",
      "Description": "Default parameter group for aurora-mysql5.7",
      "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora-mysql5.7"
    },
    {
      "DBParameterGroupName": "default.aurora-postgresql9.6",
      "DBParameterGroupFamily": "aurora-postgresql9.6",
      "Description": "Default parameter group for aurora-postgresql9.6",
      "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora-postgresql9.6"
    },
    {
      "DBParameterGroupName": "default.aurora5.6",
      "DBParameterGroupFamily": "aurora5.6",
      "Description": "Default parameter group for aurora5.6",
      "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora5.6"
    },
    {
      "DBParameterGroupName": "default.mariadb10.1",
      "DBParameterGroupFamily": "mariadb10.1",
      "Description": "Default parameter group for mariadb10.1",
      "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.mariadb10.1"
    },
    ...some output truncated...
  ]
}

```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[使用数据库参数组](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》ParameterGroups 中的 [describeDB](#)。

## describe-db-parameters

以下代码示例演示了如何使用 describe-db-parameters。

### AWS CLI

#### 描述数据库参数组中的参数

以下 `describe-db-parameters` 示例将检索有关指定数据库参数组的详细信息。

```
aws rds describe-db-parameters \  
  --db-parameter-group-name mydbpg
```

输出：

```
{  
  "Parameters": [  
    {  
      "ParameterName": "allow-suspicious-udfs",  
      "Description": "Controls whether user-defined functions that have only  
an xxx symbol for the main function can be loaded",  
      "Source": "engine-default",  
      "ApplyType": "static",  
      "DataType": "boolean",  
      "AllowedValues": "0,1",  
      "IsModifiable": false,  
      "ApplyMethod": "pending-reboot"  
    },  
    {  
      "ParameterName": "auto_generate_certs",  
      "Description": "Controls whether the server autogenerates SSL key and  
certificate files in the data directory, if they do not already exist.",  
      "Source": "engine-default",  
      "ApplyType": "static",  
      "DataType": "boolean",  
      "AllowedValues": "0,1",  
      "IsModifiable": false,  
      "ApplyMethod": "pending-reboot"  
    },  
    ...some output truncated...  
  ]  
}
```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[使用数据库参数组](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DescribeDBParameters](#)。

## describe-db-proxies

以下代码示例演示了如何使用 `describe-db-proxies`。

## AWS CLI

描述 RDS 数据库的数据库代理

以下describe-db-proxies示例返回有关数据库代理的信息。

```
aws rds describe-db-proxies
```

输出：

```
{
  "DBProxies": [
    {
      "DBProxyName": "proxyExample1",
      "DBProxyArn": "arn:aws:rds:us-east-1:123456789012:db-proxy:prx-0123a01b12345c0ab",
      "Status": "available",
      "EngineFamily": "PostgreSQL",
      "VpcId": "vpc-1234567",
      "VpcSecurityGroupIds": [
        "sg-1234"
      ],
      "VpcSubnetIds": [
        "subnetgroup1",
        "subnetgroup2"
      ],
      "Auth": "[
        {
          "Description": "proxydescription1"
          "AuthScheme": "SECRETS",
          "SecretArn": "arn:aws:secretsmanager:us-west-2:123456789123:secret:secretName-1234f",
          "IAMAuth": "DISABLED"
        }
      ]",
      "RoleArn": "arn:aws:iam::12345678912??:role/ProxyPostgreSQLRole",
      "Endpoint": "proxyExample1.proxy-ab0cd1efghij.us-east-1.rds.amazonaws.com",
      "RequireTLS": false,
      "IdleClientTimeout": 1800,
      "DebuggingLogging": false,
      "CreateDate": "2023-04-05T16:09:33.452000+00:00",
      "UpdatedDate": "2023-04-13T01:49:38.568000+00:00"
    }
  ]
}
```

```

    },
    {
      "DBProxyName": "proxyExample2",
      "DBProxyArn": "arn:aws:rds:us-east-1:123456789012:db-
proxy:prx-1234a12b23456c1ab",
      "Status": "available",
      "EngineFamily": "PostgreSQL",
      "VpcId": "sg-1234567",
      "VpcSecurityGroupIds": [
        "sg-1234"
      ],
      "VpcSubnetIds": [
        "subnetgroup1",
        "subnetgroup2"
      ],
      "Auth": "[
        {
          "Description": "proxydescription2"
          "AuthScheme": "SECRETS",
          "SecretArn": "aarn:aws:secretsmanager:us-
west-2:123456789123:secret:secretName-1234f",
          "IAMAuth": "DISABLED"
        }
      ]",
      "RoleArn": "arn:aws:iam::12345678912:role/ProxyPostgreSQLRole",
      "Endpoint": "proxyExample2.proxy-ab0cd1efghij.us-
east-1.rds.amazonaws.com",
      "RequireTLS": false,
      "IdleClientTimeout": 1800,
      "DebuggingLogging": false,
      "CreateDate": "2022-01-05T16:19:33.452000+00:00",
      "UpdateDate": "2023-04-13T01:49:38.568000+00:00"
    }
  ]
}

```

有关更多信息，请参阅 Amazon [RDS 用户指南中的查看 RDS 代理](#)和亚马逊 Aurora 用户指南中的 [查看 RDS 代理](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDbProxies](#) 中的。

## describe-db-proxy-endpoints

以下代码示例演示了如何使用 `describe-db-proxy-endpoints`。

### AWS CLI

描述数据库代理终端节点

以下 `describe-db-proxy-endpoints` 示例返回有关数据库代理终端节点的信息。

```
aws rds describe-db-proxy-endpoints
```

输出：

```
{
  "DBProxyEndpoints": [
    {
      "DBProxyEndpointName": "proxyEndpoint1",
      "DBProxyEndpointArn": "arn:aws:rds:us-east-1:123456789012:db-proxy-
endpoint:prx-endpoint-0123a01b12345c0ab",
      "DBProxyName": "proxyExample",
      "Status": "available",
      "VpcId": "vpc-1234567",
      "VpcSecurityGroupIds": [
        "sg-1234"
      ],
      "VpcSubnetIds": [
        "subnetgroup1",
        "subnetgroup2"
      ],
      "Endpoint": "proxyEndpoint1.endpoint.proxy-ab0cd1efghij.us-
east-1.rds.amazonaws.com",
      "CreateDate": "2023-04-05T16:09:33.452000+00:00",
      "TargetRole": "READ_WRITE",
      "IsDefault": false
    },
    {
      "DBProxyEndpointName": "proxyEndpoint2",
      "DBProxyEndpointArn": "arn:aws:rds:us-east-1:123456789012:db-proxy-
endpoint:prx-endpoint-4567a01b12345c0ab",
      "DBProxyName": "proxyExample2",
      "Status": "available",
      "VpcId": "vpc1234567",
      "VpcSecurityGroupIds": [
```

```
        "sg-5678"
      ],
      "VpcSubnetIds": [
        "subnetgroup1",
        "subnetgroup2"
      ],
      "Endpoint": "proxyEndpoint2.endpoint.proxy-cd1ef2klmnop.us-
east-1.rds.amazonaws.com",
      "CreateDate": "2023-04-05T16:09:33.452000+00:00",
      "TargetRole": "READ_WRITE",
      "IsDefault": false
    }
  ]
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[查看代理终端节点](#)和 Amazon Aurora 用户指南中的[创建代理终端节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeDbProxyEndpoints](#)中的。

## describe-db-proxy-target-groups

以下代码示例演示了如何使用 describe-db-proxy-target-groups。

### AWS CLI

描述数据库代理终端节点

以下 describe-db-proxy-target-groups 示例返回有关数据库代理目标组的信息。

```
aws rds describe-db-proxy-target-groups \
  --db-proxy-name proxyExample
```

输出：

```
{
  "TargetGroups":
    {
      "DBProxyName": "proxyExample",
      "TargetGroupName": "default",
      "TargetGroupArn": "arn:aws:rds:us-east-1:123456789012:target-group:prx-
tg-0123a01b12345c0ab",
      "IsDefault": true,
    }
}
```

```
    "Status": "available",
    "ConnectionPoolConfig": {
      "MaxConnectionsPercent": 100,
      "MaxIdleConnectionsPercent": 50,
      "ConnectionBorrowTimeout": 120,
      "SessionPinningFilters": []
    },
    "CreateDate": "2023-05-02T18:41:19.495000+00:00",
    "UpdateDate": "2023-05-02T18:41:21.762000+00:00"
  }
}
```

有关更多信息，请参阅 Amazon [RDS 用户指南中的查看 RDS 代理](#)和亚马逊 Aurora 用户指南中的[查看 RDS 代理](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDbProxyTargetGroups](#) 中的。

## describe-db-proxy-targets

以下代码示例演示了如何使用 describe-db-proxy-targets。

### AWS CLI

描述数据库代理目标

以下 describe-db-proxy-targets 示例返回有关数据库代理目标的信息。

```
aws rds describe-db-proxy-targets \
  --db-proxy-name proxyExample
```

输出：

```
{
  "Targets": [
    {
      "Endpoint": "database1.ab0cd1efghij.us-east-1.rds.amazonaws.com",
      "TrackedClusterId": "database1",
      "RdsResourceId": "database1-instance-1",
      "Port": 3306,
      "Type": "RDS_INSTANCE",
      "Role": "READ_WRITE",
      "TargetHealth": {
        "State": "UNAVAILABLE",
```



```

        "Reason": "PENDING_PROXY_CAPACITY",
        "Description": "DBProxy Target is waiting for proxy to scale to
desired capacity"
    }
}
]
}

```

有关更多信息，请参阅 Amazon [RDS 用户指南中的查看 RDS 代理](#)和亚马逊 Aurora 用户指南中的[查看 RDS 代理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbProxyTargets](#)中的。

## describe-db-recommendations

以下代码示例演示了如何使用 describe-db-recommendations。

### AWS CLI

示例 1：列出所有数据库建议

以下describe-db-recommendations示例列出了您 AWS 账户中的所有数据库推荐。

```
aws rds describe-db-recommendations
```

输出：

```

{
  "DBRecommendations": [
    {
      "RecommendationId": "12ab3cde-f456-7g8h-9012-i3j45678k9lm",
      "TypeId": "config_recommendation::old_minor_version",
      "Severity": "informational",
      "ResourceArn": "arn:aws:rds:us-west-2:111122223333:db:database-1",
      "Status": "active",
      "CreatedTime": "2024-02-21T23:14:19.292000+00:00",
      "UpdatedTime": "2024-02-21T23:14:19+00:00",
      "Detection": "***[resource-name]** is not running the latest minor DB
engine version",
      "Recommendation": "Upgrade to latest engine version",
      "Description": "Your database resources aren't running the latest minor
DB engine version. The latest minor version contains the latest security fixes and
other improvements.",
    }
  ]
}

```

```
"RecommendedActions": [
  {
    "ActionId": "12ab34c5de6fg7h89i0jk1lm234n5678",
    "Operation": "modifyDbInstance",
    "Parameters": [
      {
        "Key": "EngineVersion",
        "Value": "5.7.44"
      },
      {
        "Key": "DBInstanceIdentifier",
        "Value": "database-1"
      }
    ],
    "ApplyModes": [
      "immediately",
      "next-maintenance-window"
    ],
    "Status": "ready",
    "ContextAttributes": [
      {
        "Key": "Recommended value",
        "Value": "5.7.44"
      },
      {
        "Key": "Current engine version",
        "Value": "5.7.42"
      }
    ]
  }
],
"Category": "security",
"Source": "RDS",
"TypeDetection": "***[resource-count] resources** are not running the latest minor DB engine version",
"TypeRecommendation": "Upgrade to latest engine version",
"Impact": "Reduced database performance and data security at risk",
"AdditionalInfo": "We recommend that you maintain your database with the latest DB engine minor version as this version includes the latest security and functionality fixes. The DB engine minor version upgrades contain only the changes which are backward-compatible with earlier minor versions of the same major version of the DB engine.",
"Links": [
  {
```

```

        "Text": "Upgrading an RDS DB instance engine version",
        "Url": "https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
USER_UpgradeDBInstance.Upgrading.html"
    },
    {
        "Text": "Using Amazon RDS Blue/Green Deployments for database
updates for Amazon Aurora",
        "Url": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/blue-green-deployments.html"
    },
    {
        "Text": "Using Amazon RDS Blue/Green Deployments for database
updates for Amazon RDS",
        "Url": "https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
blue-green-deployments.html"
    }
]
}
]
}

```

有关更多信息，请参阅[亚马逊 RDS 用户指南中的查看和回应 Amazon RDS 建议](#)和亚马逊 Aurora 用户指南中的[查看和回复亚马逊 RDS 建议](#)。

#### 示例 2：列出高严重性数据库建议

以下describe-db-recommendations示例列出了您 AWS 账户中的高严重性数据库建议。

```
aws rds describe-db-recommendations \
  --filters Name=severity,Values=high
```

输出：

```

{
  "DBRecommendations": [
    {
      "RecommendationId": "12ab3cde-f456-7g8h-9012-i3j45678k9lm",
      "TypeId": "config_recommendation::rds_extended_support",
      "Severity": "high",
      "ResourceArn": "arn:aws:rds:us-west-2:111122223333:db:database-1",
      "Status": "active",
      "CreatedTime": "2024-02-21T23:14:19.392000+00:00",
      "UpdatedTime": "2024-02-21T23:14:19+00:00",
    }
  ]
}

```

```
"Detection": "Your databases will be auto-enrolled to RDS Extended Support on February 29",
  "Recommendation": "Upgrade your major version before February 29, 2024 to avoid additional charges",
  "Description": "Your PostgreSQL 11 and MySQL 5.7 databases will be automatically enrolled into RDS Extended Support on February 29, 2024. To avoid the increase in charges due to RDS Extended Support, we recommend upgrading your databases to a newer major engine version before February 29, 2024.\n\nTo learn more about the RDS Extended Support pricing, refer to the pricing page.",
  "RecommendedActions": [
    {
      "ActionId": "12ab34c5de6fg7h89i0jk1lm234n5678",
      "Parameters": [],
      "ApplyModes": [
        "manual"
      ],
      "Status": "ready",
      "ContextAttributes": []
    }
  ],
  "Category": "cost optimization",
  "Source": "RDS",
  "TypeDetection": "Your database will be auto-enrolled to RDS Extended Support on February 29",
  "TypeRecommendation": "Upgrade your major version before February 29, 2024 to avoid additional charges",
  "Impact": "Increase in charges due to RDS Extended Support",
  "AdditionalInfo": "With Amazon RDS Extended Support, you can continue running your database on a major engine version past the RDS end of standard support date for an additional cost. This paid feature gives you more time to upgrade to a supported major engine version.\n\nDuring Extended Support, Amazon RDS will supply critical CVE patches and bug fixes.",
  "Links": [
    {
      "Text": "Amazon RDS Extended Support pricing for RDS for MySQL",
      "Url": "https://aws.amazon.com/rds/mysql/pricing/"
    },
    {
      "Text": "Amazon RDS Extended Support for RDS for MySQL and PostgreSQL databases",
      "Url": "https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/extended-support.html"
    }
  ]
}
```

```

    "Text": "Amazon RDS Extended Support pricing for Amazon Aurora
PostgreSQL",
    "Url": "https://aws.amazon.com/rds/aurora/pricing/"
  },
  {
    "Text": "Amazon RDS Extended Support for Aurora PostgreSQL
databases",
    "Url": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/extended-support.html"
  },
  {
    "Text": "Amazon RDS Extended Support pricing for RDS for
PostgreSQL",
    "Url": "https://aws.amazon.com/rds/postgresql/pricing/"
  }
]
}
]
}

```

有关更多信息，请参阅[亚马逊 RDS 用户指南中的查看和回应 Amazon RDS 建议](#)和[亚马逊 Aurora 用户指南中的查看和回复亚马逊 RDS 建议](#)。

示例 3：列出指定数据库实例的数据库建议

以下describe-db-recommendations示例列出了指定数据库实例的所有数据库建议。

```

aws rds describe-db-recommendations \
  --filters Name=dbi-resource-id,Values=database-1

```

输出：

```

{
  "DBRecommendations": [
    {
      "RecommendationId": "12ab3cde-f456-7g8h-9012-i3j45678k9lm",
      "TypeId": "config_recommendation::old_minor_version",
      "Severity": "informational",
      "ResourceArn": "arn:aws:rds:us-west-2:111122223333:db:database-1",
      "Status": "active",
      "CreatedTime": "2024-02-21T23:14:19.292000+00:00",
      "UpdatedTime": "2024-02-21T23:14:19+00:00",
    }
  ]
}

```

```
    "Detection": "***[resource-name]** is not running the latest minor DB
engine version",
    "Recommendation": "Upgrade to latest engine version",
    "Description": "Your database resources aren't running the latest minor
DB engine version. The latest minor version contains the latest security fixes and
other improvements.",
    "RecommendedActions": [
      {
        "ActionId": "12ab34c5de6fg7h89i0jk1lm234n5678",
        "Operation": "modifyDbInstance",
        "Parameters": [
          {
            "Key": "EngineVersion",
            "Value": "5.7.44"
          },
          {
            "Key": "DBInstanceIdentifier",
            "Value": "database-1"
          }
        ],
        "ApplyModes": [
          "immediately",
          "next-maintenance-window"
        ],
        "Status": "ready",
        "ContextAttributes": [
          {
            "Key": "Recommended value",
            "Value": "5.7.44"
          },
          {
            "Key": "Current engine version",
            "Value": "5.7.42"
          }
        ]
      }
    ],
    "Category": "security",
    "Source": "RDS",
    "TypeDetection": "***[resource-count] resources** are not running the
latest minor DB engine version",
    "TypeRecommendation": "Upgrade to latest engine version",
    "Impact": "Reduced database performance and data security at risk",
```

```

        "AdditionalInfo": "We recommend that you maintain your database with the
latest DB engine minor version as this version includes the latest security and
functionality fixes. The DB engine minor version upgrades contain only the changes
which are backward-compatible with earlier minor versions of the same major version
of the DB engine.",
        "Links": [
            {
                "Text": "Upgrading an RDS DB instance engine version",
                "Url": "https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
USER_UpgradeDBInstance.Upgrading.html"
            },
            {
                "Text": "Using Amazon RDS Blue/Green Deployments for database
updates for Amazon Aurora",
                "Url": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/blue-green-deployments.html"
            },
            {
                "Text": "Using Amazon RDS Blue/Green Deployments for database
updates for Amazon RDS",
                "Url": "https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
blue-green-deployments.html"
            }
        ]
    }
]
}

```

有关更多信息，请参阅[亚马逊 RDS 用户指南中的查看和回应 Amazon RDS 建议](#)和[亚马逊 Aurora 用户指南中的查看和回复亚马逊 RDS 建议](#)。

示例 4：列出所有有效的数据库建议

以下describe-db-recommendations示例列出了您 AWS 账户中所有有效的数据库推荐。

```
aws rds describe-db-recommendations \
  --filters Name=status,Values=active
```

输出：

```
{
  "DBRecommendations": [
    {
```

```
"RecommendationId": "12ab3cde-f456-7g8h-9012-i3j45678k9lm",
"TypeId": "config_recommendation::old_minor_version",
"Severity": "informational",
"ResourceArn": "arn:aws:rds:us-west-2:111122223333:db:database-1",
"Status": "active",
"CreatedTime": "2024-02-21T23:14:19.292000+00:00",
"UpdatedTime": "2024-02-21T23:14:19+00:00",
"Detection": "***[resource-name]** is not running the latest minor DB
engine version",
"Recommendation": "Upgrade to latest engine version",
>Description": "Your database resources aren't running the latest minor
DB engine version. The latest minor version contains the latest security fixes and
other improvements.",
"RecommendedActions": [
  {
    "ActionId": "12ab34c5de6fg7h89i0jk1lm234n5678",
    "Operation": "modifyDbInstance",
    "Parameters": [
      {
        "Key": "EngineVersion",
        "Value": "5.7.44"
      },
      {
        "Key": "DBInstanceIdentifier",
        "Value": "database-1"
      }
    ],
    "ApplyModes": [
      "immediately",
      "next-maintenance-window"
    ],
    "Status": "ready",
    "ContextAttributes": [
      {
        "Key": "Recommended value",
        "Value": "5.7.44"
      },
      {
        "Key": "Current engine version",
        "Value": "5.7.42"
      }
    ]
  }
],
```



```

    "Category": "security",
    "Source": "RDS",
    "TypeDetection": "**[resource-count] resources** are not running the
latest minor DB engine version",
    "TypeRecommendation": "Upgrade to latest engine version",
    "Impact": "Reduced database performance and data security at risk",
    "AdditionalInfo": "We recommend that you maintain your database with the
latest DB engine minor version as this version includes the latest security and
functionality fixes. The DB engine minor version upgrades contain only the changes
which are backward-compatible with earlier minor versions of the same major version
of the DB engine.",
    "Links": [
        {
            "Text": "Upgrading an RDS DB instance engine version",
            "Url": "https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
USER_UpgradeDBInstance.Upgrading.html"
        },
        {
            "Text": "Using Amazon RDS Blue/Green Deployments for database
updates for Amazon Aurora",
            "Url": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/blue-green-deployments.html"
        },
        {
            "Text": "Using Amazon RDS Blue/Green Deployments for database
updates for Amazon RDS",
            "Url": "https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
blue-green-deployments.html"
        }
    ]
}
]
}

```

有关更多信息，请参阅[亚马逊 RDS 用户指南中的查看和回应 Amazon RDS 建议](#)和[亚马逊 Aurora 用户指南中的查看和回复亚马逊 RDS 建议](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbRecommendations](#)中的。

## describe-db-security-groups

以下代码示例演示了如何使用 describe-db-security-groups。

## AWS CLI

### 列出数据库安全组

以下describe-db-security-groups示例列出了数据库安全组。

```
aws rds describe-db-security-groups
```

输出：

```
{
  "DBSecurityGroups": [
    {
      "OwnerId": "123456789012",
      "DBSecurityGroupName": "default",
      "DBSecurityGroupDescription": "default",
      "EC2SecurityGroups": [],
      "IPRanges": [],
      "DBSecurityGroupArn": "arn:aws:rds:us-west-1:111122223333:secgrp:default"
    },
    {
      "OwnerId": "123456789012",
      "DBSecurityGroupName": "mysecgroup",
      "DBSecurityGroupDescription": "My Test Security Group",
      "VpcId": "vpc-1234567f",
      "EC2SecurityGroups": [],
      "IPRanges": [],
      "DBSecurityGroupArn": "arn:aws:rds:us-west-1:111122223333:secgrp:mysecgroup"
    }
  ]
}
```

有关更多信息，请参阅 Amazon RDS 用户指南 [中的列出可用的数据库安全组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考 [DescribeDbSecurityGroups](#) 中的。

### describe-db-shard-groups

以下代码示例演示了如何使用 describe-db-shard-groups。

## AWS CLI

### 示例 1：描述数据库分片组

以下describe-db-shard-groups示例检索数据库分片组的详细信息。

```
aws rds describe-db-shard-groups
```

输出：

```
{
  "DBShardGroups": [
    {
      "DBShardGroupResourceId": "shardgroup-7bb446329da94788b3f957746example",
      "DBShardGroupIdentifier": "limitless-test-shard-grp",
      "DBClusterIdentifier": "limitless-test-cluster",
      "MaxACU": 768.0,
      "ComputeRedundancy": 0,
      "Status": "available",
      "PubliclyAccessible": true,
      "Endpoint": "limitless-test-cluster.limitless-cekycexample.us-east-2.rds.amazonaws.com"
    },
    {
      "DBShardGroupResourceId": "shardgroup-a6e3a0226aa243e2ac6c7a1234567890",
      "DBShardGroupIdentifier": "my-db-shard-group",
      "DBClusterIdentifier": "my-sv2-cluster",
      "MaxACU": 768.0,
      "ComputeRedundancy": 0,
      "Status": "available",
      "PubliclyAccessible": false,
      "Endpoint": "my-sv2-cluster.limitless-cekycexample.us-east-2.rds.amazonaws.com"
    }
  ]
}
```

有关更多信息，请参阅[亚马逊 Aurora 用户指南中的亚马逊 Aurora 数据库集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbShardGroups](#)中的。

## describe-db-snapshot-attributes

以下代码示例演示了如何使用 describe-db-snapshot-attributes。

### AWS CLI

描述数据库快照的属性名称和值

以下describe-db-snapshot-attributes示例描述了数据库快照的属性名称和值。

```
aws rds describe-db-snapshot-attributes \
  --db-snapshot-identifier mydbsnapshot
```

输出：

```
{
  "DBSnapshotAttributesResult": {
    "DBSnapshotIdentifier": "mydbsnapshot",
    "DBSnapshotAttributes": [
      {
        "AttributeName": "restore",
        "AttributeValues": [
          "123456789012",
          "210987654321"
        ]
      }
    ]
  }
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[共享数据库快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDbSnapshotAttributes](#)中的。

## describe-db-snapshots

以下代码示例演示了如何使用 describe-db-snapshots。

### AWS CLI

示例 1：描述数据库实例的数据库快照

以下 `describe-db-snapshots` 示例将检索有关数据库实例的数据库快照的详细信息。

```
aws rds describe-db-snapshots \  
  --db-snapshot-identifier mydbsnapshot
```

输出：

```
{  
  "DBSnapshots": [  
    {  
      "DBSnapshotIdentifier": "mydbsnapshot",  
      "DBInstanceIdentifier": "mysqldb",  
      "SnapshotCreateTime": "2018-02-08T22:28:08.598Z",  
      "Engine": "mysql",  
      "AllocatedStorage": 20,  
      "Status": "available",  
      "Port": 3306,  
      "AvailabilityZone": "us-east-1f",  
      "VpcId": "vpc-6594f31c",  
      "InstanceCreateTime": "2018-02-08T22:24:55.973Z",  
      "MasterUsername": "mysqladmin",  
      "EngineVersion": "5.6.37",  
      "LicenseModel": "general-public-license",  
      "SnapshotType": "manual",  
      "OptionGroupName": "default:mysql-5-6",  
      "PercentProgress": 100,  
      "StorageType": "gp2",  
      "Encrypted": false,  
      "DBSnapshotArn": "arn:aws:rds:us-  
east-1:123456789012:snapshot:mydbsnapshot",  
      "IAMDatabaseAuthenticationEnabled": false,  
      "ProcessorFeatures": [],  
      "DbiResourceId": "db-AKIAIOSFODNN7EXAMPLE"  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[创建数据库快照](#)。

示例 2：查找手动拍摄的快照数量

以下 `describe-db-snapshots` 示例使用 `--query` 选项中的 `length` 运算符来返回在特定 AWS 区域拍摄的手动快照的数量。

```
aws rds describe-db-snapshots \  
  --snapshot-type manual \  
  --query "length(*[].{DBSnapshots:SnapshotType})" \  
  --region eu-central-1
```

输出：

```
35
```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[创建数据库快照](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [DescribeDBSnapshots](#)。

## describe-db-subnet-groups

以下代码示例演示了如何使用 describe-db-subnet-groups。

### AWS CLI

描述数据库子网组

以下 describe-db-subnet-groups 示例检索指定数据库子网组的详细信息。

```
aws rds describe-db-subnet-groups
```

输出：

```
{  
  "DBSubnetGroups": [  
    {  
      "DBSubnetGroupName": "mydbsubnetgroup",  
      "DBSubnetGroupDescription": "My DB Subnet Group",  
      "VpcId": "vpc-971c12ee",  
      "SubnetGroupStatus": "Complete",  
      "Subnets": [  
        {  
          "SubnetIdentifier": "subnet-d8c8e7f4",  
          "SubnetAvailabilityZone": {  
            "Name": "us-east-1a"  
          },  
          "SubnetStatus": "Active"  
        },  
      ],  
    },  
  ],  
}
```

```

        {
            "SubnetIdentifier": "subnet-718fdc7d",
            "SubnetAvailabilityZone": {
                "Name": "us-east-1f"
            },
            "SubnetStatus": "Active"
        },
        {
            "SubnetIdentifier": "subnet-cbc8e7e7",
            "SubnetAvailabilityZone": {
                "Name": "us-east-1a"
            },
            "SubnetStatus": "Active"
        },
        {
            "SubnetIdentifier": "subnet-0ccde220",
            "SubnetAvailabilityZone": {
                "Name": "us-east-1a"
            },
            "SubnetStatus": "Active"
        }
    ],
    "DBSubnetGroupArn": "arn:aws:rds:us-
east-1:123456789012:subgrp:mydbsubnetgroup"
}
]
}

```

有关更多信息，请参阅[亚马逊 RDS 用户指南中的亚马逊 Virtual Private Cloud VPC 和 Amazon RDS](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeDbSubnetGroups](#)中的。

## describe-engine-default-cluster-parameters

以下代码示例演示了如何使用 describe-engine-default-cluster-parameters。

### AWS CLI

描述 Aurora 数据库引擎的默认引擎和系统参数信息

以下 describe-engine-default-cluster-parameters 示例检索兼容 MySQL 5.7 的 Aurora 数据库集群的默认引擎和系统参数信息的详细信息。

```
aws rds describe-engine-default-cluster-parameters \  
  --db-parameter-group-family aurora-mysql5.7
```

输出：

```
{  
  "EngineDefaults": {  
    "Parameters": [  
      {  
        "ParameterName": "aurora_load_from_s3_role",  
        "Description": "IAM role ARN used to load data from AWS S3",  
        "Source": "engine-default",  
        "ApplyType": "dynamic",  
        "DataType": "string",  
        "IsModifiable": true,  
        "SupportedEngineModes": [  
          "provisioned"  
        ]  
      },  
      ...some output truncated...  
    ]  
  }  
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的使用[数据库参数组](#)和[数据库集群参数组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeEngineDefaultClusterParameters](#)中的。

## describe-engine-default-parameters

以下代码示例演示了如何使用 describe-engine-default-parameters。

### AWS CLI

描述数据库引擎的默认引擎和系统参数信息

以下 describe-engine-default-parameters 示例检索 MySQL 5.7 数据库实例的默认引擎和系统参数信息的详细信息。

```
aws rds describe-engine-default-parameters \  
  --db-parameter-group-family mysql5.7
```



输出：

```
{
  "EngineDefaults": {
    "Parameters": [
      {
        "ParameterName": "allow-suspicious-udfs",
        "Description": "Controls whether user-defined functions that have
only an xxx symbol for the main function can be loaded",
        "Source": "engine-default",
        "ApplyType": "static",
        "DataType": "boolean",
        "AllowedValues": "0,1",
        "IsModifiable": false
      },
      ...some output truncated...
    ]
  }
}
```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[使用数据库参数组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEngineDefaultParameters](#)中的。

## describe-event-categories

以下代码示例演示了如何使用 describe-event-categories。

### AWS CLI

#### 描述事件类别

以下describe-event-categories示例检索有关所有可用事件源的事件类别的详细信息。

```
aws rds describe-event-categories
```

输出：

```
{
  "EventCategoriesMapList": [
    {
      "SourceType": "db-instance",
      "EventCategories": [
```

```
        "deletion",
        "read replica",
        "failover",
        "restoration",
        "maintenance",
        "low storage",
        "configuration change",
        "backup",
        "creation",
        "availability",
        "recovery",
        "failure",
        "backtrack",
        "notification"
    ]
},
{
    "SourceType": "db-security-group",
    "EventCategories": [
        "configuration change",
        "failure"
    ]
},
{
    "SourceType": "db-parameter-group",
    "EventCategories": [
        "configuration change"
    ]
},
{
    "SourceType": "db-snapshot",
    "EventCategories": [
        "deletion",
        "creation",
        "restoration",
        "notification"
    ]
},
{
    "SourceType": "db-cluster",
    "EventCategories": [
        "failover",
        "failure",
        "notification"
    ]
}
```

```
    ]
  },
  {
    "SourceType": "db-cluster-snapshot",
    "EventCategories": [
      "backup"
    ]
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEventCategories](#)中的。

## describe-event-subscriptions

以下代码示例演示了如何使用 describe-event-subscriptions。

### AWS CLI

#### 描述活动订阅

此示例描述了当前 AWS 账户的所有 Amazon RDS 事件订阅。

```
aws rds describe-event-subscriptions
```

输出：

```
{
  "EventSubscriptionsList": [
    {
      "EventCategoriesList": [
        "backup",
        "recovery"
      ],
      "Enabled": true,
      "EventSubscriptionArn": "arn:aws:rds:us-east-1:123456789012:es:my-
instance-events",
      "Status": "creating",
      "SourceType": "db-instance",
      "CustomerAwsId": "123456789012",
      "SubscriptionCreationTime": "2018-07-31 23:22:01.893",
      "CustSubscriptionId": "my-instance-events",
      "SnsTopicArn": "arn:aws:sns:us-east-1:123456789012:interesting-events"
```

```
    },
    ...some output truncated...
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEventSubscriptions](#)中的。

## describe-events

以下代码示例演示了如何使用 describe-events。

### AWS CLI

#### 描述事件

以下describe-events示例检索指定数据库实例发生的事件的详细信息。

```
aws rds describe-events \
  --source-identifier test-instance \
  --source-type db-instance
```

输出：

```
{
  "Events": [
    {
      "SourceType": "db-instance",
      "SourceIdentifier": "test-instance",
      "EventCategories": [
        "backup"
      ],
      "Message": "Backing up DB instance",
      "Date": "2018-07-31T23:09:23.983Z",
      "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
    },
    {
      "SourceType": "db-instance",
      "SourceIdentifier": "test-instance",
      "EventCategories": [
        "backup"
      ],
      "Message": "Finished DB Instance backup",
      "Date": "2018-07-31T23:15:13.049Z",
```

```

        "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEvents](#)中的。

## describe-export-tasks

以下代码示例演示了如何使用 describe-export-tasks。

### AWS CLI

#### 描述快照导出任务

以下describe-export-tasks示例返回有关将快照导出到 Amazon S3 的信息。

```
aws rds describe-export-tasks
```

输出：

```

{
  "ExportTasks": [
    {
      "ExportTaskIdentifier": "test-snapshot-export",
      "SourceArn": "arn:aws:rds:us-west-2:123456789012:snapshot:test-
snapshot",
      "SnapshotTime": "2020-03-02T18:26:28.163Z",
      "TaskStartTime": "2020-03-02T18:57:56.896Z",
      "TaskEndTime": "2020-03-02T19:10:31.985Z",
      "S3Bucket": "mybucket",
      "S3Prefix": "",
      "IamRoleArn": "arn:aws:iam::123456789012:role/service-role/ExportRole",
      "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/
abcd0000-7fca-4128-82f2-aabbccddeeff",
      "Status": "COMPLETE",
      "PercentProgress": 100,
      "TotalExtractedDataInGB": 0
    },
    {
      "ExportTaskIdentifier": "my-s3-export",
      "SourceArn": "arn:aws:rds:us-west-2:123456789012:snapshot:db5-snapshot-
test",

```

```

        "SnapshotTime": "2020-03-27T20:48:42.023Z",
        "S3Bucket": "mybucket",
        "S3Prefix": "",
        "IamRoleArn": "arn:aws:iam::123456789012:role/service-role/ExportRole",
        "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/
abcd0000-7fca-4128-82f2-aabbccddeeff",
        "Status": "STARTING",
        "PercentProgress": 0,
        "TotalExtractedDataInGB": 0
    }
]
}

```

有关更多信息，请参阅 Amazon RDS 用户指南中的[监控快照导出](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeExportTasks](#) 中的。

## describe-global-clusters

以下代码示例演示了如何使用 describe-global-clusters。

### AWS CLI

#### 描述全局数据库集群

以下 describe-global-clusters 示例列出了当前 AWS 区域中的 Aurora 全局数据库集群。

```
aws rds describe-global-clusters
```

输出：

```

{
  "GlobalClusters": [
    {
      "GlobalClusterIdentifier": "myglobalcluster",
      "GlobalClusterResourceId": "cluster-f5982077e3b5aabb",
      "GlobalClusterArn": "arn:aws:rds::123456789012:global-
cluster:myglobalcluster",
      "Status": "available",
      "Engine": "aurora-mysql",
      "EngineVersion": "5.7.mysql_aurora.2.07.2",
      "StorageEncrypted": false,
      "DeletionProtection": false,

```

```
        "GlobalClusterMembers": []
      }
    ]
  }
```

有关更多信息，请参阅亚马逊 Aurora 用户指南中的管理 Aurora [全球数据库](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeGlobalClusters](#)中的。

## describe-option-group-options

以下代码示例演示了如何使用 describe-option-group-options。

### AWS CLI

描述所有可用选项

以下describe-option-group-options示例列出了 Oracle 数据库 19c 实例的两个选项。

```
aws rds describe-option-group-options \
  --engine-name oracle-ee \
  --major-engine-version 19 \
  --max-items 2
```

输出：

```
{
  "OptionGroupOptions": [
    {
      "Name": "APEX",
      "Description": "Oracle Application Express Runtime Environment",
      "EngineName": "oracle-ee",
      "MajorEngineVersion": "19",
      "MinimumRequiredMinorEngineVersion": "0.0.0.ru-2019-07.rur-2019-07.r1",
      "PortRequired": false,
      "OptionsDependedOn": [],
      "OptionsConflictsWith": [],
      "Persistent": false,
      "Permanent": false,
      "RequiresAutoMinorEngineVersionUpgrade": false,
      "VpcOnly": false,
      "SupportsOptionVersionDowngrade": false,
      "OptionGroupOptionSettings": [],
    }
  ]
}
```

```

    "OptionGroupOptionVersions": [
      {
        "Version": "19.1.v1",
        "IsDefault": true
      },
      {
        "Version": "19.2.v1",
        "IsDefault": false
      }
    ]
  },
  {
    "Name": "APEX-DEV",
    "Description": "Oracle Application Express Development Environment",
    "EngineName": "oracle-ee",
    "MajorEngineVersion": "19",
    "MinimumRequiredMinorEngineVersion": "0.0.0.ru-2019-07.rur-2019-07.r1",
    "PortRequired": false,
    "OptionsDependedOn": [
      "APEX"
    ],
    "OptionsConflictsWith": [],
    "Persistent": false,
    "Permanent": false,
    "RequiresAutoMinorEngineVersionUpgrade": false,
    "VpcOnly": false,
    "OptionGroupOptionSettings": []
  }
],
"NextToken": "eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ=="
}

```

有关更多信息，请参阅 Amazon RDS 用户指南中的[列出选项组的选项和选项设置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeOptionGroupOptions](#)中的。

## describe-option-groups

以下代码示例演示了如何使用 describe-option-groups。

### AWS CLI

#### 描述可用的选项组



以下describe-option-groups示例列出了 Oracle 数据库 19c 实例的选项组。

```
aws rds describe-option-groups \  
  --engine-name oracle-ee \  
  --major-engine-version 19
```

输出：

```
{  
  "OptionGroupsList": [  
    {  
      "OptionGroupName": "default:oracle-ee-19",  
      "OptionGroupDescription": "Default option group for oracle-ee 19",  
      "EngineName": "oracle-ee",  
      "MajorEngineVersion": "19",  
      "Options": [],  
      "AllowsVpcAndNonVpcInstanceMemberships": true,  
      "OptionGroupArn": "arn:aws:rds:us-west-1:111122223333:og:default:oracle-  
ee-19"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[列出选项组的选项和选项设置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeOptionGroups](#)中的。

## describe-orderable-db-instance-options

以下代码示例演示了如何使用 describe-orderable-db-instance-options。

AWS CLI

描述可订购的数据库实例选项

以下 describe-orderable-db-instance-options 示例将检索有关运行 MySQL 数据库引擎的数据库实例的可订购选项的详细信息。

```
aws rds describe-orderable-db-instance-options \  
  --engine mysql
```

输出：

```
{
  "OrderableDBInstanceOptions": [
    {
      "MinStorageSize": 5,
      "ReadReplicaCapable": true,
      "MaxStorageSize": 6144,
      "AvailabilityZones": [
        {
          "Name": "us-east-1a"
        },
        {
          "Name": "us-east-1b"
        },
        {
          "Name": "us-east-1c"
        },
        {
          "Name": "us-east-1d"
        }
      ],
      "SupportsIops": false,
      "AvailableProcessorFeatures": [],
      "MultiAZCapable": true,
      "DBInstanceClass": "db.m1.large",
      "Vpc": true,
      "StorageType": "gp2",
      "LicenseModel": "general-public-license",
      "EngineVersion": "5.5.46",
      "SupportsStorageEncryption": false,
      "SupportsEnhancedMonitoring": true,
      "Engine": "mysql",
      "SupportsIAMDatabaseAuthentication": false,
      "SupportsPerformanceInsights": false
    }
  ]
  ...some output truncated...
}
```

- 有关 API 的详细信息，请参阅InstanceOptions 《AWS CLI 命令参考》中的 [DescribeOrderableDB](#)。

## describe-pending-maintenance-actions

以下代码示例演示了如何使用 `describe-pending-maintenance-actions`。

### AWS CLI

列出至少有一项待执行维护操作的资源

以下 `describe-pending-maintenance-actions` 示例列出了数据库实例的待处理维护操作。

```
aws rds describe-pending-maintenance-actions
```

输出：

```
{
  "PendingMaintenanceActions": [
    {
      "ResourceIdentifier": "arn:aws:rds:us-west-2:123456789012:cluster:global-db1-cl1",
      "PendingMaintenanceActionDetails": [
        {
          "Action": "system-update",
          "Description": "Upgrade to Aurora PostgreSQL 2.4.2"
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅 Amazon RDS 用户指南 [中的维护数据库实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribePendingMaintenanceActions](#) 中的。

## describe-reserved-db-instances-offerings

以下代码示例演示了如何使用 `describe-reserved-db-instances-offerings`。

### AWS CLI

描述预留数据库实例产品

以下 `describe-reserved-db-instances-offerings` 示例检索有关预留数据库实例选项的 `oracle` 详细信息。

```
aws rds describe-reserved-db-instances-offerings \  
--product-description oracle
```

输出：

```
{  
  "ReservedDBInstancesOfferings": [  
    {  
      "CurrencyCode": "USD",  
      "UsagePrice": 0.0,  
      "ProductDescription": "oracle-se2(li)",  
      "ReservedDBInstancesOfferingId": "005bdee3-9ef4-4182-aa0c-58ef7cb6c2f8",  
      "MultiAZ": true,  
      "DBInstanceClass": "db.m4.xlarge",  
      "OfferingType": "Partial Upfront",  
      "RecurringCharges": [  
        {  
          "RecurringChargeAmount": 0.594,  
          "RecurringChargeFrequency": "Hourly"  
        }  
      ],  
      "FixedPrice": 4089.0,  
      "Duration": 31536000  
    },  
    ...some output truncated...  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeReservedDbInstancesOfferings](#)中的。

## describe-reserved-db-instances

以下代码示例演示了如何使用 describe-reserved-db-instances。

### AWS CLI

#### 描述预留数据库实例

以下describe-reserved-db-instances示例检索有关当前 AWS 账户中所有预留数据库实例的详细信息。

```
aws rds describe-reserved-db-instances
```

输出：

```
{
  "ReservedDBInstances": [
    {
      "ReservedDBInstanceId": "myreservedinstance",
      "ReservedDBInstancesOfferingId": "12ab34cd-59af-4b2c-a660-1abcdef23456",
      "DBInstanceClass": "db.t3.micro",
      "StartTime": "2020-06-01T13:44:21.436Z",
      "Duration": 31536000,
      "FixedPrice": 0.0,
      "UsagePrice": 0.0,
      "CurrencyCode": "USD",
      "DBInstanceCount": 1,
      "ProductDescription": "sqlserver-ex(li)",
      "OfferingType": "No Upfront",
      "MultiAZ": false,
      "State": "payment-pending",
      "RecurringCharges": [
        {
          "RecurringChargeAmount": 0.014,
          "RecurringChargeFrequency": "Hourly"
        }
      ],
      "ReservedDBInstanceArn": "arn:aws:rds:us-west-2:123456789012:ri:myreservedinstance",
      "LeaseId": "a1b2c3d4-6b69-4a59-be89-5e11aa446666"
    }
  ]
}
```

有关更多信息，请参阅 Amazon RDS [用户指南中的 Amazon RDS 预留数据库实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeReservedDbInstances](#) 中的。

## describe-source-regions

以下代码示例演示了如何使用 describe-source-regions。

AWS CLI

描述来源区域

以下describe-source-regions示例检索有关所有来源 AWS 区域的详细信息。它还显示，自动备份只能从美国西部（俄勒冈）复制到目标 AWS 区域，即美国东部（弗吉尼亚北部）。

```
aws rds describe-source-regions \  
  --region us-east-1
```

输出：

```
{  
  "SourceRegions": [  
    {  
      "RegionName": "af-south-1",  
      "Endpoint": "https://rds.af-south-1.amazonaws.com",  
      "Status": "available",  
      "SupportsDBInstanceAutomatedBackupsReplication": false  
    },  
    {  
      "RegionName": "ap-east-1",  
      "Endpoint": "https://rds.ap-east-1.amazonaws.com",  
      "Status": "available",  
      "SupportsDBInstanceAutomatedBackupsReplication": false  
    },  
    {  
      "RegionName": "ap-northeast-1",  
      "Endpoint": "https://rds.ap-northeast-1.amazonaws.com",  
      "Status": "available",  
      "SupportsDBInstanceAutomatedBackupsReplication": true  
    },  
    {  
      "RegionName": "ap-northeast-2",  
      "Endpoint": "https://rds.ap-northeast-2.amazonaws.com",  
      "Status": "available",  
      "SupportsDBInstanceAutomatedBackupsReplication": true  
    },  
    {  
      "RegionName": "ap-northeast-3",  
      "Endpoint": "https://rds.ap-northeast-3.amazonaws.com",  
      "Status": "available",  
      "SupportsDBInstanceAutomatedBackupsReplication": false  
    },  
    {  
      "RegionName": "ap-south-1",  
      "Endpoint": "https://rds.ap-south-1.amazonaws.com",
```

```
    "Status": "available",
    "SupportsDBInstanceAutomatedBackupsReplication": true
  },
  {
    "RegionName": "ap-southeast-1",
    "Endpoint": "https://rds.ap-southeast-1.amazonaws.com",
    "Status": "available",
    "SupportsDBInstanceAutomatedBackupsReplication": true
  },
  {
    "RegionName": "ap-southeast-2",
    "Endpoint": "https://rds.ap-southeast-2.amazonaws.com",
    "Status": "available",
    "SupportsDBInstanceAutomatedBackupsReplication": true
  },
  {
    "RegionName": "ap-southeast-3",
    "Endpoint": "https://rds.ap-southeast-3.amazonaws.com",
    "Status": "available",
    "SupportsDBInstanceAutomatedBackupsReplication": false
  },
  {
    "RegionName": "ca-central-1",
    "Endpoint": "https://rds.ca-central-1.amazonaws.com",
    "Status": "available",
    "SupportsDBInstanceAutomatedBackupsReplication": true
  },
  {
    "RegionName": "eu-north-1",
    "Endpoint": "https://rds.eu-north-1.amazonaws.com",
    "Status": "available",
    "SupportsDBInstanceAutomatedBackupsReplication": true
  },
  {
    "RegionName": "eu-south-1",
    "Endpoint": "https://rds.eu-south-1.amazonaws.com",
    "Status": "available",
    "SupportsDBInstanceAutomatedBackupsReplication": false
  },
  {
    "RegionName": "eu-west-1",
    "Endpoint": "https://rds.eu-west-1.amazonaws.com",
    "Status": "available",
    "SupportsDBInstanceAutomatedBackupsReplication": true
  }
```

```
},
{
  "RegionName": "eu-west-2",
  "Endpoint": "https://rds.eu-west-2.amazonaws.com",
  "Status": "available",
  "SupportsDBInstanceAutomatedBackupsReplication": true
},
{
  "RegionName": "eu-west-3",
  "Endpoint": "https://rds.eu-west-3.amazonaws.com",
  "Status": "available",
  "SupportsDBInstanceAutomatedBackupsReplication": true
},
{
  "RegionName": "me-central-1",
  "Endpoint": "https://rds.me-central-1.amazonaws.com",
  "Status": "available",
  "SupportsDBInstanceAutomatedBackupsReplication": false
},
{
  "RegionName": "me-south-1",
  "Endpoint": "https://rds.me-south-1.amazonaws.com",
  "Status": "available",
  "SupportsDBInstanceAutomatedBackupsReplication": false
},
{
  "RegionName": "sa-east-1",
  "Endpoint": "https://rds.sa-east-1.amazonaws.com",
  "Status": "available",
  "SupportsDBInstanceAutomatedBackupsReplication": true
},
{
  "RegionName": "us-east-2",
  "Endpoint": "https://rds.us-east-2.amazonaws.com",
  "Status": "available",
  "SupportsDBInstanceAutomatedBackupsReplication": true
},
{
  "RegionName": "us-west-1",
  "Endpoint": "https://rds.us-west-1.amazonaws.com",
  "Status": "available",
  "SupportsDBInstanceAutomatedBackupsReplication": true
},
{
```



```
        "RegionName": "us-west-2",
        "Endpoint": "https://rds.us-west-2.amazonaws.com",
        "Status": "available",
        "SupportsDBInstanceAutomatedBackupsReplication": true
    }
]
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[查找有关复制备份的信息](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeSourceRegions](#)中的。

## describe-valid-db-instance-modifications

以下代码示例演示了如何使用 describe-valid-db-instance-modifications。

### AWS CLI

#### 描述数据库实例的有效修改

以下 describe-valid-db-instance-modifications 示例检索有关指定数据库实例的有效修改的详细信息。

```
aws rds describe-valid-db-instance-modifications \
    --db-instance-identifier test-instance
```

输出：

```
{
  "ValidDBInstanceModificationsMessage": {
    "ValidProcessorFeatures": [],
    "Storage": [
      {
        "StorageSize": [
          {
            "Step": 1,
            "To": 20,
            "From": 20
          },
          {
            "Step": 1,
            "To": 6144,
            "From": 22
          }
        ]
      }
    ]
  }
}
```

```
    }
  ],
  "ProvisionedIops": [
    {
      "Step": 1,
      "To": 0,
      "From": 0
    }
  ],
  "IopsToStorageRatio": [
    {
      "To": 0.0,
      "From": 0.0
    }
  ],
  "StorageType": "gp2"
},
{
  "StorageSize": [
    {
      "Step": 1,
      "To": 6144,
      "From": 100
    }
  ],
  "ProvisionedIops": [
    {
      "Step": 1,
      "To": 40000,
      "From": 1000
    }
  ],
  "IopsToStorageRatio": [
    {
      "To": 50.0,
      "From": 1.0
    }
  ],
  "StorageType": "io1"
},
{
  "StorageSize": [
    {
      "Step": 1,
```

```

        "To": 20,
        "From": 20
      },
      {
        "Step": 1,
        "To": 3072,
        "From": 22
      }
    ],
    "ProvisionedIops": [
      {
        "Step": 1,
        "To": 0,
        "From": 0
      }
    ],
    "IopsToStorageRatio": [
      {
        "To": 0.0,
        "From": 0.0
      }
    ],
    "StorageType": "magnetic"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeValidDbInstanceModifications](#)中的。

## download-db-log-file-portion

以下代码示例演示了如何使用 `download-db-log-file-portion`。

### AWS CLI

#### 下载数据库日志文件

以下`download-db-log-file-portion`示例仅下载日志文件的最新部分，将其保存到名为的本地文件中`tail.txt`。

```
aws rds download-db-log-file-portion \
  --db-instance-identifier test-instance \
```

```
--log-file-name log.txt \  
--output text > tail.txt
```

要下载整个文件，需要包含 `--starting-token 0` 参数。以下示例将输出保存到名为的本地文件中 `full.txt`。

```
aws rds download-db-log-file-portion \  
--db-instance-identifier test-instance \  
--log-file-name log.txt \  
--starting-token 0 \  
--output text > full.txt
```

保存的文件可能包含空行。下载时，它们会出现在日志文件每个部分的末尾。这通常不会给您的日志文件分析带来任何麻烦。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DownloadDbLogFilePortion](#)中的。

## generate-auth-token

以下代码示例演示了如何使用 `generate-auth-token`。

### AWS CLI

#### 生成身份验证令牌

以下 `generate-db-auth-token` 示例生成了用于 IAM 数据库身份验证的身份验证令牌。

```
aws rds generate-db-auth-token \  
--hostname aumysql-test.cdgmuiadpid.us-west-2.rds.amazonaws.com \  
--port 3306 \  
--region us-east-1 \  
--username jane_doe
```

输出：

```
aumysql-test.cdgmuiadpid.us-west-2.rds.amazonaws.com:3306/?  
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-  
Credential=AKIAIESZCNJ30EXAMPLE%2F20180731%2Fus-east-1%2Frds-db%2Faws4_request&X-  
Amz-Date=20180731T235209Z&X-Amz-Expires=900&X-Amz-SignedHeaders=host&X-Amz-  
Signature=5a8753ebEXAMPLEa2c724e5667797EXAMPLE9d6ec6e3f427191fa41aeEXAMPLE
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GenerateAuthToken](#)中的。

## generate-db-auth-token

以下代码示例演示了如何使用 generate-db-auth-token。

### AWS CLI

生成 IAM 身份验证令牌

以下generate-db-auth-token示例生成用于连接到数据库的 IAM 身份验证令牌。

```
aws rds generate-db-auth-token \  
  --hostname mydb.123456789012.us-east-1.rds.amazonaws.com \  
  --port 3306 \  
  --region us-east-1 \  
  --username db_user
```

输出：

```
mydb.123456789012.us-east-1.rds.amazonaws.com:3306/?  
Action=connect&DBUser=db_user&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-  
Credential=AKIAIEXAMPLE%2Fus-east-1%2Frds-db%2Faws4_request&X-Amz-  
Date=20210123T011543Z&X-Amz-Expires=900&X-Amz-SignedHeaders=host&X-Amz-  
Signature=88987EXAMPLE1EXAMPLE2EXAMPLE3EXAMPLE4EXAMPLE5EXAMPLE6
```

有关更多信息，请参阅 Amazon RD [S 用户指南中的使用 IAM 身份验证连接到数据库实例](#)和 [Amazon A urora 用户指南中的使用 IAM 身份验证连接到数据库集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GenerateDbAuthToken](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 list-tags-for-resource。

### AWS CLI

在 Amazon RDS 资源上列出标签

以下list-tags-for-resource示例列出了数据库实例上的所有标签。

```
aws rds list-tags-for-resource \  
  --resource-name arn:aws:rds:us-east-1:123456789012:db:orcl1
```

输出：

```
{
  "TagList": [
    {
      "Key": "Environment",
      "Value": "test"
    },
    {
      "Key": "Name",
      "Value": "MyDatabase"
    }
  ]
}
```

有关更多信息，请参阅 [Amazon RDS 用户指南中的为 Amazon RDS 资源添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## modify-certificates

以下代码示例演示了如何使用 modify-certificates。

### AWS CLI

临时替换新数据库实例的系统默认 SSL/TLS 证书

以下 modify-certificates 示例暂时覆盖新数据库实例的系统默认 SSL/TLS 证书。

```
aws rds modify-certificates \
  --certificate-identifier rds-ca-2019
```

输出：

```
{
  "Certificate": {
    "CertificateIdentifier": "rds-ca-2019",
    "CertificateType": "CA",
    "Thumbprint": "EXAMPLE123456789012",
    "ValidFrom": "2019-09-19T18:16:53Z",
    "ValidTill": "2024-08-22T17:08:50Z",
    "CertificateArn": "arn:aws:rds:us-east-1::cert:rds-ca-2019",
    "CustomerOverride": true,
  }
}
```

```
    "CustomerOverrideValidTill": "2024-08-22T17:08:50Z"  
  }  
}
```

有关更多信息，请参阅亚马逊 RDS 用户指南 [中的轮换 SSL/TLS 证书](#) 和亚马逊 Aurora 用户指南 [中的“轮换 SSL/ TLS 证书”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyCertificates](#) 中的。

## modify-current-db-cluster-capacity

以下代码示例演示了如何使用 `modify-current-db-cluster-capacity`。

### AWS CLI

扩展 Aurora 无服务器数据库集群的容量

以下 `modify-current-db-cluster-capacity` 示例将 Aurora 无服务器数据库集群的容量扩展到 8。

```
aws rds modify-current-db-cluster-capacity \  
  --db-cluster-identifier mydbcluster \  
  --capacity 8
```

输出：

```
{  
  "DBClusterIdentifier": "mydbcluster",  
  "PendingCapacity": 8,  
  "CurrentCapacity": 1,  
  "SecondsBeforeTimeout": 300,  
  "TimeoutAction": "ForceApplyCapacityChange"  
}
```

有关更多信息，请参阅 Amazon Aurora 用户 [指南中的手动扩展 Aurora Serverless v1 数据库集群容量](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyCurrentDbClusterCapacity](#) 中的。

## modify-db-cluster-endpoint

以下代码示例演示了如何使用 `modify-db-cluster-endpoint`。

## AWS CLI

### 修改自定义数据库集群终端节点

以下`modify-db-cluster-endpoint`示例修改了指定的自定义数据库集群终端节点。

```
aws rds modify-db-cluster-endpoint \  
  --db-cluster-endpoint-identifier mycustomendpoint \  
  --static-members dbinstance1 dbinstance2 dbinstance3
```

输出：

```
{  
  "DBClusterEndpointIdentifier": "mycustomendpoint",  
  "DBClusterIdentifier": "mydbcluster",  
  "DBClusterEndpointResourceIdentifier": "cluster-endpoint-ANPAJ4AE5446DAEXAMPLE",  
  "Endpoint": "mycustomendpoint.cluster-custom-cnpxample.us-  
east-1.rds.amazonaws.com",  
  "Status": "modifying",  
  "EndpointType": "CUSTOM",  
  "CustomEndpointType": "READER",  
  "StaticMembers": [  
    "dbinstance1",  
    "dbinstance2",  
    "dbinstance3"  
  ],  
  "ExcludedMembers": [],  
  "DBClusterEndpointArn": "arn:aws:rds:us-east-1:123456789012:cluster-  
endpoint:mycustomendpoint"  
}
```

有关更多信息，请参阅[亚马逊 Aurora 用户指南中的亚马逊 Aurora 连接管理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyDbClusterEndpoint](#)中的。

## `modify-db-cluster-parameter-group`

以下代码示例演示了如何使用 `modify-db-cluster-parameter-group`。

## AWS CLI

### 修改数据库集群参数组中的参数



以下`modify-db-cluster-parameter-group`示例修改了数据库集群参数组中的参数值。

```
aws rds modify-db-cluster-parameter-group \  
  --db-cluster-parameter-group-name mydbclusterpg \  
  --parameters  
  "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" \  
  
  "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

输出：

```
{  
  "DBClusterParameterGroupName": "mydbclusterpg"  
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的使用[数据库参数组和数据库集群参数组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyDbClusterParameterGroup](#)中的。

## modify-db-cluster-snapshot-attribute

以下代码示例演示了如何使用 `modify-db-cluster-snapshot-attribute`。

### AWS CLI

#### 修改数据库集群快照属性

以下`modify-db-cluster-snapshot-attribute`示例对指定的数据库集群快照属性进行了更改。

```
aws rds modify-db-cluster-snapshot-attribute \  
  --db-cluster-snapshot-identifier myclustersnapshot \  
  --attribute-name restore \  
  --values-to-add 123456789012
```

输出：

```
{  
  "DBClusterSnapshotAttributesResult": {  
    "DBClusterSnapshotIdentifier": "myclustersnapshot",  
    "DBClusterSnapshotAttributes": [  
      {
```

```

        "AttributeName": "restore",
        "AttributeValues": [
            "123456789012"
        ]
    }
]
}
}
}

```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[从数据库集群快照还原](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ModifyDbClusterSnapshotAttribute](#)中的。

## modify-db-cluster

以下代码示例演示了如何使用 modify-db-cluster。

### AWS CLI

#### 示例 1：修改数据库集群

以下 modify-db-cluster 示例更改名为的数据库集群的主用户密码，cluster-2 并将备份保留期设置为 14 天。该 --apply-immediately 参数使更改立即生效，而不是等到下一个维护时段。

```

aws rds modify-db-cluster \
  --db-cluster-identifier cluster-2 \
  --backup-retention-period 14 \
  --master-user-password newpassword99 \
  --apply-immediately

```

输出：

```

{
  "DBCluster": {
    "AllocatedStorage": 1,
    "AvailabilityZones": [
      "eu-central-1b",
      "eu-central-1c",
      "eu-central-1a"
    ],
    "BackupRetentionPeriod": 14,
    "DatabaseName": "",
    "DBClusterIdentifier": "cluster-2",

```

```
"DBClusterParameterGroup": "default.aurora5.6",
"DBSubnetGroup": "default-vpc-2305ca49",
"Status": "available",
"EarliestRestorableTime": "2020-06-03T02:07:29.637Z",
"Endpoint": "cluster-2.cluster-#####.eu-central-1.rds.amazonaws.com",
"ReaderEndpoint": "cluster-2.cluster-ro-#####.eu-
central-1.rds.amazonaws.com",
"MultiAZ": false,
"Engine": "aurora",
"EngineVersion": "5.6.10a",
"LatestRestorableTime": "2020-06-04T15:11:25.748Z",
"Port": 3306,
"MasterUsername": "admin",
"PreferredBackupWindow": "01:55-02:25",
"PreferredMaintenanceWindow": "thu:21:14-thu:21:44",
"ReadReplicaIdentifiers": [],
"DBClusterMembers": [
  {
    "DBInstanceIdentifier": "cluster-2-instance-1",
    "IsClusterWriter": true,
    "DBClusterParameterGroupStatus": "in-sync",
    "PromotionTier": 1
  }
],
"VpcSecurityGroups": [
  {
    "VpcSecurityGroupId": "sg-20a5c047",
    "Status": "active"
  }
],
"HostedZoneId": "Z1RLNU0EXAMPLE",
"StorageEncrypted": true,
"KmsKeyId": "arn:aws:kms:eu-central-1:123456789012:key/
d1bd7c8f-5cdb-49ca-8a62-a1b2c3d4e5f6",
"DbClusterResourceId": "cluster-AGJ7XI77XVIS6FUXHU1EXAMPLE",
"DBClusterArn": "arn:aws:rds:eu-central-1:123456789012:cluster:cluster-2",
"AssociatedRoles": [],
"IAMDatabaseAuthenticationEnabled": false,
"ClusterCreateTime": "2020-04-03T14:44:02.764Z",
"EngineMode": "provisioned",
"DeletionProtection": false,
"HttpEndpointEnabled": false,
"CopyTagsToSnapshot": true,
"CrossAccountClone": false,
```

```

    "DomainMemberships": []
  }
}

```

有关更多信息，请参阅[亚马逊 Aurora 用户指南中的修改亚马逊 Aurora 数据库集群](#)。

示例 2：将 VPC 安全组与数据库集群关联

以下 `modify-db-instance` 示例关联特定 VPC 安全组并从数据库集群中移除数据库安全组。

```

aws rds modify-db-cluster \
  --db-cluster-identifier dbName \
  --vpc-security-group-ids sg-ID

```

输出：

```

{
  "DBCluster": {
    "AllocatedStorage": 1,
    "AvailabilityZones": [
      "us-west-2c",
      "us-west-2b",
      "us-west-2a"
    ],
    "BackupRetentionPeriod": 1,
    "DBClusterIdentifier": "dbName",
    "DBClusterParameterGroup": "default.aurora-mysql8.0",
    "DBSubnetGroup": "default",
    "Status": "available",
    "EarliestRestorableTime": "2024-02-15T01:12:13.966000+00:00",
    "Endpoint": "dbName.cluster-abcdefghji.us-west-2.rds.amazonaws.com",
    "ReaderEndpoint": "dbName.cluster-ro-abcdefghji.us-
west-2.rds.amazonaws.com",
    "MultiAZ": false,
    "Engine": "aurora-mysql",
    "EngineVersion": "8.0.mysql_aurora.3.04.1",
    "LatestRestorableTime": "2024-02-15T02:25:33.696000+00:00",
    "Port": 3306,
    "MasterUsername": "admin",
    "PreferredBackupWindow": "10:59-11:29",
    "PreferredMaintenanceWindow": "thu:08:54-thu:09:24",
    "ReadReplicaIdentifiers": [],
    "DBClusterMembers": [

```

```
    {
      "DBInstanceIdentifier": "dbName-instance-1",
      "IsClusterWriter": true,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    }
  ],
  "VpcSecurityGroups": [
    {
      "VpcSecurityGroupId": "sg-ID",
      "Status": "active"
    }
  ],
  ...output omitted...
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的使用[安全组控制访问权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ModifyDbCluster](#)中的。

## modify-db-instance

以下代码示例演示了如何使用 modify-db-instance。

### AWS CLI

#### 示例 1：修改数据库实例

以下 modify-db-instance 示例将选项组和参数组与兼容的 Microsoft SQL Server 数据库实例关联起来。--apply-immediately 参数使选项和参数组立即关联，而不是等到下一个维护时段。

```
aws rds modify-db-instance \  
  --db-instance-identifier database-2 \  
  --option-group-name test-se-2017 \  
  --db-parameter-group-name test-sqlserver-se-2017 \  
  --apply-immediately
```

输出：

```
{  
  "DBInstance": {
```

```
"DBInstanceIdentifier": "database-2",
"DBInstanceClass": "db.r4.large",
"Engine": "sqlserver-se",
"DBInstanceStatus": "available",

...output omitted...

"DBParameterGroups": [
  {
    "DBParameterGroupName": "test-sqlserver-se-2017",
    "ParameterApplyStatus": "applying"
  }
],
"AvailabilityZone": "us-west-2d",

...output omitted...

"MultiAZ": true,
"EngineVersion": "14.00.3281.6.v1",
"AutoMinorVersionUpgrade": false,
"ReadReplicaDBInstanceIdentifiers": [],
"LicenseModel": "license-included",
"OptionGroupMemberships": [
  {
    "OptionGroupName": "test-se-2017",
    "Status": "pending-apply"
  }
],
"CharacterSetName": "SQL_Latin1_General_CP1_CI_AS",
"SecondaryAvailabilityZone": "us-west-2c",
"PubliclyAccessible": true,
"StorageType": "gp2",

...output omitted...

"DeletionProtection": false,
"AssociatedRoles": [],
"MaxAllocatedStorage": 1000
}
}
```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[修改 Amazon RDS 数据库实例](#)。

示例 2：将 VPC 安全组与数据库实例关联

以下 `modify-db-instance` 示例关联特定 VPC 安全组并从数据库实例中移除数据库安全组：

```
aws rds modify-db-instance \  
  --db-instance-identifier dbName \  
  --vpc-security-group-ids sg-ID
```

输出：

```
{  
  "DBInstance": {  
    "DBInstanceIdentifier": "dbName",  
    "DBInstanceClass": "db.t3.micro",  
    "Engine": "mysql",  
    "DBInstanceStatus": "available",  
    "MasterUsername": "admin",  
    "Endpoint": {  
      "Address": "dbName.abcdefghijkl.us-west-2.rds.amazonaws.com",  
      "Port": 3306,  
      "HostedZoneId": "ABCDEFGHIJK1234"  
    },  
    "AllocatedStorage": 20,  
    "InstanceCreateTime": "2024-02-15T00:37:58.793000+00:00",  
    "PreferredBackupWindow": "11:57-12:27",  
    "BackupRetentionPeriod": 7,  
    "DBSecurityGroups": [],  
    "VpcSecurityGroups": [  
      {  
        "VpcSecurityGroupId": "sg-ID",  
        "Status": "active"  
      }  
    ],  
    ... output omitted ...  
    "MultiAZ": false,  
    "EngineVersion": "8.0.35",  
    "AutoMinorVersionUpgrade": true,  
    "ReadReplicaDBInstanceIdentifiers": [],  
    "LicenseModel": "general-public-license",  
    ... output omitted ...  
  }  
}
```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[使用安全组控制访问权限](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [ModifyDBInstance](#)。

## modify-db-parameter-group

以下代码示例演示了如何使用 `modify-db-parameter-group`。

### AWS CLI

#### 修改数据库参数组

以下 `modify-db-parameter-group` 示例将更改数据库参数组中 `clr enabled` 参数的值。 `--apply-immediately` 参数使数据库参数组得到立即修改，而不是等到下一个维护时段。

```
aws rds modify-db-parameter-group \  
  --db-parameter-group-name test-sqlserver-se-2017 \  
  --parameters "ParameterName='clr  
enabled',ParameterValue=1,ApplyMethod=immediate"
```

输出：

```
{  
  "DBParameterGroupName": "test-sqlserver-se-2017"  
}
```

有关更多信息，请参阅《Amazon RDS 用户指南》中的 [修改数据库参数组中的参数](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》ParameterGroup 中的 [m odifyDB](#)。

## modify-db-proxy-endpoint

以下代码示例演示了如何使用 `modify-db-proxy-endpoint`。

### AWS CLI

#### 修改 RDS 数据库的数据库代理终端节点

以下 `modify-db-proxy-endpoint` 示例修改数据库代理终端节点，`proxyEndpoint` 将读取超时设置为 65 秒。

```
aws rds modify-db-proxy-endpoint \  
  --db-proxy-endpoint-name proxyEndpoint \  
  --cli-read-timeout 65
```



输出：

```
{
  "DBProxyEndpoint":
    {
      "DBProxyEndpointName": "proxyEndpoint",
      "DBProxyEndpointArn": "arn:aws:rds:us-east-1:123456789012:db-proxy-
endpoint:prx-endpoint-0123a01b12345c0ab",
      "DBProxyName": "proxyExample",
      "Status": "available",
      "VpcId": "vpc-1234567",
      "VpcSecurityGroupIds": [
        "sg-1234"
      ],
      "VpcSubnetIds": [
        "subnetgroup1",
        "subnetgroup2"
      ],
      "Endpoint": "proxyEndpoint.endpoint.proxyExample-ab0cd1efghij.us-
east-1.rds.amazonaws.com",
      "CreateDate": "2023-04-05T16:09:33.452000+00:00",
      "TargetRole": "READ_WRITE",
      "IsDefault": "false"
    }
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[修改代理终端节点](#)和 Amazon Aurora 用户指南中的[修改代理终端节点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyDbProxyEndpoint](#)中的。

## modify-db-proxy-target-group

以下代码示例演示了如何使用 modify-db-proxy-target-group。

### AWS CLI

#### 修改数据库代理终端节点

以下modify-db-proxy-target-group示例修改数据库代理目标组，将最大连接数设置为 80%，将最大空闲连接数设置为 10%。

```
aws rds modify-db-proxy-target-group \
```

```
--target-group-name default \  
--db-proxy-name proxyExample \  
--connection-pool-config MaxConnectionsPercent=80,MaxIdleConnectionsPercent=10
```

输出：

```
{  
  "DBProxyTargetGroup":  
    {  
      "DBProxyName": "proxyExample",  
      "TargetGroupName": "default",  
      "TargetGroupArn": "arn:aws:rds:us-east-1:123456789012:target-group:prx-  
tg-0123a01b12345c0ab",  
      "IsDefault": true,  
      "Status": "available",  
      "ConnectionPoolConfig": {  
        "MaxConnectionsPercent": 80,  
        "MaxIdleConnectionsPercent": 10,  
        "ConnectionBorrowTimeout": 120,  
        "SessionPinningFilters": []  
      },  
      "CreateDate": "2023-05-02T18:41:19.495000+00:00",  
      "UpdatedDate": "2023-05-02T18:41:21.762000+00:00"  
    }  
}
```

有关更多信息，请参阅 Amazon [RDS 用户指南中的修改 RDS 代理](#)和亚马逊 Aurora 用户指南中的 [修改 RDS 代理](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyDbProxyTargetGroup](#) 中的。

## modify-db-proxy

以下代码示例演示了如何使用 modify-db-proxy。

### AWS CLI

修改 RDS 数据库的数据库代理

以下 modify-db-proxy 示例修改了名为的数据库代理 proxyExample，使其连接需要 SSL。

```
aws rds modify-db-proxy \  
  --db-proxy-name proxyExample \  
  --ssl-mode require
```

```
--require-tls
```

输出：

```
{
  "DBProxy": {
    "DBProxyName": "proxyExample",
    "DBProxyArn": "arn:aws:rds:us-east-1:123456789012:db-proxy:prx-0123a01b12345c0ab",
    "Status": "modifying"
    "EngineFamily": "PostgreSQL",
    "VpcId": "sg-1234567",
    "VpcSecurityGroupIds": [
      "sg-1234"
    ],
    "VpcSubnetIds": [
      "subnetgroup1",
      "subnetgroup2"
    ],
    "Auth": "[
      {
        "Description": "proxydescription1",
        "AuthScheme": "SECRETS",
        "SecretArn": "arn:aws:secretsmanager:us-west-2:123456789123:secret:proxysecret1-Abcd1e",
        "IAMAuth": "DISABLED"
      }
    ]",
    "RoleArn": "arn:aws:iam::12345678912:role/ProxyPostgreSQLRole",
    "Endpoint": "proxyExample.proxy-ab0cd1efghij.us-east-1.rds.amazonaws.com",
    "RequireTLS": true,
    "IdleClientTimeout": 1800,
    "DebuggingLogging": false,
    "CreateDate": "2023-04-05T16:09:33.452000+00:00",
    "UpdateDate": "2023-04-13T01:49:38.568000+00:00"
  }
}
```

有关更多信息，请参阅 Amazon [RDS 用户指南中的修改 RDS 代理](#)和亚马逊 A [urora 用户指南中的创建 RDS 代理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyDbProxy](#)中的。

## modify-db-shard-group

以下代码示例演示了如何使用 `modify-db-shard-group`。

### AWS CLI

#### 示例 1：修改数据库分片组

以下 `modify-db-shard-group` 示例更改了数据库分片组的最大容量。

```
aws rds modify-db-shard-group \  
  --db-shard-group-identifier my-db-shard-group \  
  --max-acu 1000
```

输出：

```
{  
  "DBShardGroups": [  
    {  
      "DBShardGroupResourceId": "shardgroup-a6e3a0226aa243e2ac6c7a1234567890",  
      "DBShardGroupIdentifier": "my-db-shard-group",  
      "DBClusterIdentifier": "my-sv2-cluster",  
      "MaxACU": 768.0,  
      "ComputeRedundancy": 0,  
      "Status": "available",  
      "PubliclyAccessible": false,  
      "Endpoint": "my-sv2-cluster.limitless-cekyceexample.us-east-2.rds.amazonaws.com"  
    }  
  ]  
}
```

有关更多信息，请参阅[亚马逊 Aurora 用户指南中的亚马逊 Aurora 数据库集群](#)。

#### 示例 2：描述您的数据库分片组

以下 `describe-db-shard-groups` 示例在您运行命令后检索数据库分片组的 `modify-db-shard-group` 详细信息。现在，数据库分片组的最大容量 `my-db-shard-group` 为 1000 个 Aurora 容量单位 (ACU)。

```
aws rds describe-db-shard-groups
```

输出：

```
{
  "DBShardGroups": [
    {
      "DBShardGroupResourceId": "shardgroup-7bb446329da94788b3f957746example",
      "DBShardGroupIdentifier": "limitless-test-shard-grp",
      "DBClusterIdentifier": "limitless-test-cluster",
      "MaxACU": 768.0,
      "ComputeRedundancy": 0,
      "Status": "available",
      "PubliclyAccessible": true,
      "Endpoint": "limitless-test-cluster.limitless-cekycexample.us-east-2.rds.amazonaws.com"
    },
    {
      "DBShardGroupResourceId": "shardgroup-a6e3a0226aa243e2ac6c7a1234567890",
      "DBShardGroupIdentifier": "my-db-shard-group",
      "DBClusterIdentifier": "my-sv2-cluster",
      "MaxACU": 1000.0,
      "ComputeRedundancy": 0,
      "Status": "available",
      "PubliclyAccessible": false,
      "Endpoint": "my-sv2-cluster.limitless-cekycexample.us-east-2.rds.amazonaws.com"
    }
  ]
}
```

有关更多信息，请参阅[亚马逊 Aurora 用户指南中的亚马逊 Aurora 数据库集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyDbShardGroup](#)中的。

## modify-db-snapshot-attribute

以下代码示例演示了如何使用 modify-db-snapshot-attribute。

### AWS CLI

示例 1：允许两个 AWS 账户恢复数据库快照

以下modify-db-snapshot-attribute示例向标识符为111122223333和444455556666的两个 AWS 账户授予恢复名为的数据库快照的权限mydbsnapshot。

```
aws rds modify-db-snapshot-attribute \  
  --db-snapshot-identifier mydbsnapshot \  
  --attribute-name restore \  
  --values-to-add {"111122223333","444455556666"}
```

输出：

```
{  
  "DBSnapshotAttributesResult": {  
    "DBSnapshotIdentifier": "mydbsnapshot",  
    "DBSnapshotAttributes": [  
      {  
        "AttributeName": "restore",  
        "AttributeValues": [  
          "111122223333",  
          "444455556666"  
        ]  
      }  
    ]  
  }  
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[共享快照](#)。

示例 2：防止 AWS 账户还原数据库快照

以下modify-db-snapshot-attribute示例删除了特定 AWS 账户恢复名为的数据库快照的权限mydbsnapshot。指定单个账户时，账户标识符不能用引号或大括号括起来。

```
aws rds modify-db-snapshot-attribute \  
  --db-snapshot-identifier mydbsnapshot \  
  --attribute-name restore \  
  --values-to-remove 444455556666
```

输出：

```
{  
  "DBSnapshotAttributesResult": {  
    "DBSnapshotIdentifier": "mydbsnapshot",  
    "DBSnapshotAttributes": [  
      {  
        "AttributeName": "restore",
```

```

        "AttributeValues": [
            "111122223333"
        ]
    }
]
}
}

```

有关更多信息，请参阅 Amazon RDS 用户指南中的[共享快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ModifyDbSnapshotAttribute](#)中的。

## modify-db-snapshot-attributes

以下代码示例演示了如何使用 modify-db-snapshot-attributes。

### AWS CLI

#### 修改数据库快照属性

以下 modify-db-snapshot-attribute 示例允许两个 AWS 账户标识符 111122223333 和 444455556666 恢复名为 mydbsnapshot 的数据库快照。

```

aws rds modify-db-snapshot-attribute \
  --db-snapshot-identifier mydbsnapshot \
  --attribute-name restore \
  --values-to-add '["111122223333","444455556666"]'

```

输出：

```

{
  "DBSnapshotAttributesResult": {
    "DBSnapshotIdentifier": "mydbsnapshot",
    "DBSnapshotAttributes": [
      {
        "AttributeName": "restore",
        "AttributeValues": [
          "111122223333",
          "444455556666"
        ]
      }
    ]
  }
}

```

```
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[共享快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ModifyDbSnapshotAttributes](#)中的。

## modify-db-snapshot

以下代码示例演示了如何使用 modify-db-snapshot。

### AWS CLI

#### 修改数据库快照

以下 modify-db-snapshot 示例将名为 PostgreSQL 11.7 的 PostgreSQL 10.6 快照升级为 Po db5-snapshot-upg-test stgreSQL 11.7。新的数据库引擎版本将在快照完成升级且其状态变为可用后显示。

```
aws rds modify-db-snapshot \  
  --db-snapshot-identifier db5-snapshot-upg-test \  
  --engine-version 11.7
```

输出：

```
{  
  "DBSnapshot": {  
    "DBSnapshotIdentifier": "db5-snapshot-upg-test",  
    "DBInstanceIdentifier": "database-5",  
    "SnapshotCreateTime": "2020-03-27T20:49:17.092Z",  
    "Engine": "postgres",  
    "AllocatedStorage": 20,  
    "Status": "upgrading",  
    "Port": 5432,  
    "AvailabilityZone": "us-west-2a",  
    "VpcId": "vpc-2ff27557",  
    "InstanceCreateTime": "2020-03-27T19:59:04.735Z",  
    "MasterUsername": "postgres",  
    "EngineVersion": "10.6",  
    "LicenseModel": "postgresql-license",  
    "SnapshotType": "manual",  
    "OptionGroupName": "default:postgres-11",  
    "PercentProgress": 100,  
  }  
}
```



```

    "StorageType": "gp2",
    "Encrypted": false,
    "DBSnapshotArn": "arn:aws:rds:us-west-2:123456789012:snapshot:db5-snapshot-
upg-test",
    "IAMDatabaseAuthenticationEnabled": false,
    "ProcessorFeatures": [],
    "DbiResourceId": "db-GJMF75LM42IL6BTFRE4UZJ5YM4"
  }
}

```

有关更多信息，请参阅 Amazon [RDS 用户指南中的升级 PostgreSQL 数据库快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyDbSnapshot](#) 中的。

## modify-db-subnet-group

以下代码示例演示了如何使用 modify-db-subnet-group。

### AWS CLI

#### 修改数据库子网组

以下 modify-db-subnet-group 示例将 ID 为的子网添加到名为 subnet-08e41f9e230222222 为的数据库子网组 mysubnetgroup。要将现有子网保留在子网组中，请在 --subnet-ids 选项中包含其 ID 作为值。确保数据库子网组中至少有两个不同的可用区的子网。

```

aws rds modify-db-subnet-group \
  --db-subnet-group-name mysubnetgroup \
  --subnet-ids
  ["subnet-0a1dc4e1a6f123456","subnet-070dd7ecb3aaaaaaaa","subnet-00f5b198bc0abcdef","subnet-

```

输出：

```

{
  "DBSubnetGroup": {
    "DBSubnetGroupName": "mysubnetgroup",
    "DBSubnetGroupDescription": "test DB subnet group",
    "VpcId": "vpc-0f08e7610a1b2c3d4",
    "SubnetGroupStatus": "Complete",
    "Subnets": [
      {
        "SubnetIdentifier": "subnet-08e41f9e230222222",

```

```

        "SubnetAvailabilityZone": {
            "Name": "us-west-2a"
        },
        "SubnetStatus": "Active"
    },
    {
        "SubnetIdentifier": "subnet-070dd7ecb3aaaaaaaa",
        "SubnetAvailabilityZone": {
            "Name": "us-west-2b"
        },
        "SubnetStatus": "Active"
    },
    {
        "SubnetIdentifier": "subnet-00f5b198bc0abcdef",
        "SubnetAvailabilityZone": {
            "Name": "us-west-2d"
        },
        "SubnetStatus": "Active"
    },
    {
        "SubnetIdentifier": "subnet-0a1dc4e1a6f123456",
        "SubnetAvailabilityZone": {
            "Name": "us-west-2b"
        },
        "SubnetStatus": "Active"
    }
],
    "DBSubnetGroupArn": "arn:aws:rds:us-
west-2:534026745191:subgrp:mysubnetgroup"
}
}

```

有关更多信息，请参阅 Amazon RDS 用户指南中的[步骤 3：创建数据库子网组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ModifyDbSubnetGroup](#)中的。

## modify-event-subscription

以下代码示例演示了如何使用 modify-event-subscription。

### AWS CLI

#### 修改活动订阅

以下`modify-event-subscription`示例禁用了指定的事件订阅，使其不再向指定的 Amazon 简单通知服务主题发布通知。

```
aws rds modify-event-subscription \  
  --subscription-name my-instance-events \  
  --no-enabled
```

输出：

```
{  
  "EventSubscription": {  
    "EventCategoriesList": [  
      "backup",  
      "recovery"  
    ],  
    "CustomerAwsId": "123456789012",  
    "SourceType": "db-instance",  
    "SubscriptionCreationTime": "Tue Jul 31 23:22:01 UTC 2018",  
    "EventSubscriptionArn": "arn:aws:rds:us-east-1:123456789012:es:my-instance-  
events",  
    "SnsTopicArn": "arn:aws:sns:us-east-1:123456789012:interesting-events",  
    "CustSubscriptionId": "my-instance-events",  
    "Status": "modifying",  
    "Enabled": false  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyEventSubscription](#)中的。

## modify-global-cluster

以下代码示例演示了如何使用 `modify-global-cluster`。

### AWS CLI

#### 修改全局数据库集群

以下`modify-global-cluster`示例为兼容 Aurora MySQL 的全局数据库集群启用删除保护。

```
aws rds modify-global-cluster \  
  --global-cluster-identifier myglobalcluster \  
  --delete-protection-enabled
```

```
--deletion-protection
```

输出：

```
{
  "GlobalCluster": {
    "GlobalClusterIdentifier": "myglobalcluster",
    "GlobalClusterResourceId": "cluster-f0e523bfe07aabb",
    "GlobalClusterArn": "arn:aws:rds::123456789012:global-
cluster:myglobalcluster",
    "Status": "available",
    "Engine": "aurora-mysql",
    "EngineVersion": "5.7.mysql_aurora.2.07.2",
    "StorageEncrypted": false,
    "DeletionProtection": true,
    "GlobalClusterMembers": []
  }
}
```

有关更多信息，请参阅亚马逊 Aurora 用户指南中的管理 Aurora [全球数据库](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyGlobalCluster](#)中的。

## promote-read-replica-db-cluster

以下代码示例演示了如何使用 promote-read-replica-db-cluster。

AWS CLI

提升数据库集群只读副本

以下promote-read-replica-db-cluster示例将指定的只读副本提升为独立数据库集群。

```
aws rds promote-read-replica-db-cluster \
  --db-cluster-identifier mydbcluster-1
```

输出：

```
{
  "DBCluster": {
    "AllocatedStorage": 1,
```

```

    "AvailabilityZones": [
      "us-east-1a",
      "us-east-1b",
      "us-east-1c"
    ],
    "BackupRetentionPeriod": 1,
    "DatabaseName": "",
    "DBClusterIdentifier": "mydbcluster-1",
    ...some output truncated...
  }
}

```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[将只读副本提升为数据库集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PromoteReadReplicaDbCluster](#)中的。

## promote-read-replica

以下代码示例演示了如何使用 promote-read-replica。

### AWS CLI

#### 提升只读副本

以下promote-read-replica示例将指定的只读副本提升为独立数据库实例。

```

aws rds promote-read-replica \
  --db-instance-identifier test-instance-repl

```

输出：

```

{
  "DBInstance": {
    "DBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance-repl",
    "StorageType": "standard",
    "ReadReplicaSourceDBInstanceIdentifier": "test-instance",
    "DBInstanceStatus": "modifying",
    ...some output truncated...
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PromoteReadReplica](#)中的。

## purchase-reserved-db-instance

以下代码示例演示了如何使用 `purchase-reserved-db-instance`。

### AWS CLI

#### 购买预留数据库实例产品

以下 `purchase-reserved-db-instances-offering` 示例购买了预留数据库实例产品。`reserved-db-instances-offering-id` 必须是 `describe-reserved-db-instances-offering` 命令返回的有效产品编号。

```
aws rds-offering --id 438012d3-4a52-4cc7 purchase-reserved-db-instances-b2e3-8dff72
reserved-db-instances-offering e0e706
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PurchaseReservedDbInstance](#) 中的。

## purchase-reserved-db-instances-offerings

以下代码示例演示了如何使用 `purchase-reserved-db-instances-offerings`。

### AWS CLI

#### 示例 1：查找要购买的预留数据库实例

以下 `describe-reserved-db-instances-offerings` 示例列出了实例类为 `db.t2.micro` 且持续时间为一年的可用预留 MySQL 数据库实例。购买预留数据库实例需要提供产品 ID。

```
aws rds describe-reserved-db-instances-offerings \
  --product-description mysql \
  --db-instance-class db.t2.micro \
  --duration 1
```

输出：

```
{
  "ReservedDBInstancesOfferings": [
    {
      "ReservedDBInstancesOfferingId": "8ba30be1-b9ec-447f-8f23-6114e3f4c7b4",
      "DBInstanceClass": "db.t2.micro",
      "Duration": 31536000,
      "FixedPrice": 51.0,
```

```

        "UsagePrice": 0.0,
        "CurrencyCode": "USD",
        "ProductDescription": "mysql",
        "OfferingType": "Partial Upfront",
        "MultiAZ": false,
        "RecurringCharges": [
            {
                "RecurringChargeAmount": 0.006,
                "RecurringChargeFrequency": "Hourly"
            }
        ]
    },
    ... some output truncated ...
]
}

```

有关更多信息，请参阅 Amazon RDS [用户指南中的 Amazon RDS 预留数据库实例](#)。

#### 示例 2：购买预留数据库实例

以下 `purchase-reserved-db-instances-offering` 示例说明如何购买上一个示例中的预留数据库实例。

```
aws rds-offering --id 8ba30be1-purchase-reserved-db-instances b9ec-447f-8f23 reserved-db-instances-offering -6114e3f4c7b4
```

输出：

```

{
  "ReservedDBInstance": {
    "ReservedDBInstanceId": "ri-2020-06-29-16-54-57-670",
    "ReservedDBInstancesOfferingId": "8ba30be1-b9ec-447f-8f23-6114e3f4c7b4",
    "DBInstanceClass": "db.t2.micro",
    "StartTime": "2020-06-29T16:54:57.670Z",
    "Duration": 31536000,
    "FixedPrice": 51.0,
    "UsagePrice": 0.0,
    "CurrencyCode": "USD",
    "DBInstanceCount": 1,
    "ProductDescription": "mysql",
    "OfferingType": "Partial Upfront",
    "MultiAZ": false,
    "State": "payment-pending",
  }
}

```

```
    "RecurringCharges": [  
      {  
        "RecurringChargeAmount": 0.006,  
        "RecurringChargeFrequency": "Hourly"  
      }  
    ],  
    "ReservedDBInstanceArn": "arn:aws:rds:us-  
west-2:123456789012:ri:ri-2020-06-29-16-54-57-670"  
  }  
}
```

有关更多信息，请参阅 Amazon RDS [用户指南](#)中的 [Amazon RDS 预留数据库实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PurchaseReservedDbInstancesOfferings](#) 中的。

## reboot-db-instance

以下代码示例演示了如何使用 `reboot-db-instance`。

### AWS CLI

#### 重启数据库实例

以下 `reboot-db-instance` 示例开始重启指定的数据库实例。

```
aws rds reboot-db-instance \  
  --db-instance-identifier test-mysql-instance
```

输出：

```
{  
  "DBInstance": {  
    "DBInstanceIdentifier": "test-mysql-instance",  
    "DBInstanceClass": "db.t3.micro",  
    "Engine": "mysql",  
    "DBInstanceStatus": "rebooting",  
    "MasterUsername": "admin",  
    "Endpoint": {  
      "Address": "test-mysql-instance.#####.us-  
west-2.rds.amazonaws.com",  
      "Port": 3306,  
    }  
  }  
}
```



```
        "HostedZoneId": "Z1PVIF0EXAMPLE"
    },
    ... output omitted...
}
}
```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[重启数据库实例](#)。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的[RebootDBInstance](#)。

## reboot-db-shard-group

以下代码示例演示了如何使用 `reboot-db-shard-group`。

### AWS CLI

#### 示例 1：重启数据库分片组

以下 `reboot-db-shard-group` 示例重新启动数据库分片组。

```
aws rds reboot-db-shard-group \
  --db-shard-group-identifier my-db-shard-group
```

输出：

```
{
  "DBShardGroups": [
    {
      "DBShardGroupResourceId": "shardgroup-a6e3a0226aa243e2ac6c7a1234567890",
      "DBShardGroupIdentifier": "my-db-shard-group",
      "DBClusterIdentifier": "my-sv2-cluster",
      "MaxACU": 1000.0,
      "ComputeRedundancy": 0,
      "Status": "available",
      "PubliclyAccessible": false,
      "Endpoint": "my-sv2-cluster.limitless-cekycexample.us-east-2.rds.amazonaws.com"
    }
  ]
}
```

有关更多信息，请参阅[亚马逊 Aurora 用户指南中的重启亚马逊 Aurora 数据库集群或亚马逊 Aurora 数据库实例](#)。

## 示例 2：描述您的数据库分片组

以下describe-db-shard-groups示例在您运行命令后检索数据库分片组的reboot-db-shard-group详细信息。数据库分片组现在my-db-shard-group正在重新启动。

```
aws rds describe-db-shard-groups
```

输出：

```
{
  "DBShardGroups": [
    {
      "DBShardGroupResourceId": "shardgroup-7bb446329da94788b3f957746example",
      "DBShardGroupIdentifier": "limitless-test-shard-grp",
      "DBClusterIdentifier": "limitless-test-cluster",
      "MaxACU": 768.0,
      "ComputeRedundancy": 0,
      "Status": "available",
      "PubliclyAccessible": true,
      "Endpoint": "limitless-test-cluster.limitless-cekycexample.us-east-2.rds.amazonaws.com"
    },
    {
      "DBShardGroupResourceId": "shardgroup-a6e3a0226aa243e2ac6c7a1234567890",
      "DBShardGroupIdentifier": "my-db-shard-group",
      "DBClusterIdentifier": "my-sv2-cluster",
      "MaxACU": 1000.0,
      "ComputeRedundancy": 0,
      "Status": "rebooting",
      "PubliclyAccessible": false,
      "Endpoint": "my-sv2-cluster.limitless-cekycexample.us-east-2.rds.amazonaws.com"
    }
  ]
}
```

有关更多信息，请参阅[亚马逊 Aurora 用户指南中的重启亚马逊 Aurora 数据库集群或亚马逊 Aurora 数据库实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RebootDbShardGroup](#)中的。

## register-db-proxy-targets

以下代码示例演示了如何使用 register-db-proxy-targets。

### AWS CLI

向数据库注册数据库代理

以下register-db-proxy-targets示例在数据库和代理之间创建关联。

```
aws rds register-db-proxy-targets \  
  --db-proxy-name proxyExample \  
  --db-cluster-identifiers database-5
```

输出：

```
{  
  "DBProxyTargets": [  
    {  
      "RdsResourceId": "database-5",  
      "Port": 3306,  
      "Type": "TRACKED_CLUSTER",  
      "TargetHealth": {  
        "State": "REGISTERING"  
      }  
    },  
    {  
      "Endpoint": "database-5instance-1.ab0cd1efghij.us-  
east-1.rds.amazonaws.com",  
      "RdsResourceId": "database-5",  
      "Port": 3306,  
      "Type": "RDS_INSTANCE",  
      "TargetHealth": {  
        "State": "REGISTERING"  
      }  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon [RDS 用户指南中的创建 RDS 代理](#)和亚马逊 Aurora 用户指南中的 [创建 RDS 代理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterDbProxyTargets](#)中的。

## remove-from-global-cluster

以下代码示例演示了如何使用 `remove-from-global-cluster`。

### AWS CLI

将 Aurora 辅助集群与 Aurora 全局数据库集群分离

以下 `remove-from-global-cluster` 示例将 Aurora 辅助集群与 Aurora 全局数据库集群分离。集群从只读变为具有读写功能的独立集群。

```
aws rds remove-from-global-cluster \  
  --region us-west-2 \  
  --global-cluster-identifier myglobalcluster \  
  --db-cluster-identifier arn:aws:rds:us-west-2:123456789012:cluster:DB-1
```

输出：

```
{  
  "GlobalCluster": {  
    "GlobalClusterIdentifier": "myglobalcluster",  
    "GlobalClusterResourceId": "cluster-abc123def456gh",  
    "GlobalClusterArn": "arn:aws:rds::123456789012:global-  
cluster:myglobalcluster",  
    "Status": "available",  
    "Engine": "aurora-postgresql",  
    "EngineVersion": "10.11",  
    "StorageEncrypted": true,  
    "DeletionProtection": false,  
    "GlobalClusterMembers": [  
      {  
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:js-  
global-cluster",  
        "Readers": [  
          "arn:aws:rds:us-west-2:123456789012:cluster:DB-1"  
        ],  
        "IsWriter": true  
      },  
      {  
        "DBClusterArn": "arn:aws:rds:us-west-2:123456789012:cluster:DB-1",  
        "Readers": [],  
        "IsWriter": false,  
        "GlobalWriteForwardingStatus": "disabled"  
      }  
    ]  
  }  
}
```

```

    }
  ]
}

```

有关更多信息，请参阅亚马逊 Aurora [a 用户指南中的从 Amazon Aurora 全球数据库中移除集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveFromGlobalCluster](#)中的。

## remove-option-from-option-group

以下代码示例演示了如何使用 `remove-option-from-option-group`。

### AWS CLI

从选项组中删除选项

以下`remove-option-from-option-group`示例从中删除了该OEM选项`myoptiongroup`。

```

aws rds remove-option-from-option-group \
  --option-group-name myoptiongroup \
  --options OEM \
  --apply-immediately

```

输出：

```

{
  "OptionGroup": {
    "OptionGroupName": "myoptiongroup",
    "OptionGroupDescription": "Test",
    "EngineName": "oracle-ee",
    "MajorEngineVersion": "19",
    "Options": [],
    "AllowsVpcAndNonVpcInstanceMemberships": true,
    "OptionGroupArn": "arn:aws:rds:us-east-1:123456789012:og:myoptiongroup"
  }
}

```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[从选项组中移除选项](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveOptionFromOptionGroup](#)中的。

## remove-role-from-db-cluster

以下代码示例演示了如何使用 `remove-role-from-db-cluster`。

### AWS CLI

取消 AWS 身份和访问管理 (IAM) Access Management 角色与数据库集群的关联

以下 `remove-role-from-db-cluster` 示例从数据库集群中移除角色。

```
aws rds remove-role-from-db-cluster \  
  --db-cluster-identifier mydbcluster \  
  --role-arn arn:aws:iam::123456789012:role/RDSLoadFromS3
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 Aurora 用户指南中的 [“将 IAM 角色与亚马逊 Aurora MySQL 数据库集群关联”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RemoveRoleFromDbCluster](#) 中的。

## remove-role-from-db-instance

以下代码示例演示了如何使用 `remove-role-from-db-instance`。

### AWS CLI

取消 AWS 身份和访问管理 (IAM) Access Management 角色与数据库实例的关联

以下 `remove-role-from-db-instance` 示例 `rds-s3-integration-role` 从名为的 Oracle 数据库实例中删除名为的角色 `test-instance`。

```
aws rds remove-role-from-db-instance \  
  --db-instance-identifier test-instance \  
  --feature-name S3_INTEGRATION \  
  --role-arn arn:aws:iam::111122223333:role/rds-s3-integration-role
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon RDS 用户指南中的 [禁用 RDS SQL Server 与 S3 的集成](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveRoleFromDbInstance](#)中的。

## remove-source-identifier-from-subscription

以下代码示例演示了如何使用 `remove-source-identifier-from-subscription`。

### AWS CLI

从订阅中移除来源标识符

以下`remove-source-identifier`示例从现有订阅中删除指定的源标识符。

```
aws rds remove-source-identifier-from-subscription \  
  --subscription-name my-instance-events \  
  --source-identifier test-instance-repl
```

输出：

```
{  
  "EventSubscription": {  
    "EventSubscriptionArn": "arn:aws:rds:us-east-1:123456789012:es:my-instance-  
events",  
    "SubscriptionCreationTime": "Tue Jul 31 23:22:01 UTC 2018",  
    "EventCategoriesList": [  
      "backup",  
      "recovery"  
    ],  
    "SnsTopicArn": "arn:aws:sns:us-east-1:123456789012:interesting-events",  
    "Status": "modifying",  
    "CustSubscriptionId": "my-instance-events",  
    "CustomerAwsId": "123456789012",  
    "SourceIdsList": [  
      "test-instance"  
    ],  
    "SourceType": "db-instance",  
    "Enabled": false  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveSourceIdentifierFromSubscription](#)中的。

## remove-tags-from-resource

以下代码示例演示了如何使用 `remove-tags-from-resource`。

### AWS CLI

要从资源中删除标签

以下 `remove-tags-from-resource` 示例从资源中删除标签。

```
aws rds remove-tags-from-resource \  
  --resource-name arn:aws:rds:us-east-1:123456789012:db:mydbinstance \  
  --tag-keys Name Environment
```

此命令不生成任何输出。

有关更多信息，请参阅亚马逊 RDS 用户指南中的[为 Amazon RDS 资源添加标签](#)和亚马逊 Aurora 用户指南中的[标记 Amazon RDS 资源](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RemoveTagsFromResource](#)中的。

## reset-db-cluster-parameter-group

以下代码示例演示了如何使用 `reset-db-cluster-parameter-group`。

### AWS CLI

示例 1：将所有参数重置为其默认值

以下 `reset-db-cluster-parameter-group` 示例将客户创建的数据库集群参数组中的所有参数值重置为其默认值。

```
aws rds reset-db-cluster-parameter-group \  
  --db-cluster-parameter-group-name mydbclpg \  
  --reset-all-parameters
```

输出：

```
{  
  "DBClusterParameterGroupName": "mydbclpg"  
}
```



有关更多信息，请参阅 Amazon Aurora 用户指南中的使用[数据库参数组](#)和[数据库集群参数组](#)。

示例 2：将特定参数重置为其默认值

以下 `reset-db-cluster-parameter-group` 示例将特定参数的参数值重置为客户创建的数据库集群参数组中的默认值。

```
aws rds reset-db-cluster-parameter-group \  
  --db-cluster-parameter-group-name mydbclpgy \  
  --parameters "ParameterName=max_connections,ApplyMethod=immediate" \  
               "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

输出：

```
{  
  "DBClusterParameterGroupName": "mydbclpgy"  
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的使用[数据库参数组](#)和[数据库集群参数组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ResetDbClusterParameterGroup](#) 中的。

## reset-db-parameter-group

以下代码示例演示了如何使用 `reset-db-parameter-group`。

AWS CLI

示例 1：将所有参数重置为其默认值

以下 `reset-db-parameter-group` 示例将客户创建的数据库参数组中的所有参数值重置为其默认值。

```
aws rds reset-db-parameter-group \  
  --db-parameter-group-name mypg \  
  --reset-all-parameters
```

输出：

```
{  
  "DBParameterGroupName": "mypg"
```

```
}
```

有关更多信息，请参阅 Amazon RD S 用户指南中的[使用数据库参数组](#)和 Amazon Aurora 用户指南中的使用[数据库参数组](#)和[数据库集群参数组](#)。

示例 2：将特定参数重置为其默认值

以下 `reset-db-parameter-group` 示例将特定参数的参数值重置为客户创建的数据库参数组中的默认值。

```
aws rds reset-db-parameter-group \  
  --db-parameter-group-name mypg \  
  --parameters "ParameterName=max_connections,ApplyMethod=immediate" \  
               "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

输出：

```
{  
  "DBParameterGroupName": "mypg"  
}
```

有关更多信息，请参阅 Amazon RD S 用户指南中的[使用数据库参数组](#)和 Amazon Aurora 用户指南中的使用[数据库参数组](#)和[数据库集群参数组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ResetDbParameterGroup](#)中的。

## restore-db-cluster-from-s3

以下代码示例演示了如何使用 `restore-db-cluster-from-s3`。

### AWS CLI

从亚马逊 S3 恢复亚马逊 Aurora 数据库集群

以下 `restore-db-cluster-from-s3` 示例从亚马逊 S3 中的 MySQL 5.7 数据库备份文件恢复与 Amazon Aurora MySQL 版本 5.7 兼容的数据库集群。

```
aws rds restore-db-cluster-from-s3 \  
  --db-cluster-identifier cluster-s3-restore \  
  --engine aurora-mysql \  
  --master-username admin \  
  --master-userpassword mypassword
```

```
--master-user-password mypassword \  
--s3-bucket-name mybucket \  
--s3-prefix test-backup \  
--s3-ingestion-role-arn arn:aws:iam::123456789012:role/service-role/TestBackup \  
--source-engine mysql \  
--source-engine-version 5.7.28
```

输出：

```
{  
  "DBCluster": {  
    "AllocatedStorage": 1,  
    "AvailabilityZones": [  
      "us-west-2c",  
      "us-west-2a",  
      "us-west-2b"  
    ],  
    "BackupRetentionPeriod": 1,  
    "DBClusterIdentifier": "cluster-s3-restore",  
    "DBClusterParameterGroup": "default.aurora-mysql5.7",  
    "DBSubnetGroup": "default",  
    "Status": "creating",  
    "Endpoint": "cluster-s3-restore.cluster-co3xyzabc123.us-  
west-2.rds.amazonaws.com",  
    "ReaderEndpoint": "cluster-s3-restore.cluster-ro-co3xyzabc123.us-  
west-2.rds.amazonaws.com",  
    "MultiAZ": false,  
    "Engine": "aurora-mysql",  
    "EngineVersion": "5.7.12",  
    "Port": 3306,  
    "MasterUsername": "admin",  
    "PreferredBackupWindow": "11:15-11:45",  
    "PreferredMaintenanceWindow": "thu:12:19-thu:12:49",  
    "ReadReplicaIdentifiers": [],  
    "DBClusterMembers": [],  
    "VpcSecurityGroups": [  
      {  
        "VpcSecurityGroupId": "sg-#####",  
        "Status": "active"  
      }  
    ],  
    "HostedZoneId": "Z1PVIF0EXAMPLE",  
    "StorageEncrypted": false,
```

```

    "DbClusterResourceId": "cluster-SU5THYQQH0WCXZZDGXREXAMPLE",
    "DBClusterArn": "arn:aws:rds:us-west-2:123456789012:cluster:cluster-s3-
restore",
    "AssociatedRoles": [],
    "IAMDatabaseAuthenticationEnabled": false,
    "ClusterCreateTime": "2020-07-27T14:22:08.095Z",
    "EngineMode": "provisioned",
    "DeletionProtection": false,
    "HttpEndpointEnabled": false,
    "CopyTagsToSnapshot": false,
    "CrossAccountClone": false,
    "DomainMemberships": []
  }
}

```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[使用亚马逊 S3 存储桶从 MySQL 迁移数据](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的[RestoreDbClusterFromS3](#)。

## restore-db-cluster-from-snapshot

以下代码示例演示了如何使用 restore-db-cluster-from-snapshot。

### AWS CLI

#### 从快照还原数据库集群

以下内容从名为的数据库集群快照 restore-db-cluster-from-snapshot 恢复与 PostgreSQL 版本 10.7 兼容的 Aurora PostgreSQL 数据库集群。test-instance-snapshot

```

aws rds restore-db-cluster-from-snapshot \
  --db-cluster-identifier newdbcluster \
  --snapshot-identifier test-instance-snapshot \
  --engine aurora-postgresql \
  --engine-version 10.7

```

输出：

```

{
  "DBCluster": {
    "AllocatedStorage": 1,
    "AvailabilityZones": [
      "us-west-2c",

```

```
        "us-west-2a",
        "us-west-2b"
    ],
    "BackupRetentionPeriod": 7,
    "DatabaseName": "",
    "DBClusterIdentifier": "newdbcluster",
    "DBClusterParameterGroup": "default.aurora-postgresql10",
    "DBSubnetGroup": "default",
    "Status": "creating",
    "Endpoint": "newdbcluster.cluster-#####.us-west-2.rds.amazonaws.com",
    "ReaderEndpoint": "newdbcluster.cluster-ro-#####.us-
west-2.rds.amazonaws.com",
    "MultiAZ": false,
    "Engine": "aurora-postgresql",
    "EngineVersion": "10.7",
    "Port": 5432,
    "MasterUsername": "postgres",
    "PreferredBackupWindow": "09:33-10:03",
    "PreferredMaintenanceWindow": "sun:12:22-sun:12:52",
    "ReadReplicaIdentifiers": [],
    "DBClusterMembers": [],
    "VpcSecurityGroups": [
        {
            "VpcSecurityGroupId": "sg-#####",
            "Status": "active"
        }
    ],
    "HostedZoneId": "Z1PVIIF0EXAMPLE",
    "StorageEncrypted": true,
    "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/287364e4-33e3-4755-a3b0-
a1b2c3d4e5f6",
    "DbClusterResourceId": "cluster-5DSB5IFQDDUVAWOUWM1EXAMPLE",
    "DBClusterArn": "arn:aws:rds:us-west-2:123456789012:cluster:newdbcluster",
    "AssociatedRoles": [],
    "IAMDatabaseAuthenticationEnabled": false,
    "ClusterCreateTime": "2020-06-05T15:06:58.634Z",
    "EngineMode": "provisioned",
    "DeletionProtection": false,
    "HttpEndpointEnabled": false,
    "CopyTagsToSnapshot": false,
    "CrossAccountClone": false,
    "DomainMemberships": []
}
```

```
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[从数据库集群快照还原](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RestoreDbClusterFromSnapshot](#)中的。

## restore-db-cluster-to-point-in-time

以下代码示例演示了如何使用 `restore-db-cluster-to-point-in-time`。

### AWS CLI

将数据库集群还原到指定时间

以下 `restore-db-cluster-to-point-in-time` 示例将名为 `database-4` 的数据库集群恢复到尽可能晚的时间。使用 `copy-on-write` 作为还原类型可将新的数据库集群还原为源数据库集群的克隆。

```
aws rds restore-db-cluster-to-point-in-time \  
  --source-db-cluster-identifier database-4 \  
  --db-cluster-identifier sample-cluster-clone \  
  --restore-type copy-on-write \  
  --use-latest-restorable-time
```

输出：

```
{  
  "DBCluster": {  
    "AllocatedStorage": 1,  
    "AvailabilityZones": [  
      "us-west-2c",  
      "us-west-2a",  
      "us-west-2b"  
    ],  
    "BackupRetentionPeriod": 7,  
    "DatabaseName": "",  
    "DBClusterIdentifier": "sample-cluster-clone",  
    "DBClusterParameterGroup": "default.aurora-postgresql10",  
    "DBSubnetGroup": "default",  
    "Status": "creating",  
    "Endpoint": "sample-cluster-clone.cluster-#####.us-west-2.rds.amazonaws.com",
```

```

    "ReaderEndpoint": "sample-cluster-clone.cluster-ro-#####.us-
west-2.rds.amazonaws.com",
    "MultiAZ": false,
    "Engine": "aurora-postgresql",
    "EngineVersion": "10.7",
    "Port": 5432,
    "MasterUsername": "postgres",
    "PreferredBackupWindow": "09:33-10:03",
    "PreferredMaintenanceWindow": "sun:12:22-sun:12:52",
    "ReadReplicaIdentifiers": [],
    "DBClusterMembers": [],
    "VpcSecurityGroups": [
      {
        "VpcSecurityGroupId": "sg-#####",
        "Status": "active"
      }
    ],
    "HostedZoneId": "Z1PVIF0EXAMPLE",
    "StorageEncrypted": true,
    "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/287364e4-33e3-4755-a3b0-
a1b2c3d4e5f6",
    "DbClusterResourceId": "cluster-BIZ77GDSA2XBSTNPFW1EXAMPLE",
    "DBClusterArn": "arn:aws:rds:us-west-2:123456789012:cluster:sample-cluster-
clone",
    "AssociatedRoles": [],
    "IAMDatabaseAuthenticationEnabled": false,
    "CloneGroupId": "8d19331a-099a-45a4-b4aa-11aa22bb33cc44dd",
    "ClusterCreateTime": "2020-03-10T19:57:38.967Z",
    "EngineMode": "provisioned",
    "DeletionProtection": false,
    "HttpEndpointEnabled": false,
    "CopyTagsToSnapshot": false,
    "CrossAccountClone": false
  }
}

```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[将数据库集群恢复到指定时间](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RestoreDbClusterToPointInTime](#)中的。

## restore-db-instance-from-db-snapshot

以下代码示例演示了如何使用 `restore-db-instance-from-db-snapshot`。

## AWS CLI

### 从数据库快照还原数据库实例

以下 `restore-db-instance-from-db-snapshot` 示例使用指定数据库快照 `db7-new-instance` 中的数据库 `db.t3.small` 实例类创建一个名为 `db7-new-instance` 的新数据库实例。拍摄快照的源数据库实例使用已弃用的数据库实例类，因此您无法对其进行升级。

```
aws rds restore-db-instance-from-db-snapshot \
  --db-instance-identifier db7-new-instance \
  --db-snapshot-identifier db7-test-snapshot \
  --db-instance-class db.t3.small
```

输出：

```
{
  "DBInstance": {
    "DBInstanceIdentifier": "db7-new-instance",
    "DBInstanceClass": "db.t3.small",
    "Engine": "mysql",
    "DBInstanceStatus": "creating",

    ...output omitted...

    "PreferredMaintenanceWindow": "mon:07:37-mon:08:07",
    "PendingModifiedValues": {},
    "MultiAZ": false,
    "EngineVersion": "5.7.22",
    "AutoMinorVersionUpgrade": true,
    "ReadReplicaDBInstanceIdentifiers": [],
    "LicenseModel": "general-public-license",

    ...output omitted...

    "DBInstanceArn": "arn:aws:rds:us-west-2:123456789012:db:db7-new-instance",
    "IAMDatabaseAuthenticationEnabled": false,
    "PerformanceInsightsEnabled": false,
    "DeletionProtection": false,
    "AssociatedRoles": []
  }
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的 [从数据库快照恢复](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RestoreDbInstanceFromDbSnapshot](#)中的。

### restore-db-instance-from-s3

以下代码示例演示了如何使用 `restore-db-instance-from-s3`。

#### AWS CLI

从 Amazon S3 中的备份中恢复数据库实例

以下`restore-db-instance-from-s3`示例根据 `my-backups` S3 存储桶中的现有备份创建一个名为`restored-test-instance`的新数据库实例。

```
aws rds restore-db-instance-from-s3 \  
  --db-instance-identifier restored-test-instance \  
  --allocated-storage 250 --db-instance-class db.m4.large --engine mysql \  
  --master-username master --master-user-password secret99 \  
  --s3-bucket-name my-backups --s3-ingestion-role-arn \  
  arn:aws:iam::123456789012:role/my-role \  
  --source-engine mysql --source-engine-version 5.6.27
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [RestoreDbInstanceFromS3](#)。

### restore-db-instance-to-point-in-time

以下代码示例演示了如何使用 `restore-db-instance-to-point-in-time`。

#### AWS CLI

示例 1：将数据库实例还原到某个时间点

以下`restore-db-instance-to-point-in-time`示例在指定时间恢复`test-instance`到名为`restored-test-instance`的新数据库实例。

```
aws rds restore-db-instance-to-point-in-time \  
  --source-db-instance-identifier test-instance \  
  --target-db-instance restored-test-instance \  
  --restore-time 2018-07-30T23:45:00.000Z
```

输出：

```
{
  "DBInstance": {
    "AllocatedStorage": 20,
    "DBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:restored-test-
instance",
    "DBInstanceStatus": "creating",
    "DBInstanceIdentifier": "restored-test-instance",
    ...some output omitted...
  }
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的[将数据库实例恢复到指定时间](#)。

示例 2：将数据库实例从复制的备份恢复到指定时间

以下 `restore-db-instance-to-point-in-time` 示例从复制的自动备份将 Oracle 数据库实例恢复到指定时间。

```
aws rds restore-db-instance-to-point-in-time \
  --source-db-instance-automated-backups-arn "arn:aws:rds:us-
west-2:123456789012:auto-backup:ab-jkib2gfg5rv7replzadausbrktni2bn4example" \
  --target-db-instance-identifier myorclinstance-from-replicated-backup \
  --restore-time 2020-12-08T18:45:00.000Z
```

输出：

```
{
  "DBInstance": {
    "DBInstanceIdentifier": "myorclinstance-from-replicated-backup",
    "DBInstanceClass": "db.t3.micro",
    "Engine": "oracle-se2",
    "DBInstanceStatus": "creating",
    "MasterUsername": "admin",
    "DBName": "ORCL",
    "AllocatedStorage": 20,
    "PreferredBackupWindow": "07:45-08:15",
    "BackupRetentionPeriod": 14,
    ... some output omitted ...
    "DbiResourceId": "db-KGLXG75BGVIWKQT7NQ4EXAMPLE",
    "CACertificateIdentifier": "rds-ca-2019",
    "DomainMemberships": [],
    "CopyTagsToSnapshot": false,
```

```

    "MonitoringInterval": 0,
    "DBInstanceArn": "arn:aws:rds:us-west-2:123456789012:db:myorclinstance-from-
replicated-backup",
    "IAMDatabaseAuthenticationEnabled": false,
    "PerformanceInsightsEnabled": false,
    "DeletionProtection": false,
    "AssociatedRoles": [],
    "TagList": []
  }
}

```

有关更多信息，请参阅 Amazon RDS 用户指南中的[从复制的备份恢复到指定时间](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RestoreDbInstanceToPointInTime](#)中的。

## start-activity-stream

以下代码示例演示了如何使用 start-activity-stream。

### AWS CLI

#### 启动数据库活动流

以下 start-activity-stream 示例启动异步活动流以监控名为的 Aurora 集群 my-pg-cluster。

```

aws rds start-activity-stream \
  --region us-east-1 \
  --mode async \
  --kms-key-id arn:aws:kms:us-east-1:1234567890123:key/a12c345d-6ef7-890g-
h123-456i789jk011 \
  --resource-arn arn:aws:rds:us-east-1:1234567890123:cluster:my-pg-cluster \
  --apply-immediately

```

输出：

```

{
  "KmsKeyId": "arn:aws:kms:us-east-1:1234567890123:key/a12c345d-6ef7-890g-
h123-456i789jk011",
  "KinesisStreamName": "aws-rds-das-cluster-0ABCDEFGHI1JKLM2NOPQ3R4S",
  "Status": "starting",
  "Mode": "async",
  "ApplyImmediately": true
}

```

```
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[启动数据库活动流](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartActivityStream](#)中的。

## start-db-cluster

以下代码示例演示了如何使用 start-db-cluster。

### AWS CLI

#### 启动数据库集群

以下start-db-cluster示例启动数据库集群及其数据库实例。

```
aws rds start-db-cluster \  
  --db-cluster-identifier mydbcluster
```

输出：

```
{  
  "DBCluster": {  
    "AllocatedStorage": 1,  
    "AvailabilityZones": [  
      "us-east-1a",  
      "us-east-1e",  
      "us-east-1b"  
    ],  
    "BackupRetentionPeriod": 1,  
    "DatabaseName": "mydb",  
    "DBClusterIdentifier": "mydbcluster",  
    ...some output truncated...  
  }  
}
```

有关更多信息，请参阅亚马逊 [Aurora 用户指南中的停止和启动 Amazon Aurora 数据库集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartDbCluster](#)中的。

## start-db-instance-automated-backups-replication

以下代码示例演示了如何使用 start-db-instance-automated-backups-replication。

## AWS CLI

### 启用跨区域自动备份

以下 `start-db-instance-automated-backups-replication` 示例将自动备份从美国东部 ( 弗吉尼亚北部 ) 地区的数据库实例复制到美国西部 ( 俄勒冈 ) 。备份保留期为 14 天。

```
aws rds start-db-instance-automated-backups-replication \  
  --region us-west-2 \  
  --source-db-instance-arn "arn:aws:rds:us-east-1:123456789012:db:new-orcl-db" \  
  --backup-retention-period 14
```

输出 :

```
{  
  "DBInstanceAutomatedBackup": {  
    "DBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:new-orcl-db",  
    "DbiResourceId": "db-JKIB2GFQ5RV7REPLZA4EXAMPLE",  
    "Region": "us-east-1",  
    "DBInstanceIdentifier": "new-orcl-db",  
    "RestoreWindow": {},  
    "AllocatedStorage": 20,  
    "Status": "pending",  
    "Port": 1521,  
    "InstanceCreateTime": "2020-12-04T15:28:31Z",  
    "MasterUsername": "admin",  
    "Engine": "oracle-se2",  
    "EngineVersion": "12.1.0.2.v21",  
    "LicenseModel": "bring-your-own-license",  
    "OptionGroupName": "default:oracle-se2-12-1",  
    "Encrypted": false,  
    "StorageType": "gp2",  
    "IAMDatabaseAuthenticationEnabled": false,  
    "BackupRetentionPeriod": 14,  
    "DBInstanceAutomatedBackupsArn": "arn:aws:rds:us-west-2:123456789012:auto-backup:ab-jkib2gfq5rv7replzadabrktni2bn4example"  
  }  
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的 [启用跨区域自动备份](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartDbInstanceAutomatedBackupsReplication](#) 中的。

## start-db-instance

以下代码示例演示了如何使用 `start-db-instance`。

### AWS CLI

#### 启动数据库实例

以下 `start-db-instance` 示例启动指定的数据库实例。

```
aws rds start-db-instance \  
  --db-instance-identifier test-instance
```

输出：

```
{  
  "DBInstance": {  
    "DBInstanceStatus": "starting",  
    ...some output truncated...  
  }  
}
```

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 StartDbInstance](#) 中的。

## start-export-task

以下代码示例演示了如何使用 `start-export-task`。

### AWS CLI

#### 将快照导出到 Amazon S3

以下 `start-export-task` 示例将名为的数据库快照导出 `db5-snapshot-test` 到名为的 Amazon S3 存储桶 `mybucket`。

```
aws rds start-export-task \  
  --export-task-identifier my-s3-export \  
  --source-arn arn:aws:rds:us-west-2:123456789012:snapshot:db5-snapshot-test \  
  --s3-bucket-name mybucket \  
  --iam-role-arn arn:aws:iam::123456789012:role/service-role/ExportRole \  
  --kms-key-id arn:aws:kms:us-west-2:123456789012:key/abcd0000-7fca-4128-82f2-  
  aabbccddeeff
```

输出：

```
{
  "ExportTaskIdentifier": "my-s3-export",
  "SourceArn": "arn:aws:rds:us-west-2:123456789012:snapshot:db5-snapshot-test",
  "SnapshotTime": "2020-03-27T20:48:42.023Z",
  "S3Bucket": "mybucket",
  "IamRoleArn": "arn:aws:iam::123456789012:role/service-role/ExportRole",
  "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/abcd0000-7fca-4128-82f2-
aabbccddeeff",
  "Status": "STARTING",
  "PercentProgress": 0,
  "TotalExtractedDataInGB": 0
}
```

有关更多信息，请参阅 [Amazon RDS 用户指南中的将快照导出到 Amazon S3 存储桶](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartExportTask](#) 中的。

## stop-activity-stream

以下代码示例演示了如何使用 stop-activity-stream。

### AWS CLI

停止数据库活动流

以下 stop-activity-stream 示例停止名为的 Aurora 集群中的活动流 my-pg-cluster。

```
aws rds stop-activity-stream \
  --region us-east-1 \
  --resource-arn arn:aws:rds:us-east-1:1234567890123:cluster:my-pg-cluster \
  --apply-immediately
```

输出：

```
{
  "KmsKeyId": "arn:aws:kms:us-east-1:1234567890123:key/a12c345d-6ef7-890g-
h123-456i789jk0l1",
  "KinesisStreamName": "aws-rds-das-cluster-0ABCDEFGH11JKLM2NOPQ3R4S",
  "Status": "stopping"
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[停止活动流](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopActivityStream](#)中的。

## stop-db-cluster

以下代码示例演示了如何使用 stop-db-cluster。

### AWS CLI

#### 停止数据库集群

以下stop-db-cluster示例停止数据库集群及其数据库实例。

```
aws rds stop-db-cluster \  
  --db-cluster-identifier mydbcluster
```

输出：

```
{  
  "DBCluster": {  
    "AllocatedStorage": 1,  
    "AvailabilityZones": [  
      "us-east-1a",  
      "us-east-1e",  
      "us-east-1b"  
    ],  
    "BackupRetentionPeriod": 1,  
    "DatabaseName": "mydb",  
    "DBClusterIdentifier": "mydbcluster",  
    ...some output truncated...  
  }  
}
```

有关更多信息，请参阅亚马逊 [Aurora 用户指南中的停止和启动 Amazon Aurora 数据库集群](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopDbCluster](#)中的。

## stop-db-instance-automated-backups-replication

以下代码示例演示了如何使用 stop-db-instance-automated-backups-replication。



## AWS CLI

### 停止复制自动备份

以下是将自动备份复制到美国西部（俄勒冈）区域的操作 `stop-db-instance-automated-backups-replication` 结束。复制的备份将根据设定的备份保留期保留。

```
aws rds stop-db-instance-automated-backups-replication \
  --region us-west-2 \
  --source-db-instance-arn "arn:aws:rds:us-east-1:123456789012:db:new-orcl-db"
```

输出：

```
{
  "DBInstanceAutomatedBackup": {
    "DBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:new-orcl-db",
    "DbiResourceId": "db-JKIB2GFQ5RV7REPLZA4EXAMPLE",
    "Region": "us-east-1",
    "DBInstanceIdentifier": "new-orcl-db",
    "RestoreWindow": {
      "EarliestTime": "2020-12-04T23:13:21.030Z",
      "LatestTime": "2020-12-07T19:59:57Z"
    },
    "AllocatedStorage": 20,
    "Status": "replicating",
    "Port": 1521,
    "InstanceCreateTime": "2020-12-04T15:28:31Z",
    "MasterUsername": "admin",
    "Engine": "oracle-se2",
    "EngineVersion": "12.1.0.2.v21",
    "LicenseModel": "bring-your-own-license",
    "OptionGroupName": "default:oracle-se2-12-1",
    "Encrypted": false,
    "StorageType": "gp2",
    "IAMDatabaseAuthenticationEnabled": false,
    "BackupRetentionPeriod": 7,
    "DBInstanceAutomatedBackupsArn": "arn:aws:rds:us-west-2:123456789012:auto-backup:ab-jkib2gfgq5rv7replzadtausbrktni2bn4example"
  }
}
```

有关更多信息，请参阅 Amazon RDS 用户指南中的 [停止自动备份复制](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopDbInstanceAutomatedBackupsReplication](#)中的。

## stop-db-instance

以下代码示例演示了如何使用 stop-db-instance。

### AWS CLI

#### 停止数据库实例

以下stop-db-instance示例停止指定的数据库实例。

```
aws rds stop-db-instance \  
  --db-instance-identifier test-instance
```

输出：

```
{  
  "DBInstance": {  
    "DBInstanceStatus": "stopping",  
    ...some output truncated...  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopDbInstance](#)中的。

## switchover-blue-green-deployment

以下代码示例演示了如何使用 switchover-blue-green-deployment。

### AWS CLI

#### 示例 1：切换 RDS 数据库实例的蓝/绿部署

以下switchover-blue-green-deployment示例将指定的绿色环境提升为新的生产环境。

```
aws rds switchover-blue-green-deployment \  
  --blue-green-deployment-identifier bgd-wi89nwzglccsfake \  
  --switchover-timeout 300
```

输出：

```
{
  "BlueGreenDeployment": {
    "BlueGreenDeploymentIdentifier": "bgd-v53303651eexfake",
    "BlueGreenDeploymentName": "bgd-cli-test-instance",
    "Source": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance",
    "Target": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-green-blhile",
    "SwitchoverDetails": [
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-green-blhile",
        "Status": "AVAILABLE"
      },
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-replica-1",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-replica-1-green-k5fv7u",
        "Status": "AVAILABLE"
      },
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-replica-2",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-replica-2-green-ggsh8m",
        "Status": "AVAILABLE"
      },
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-replica-3",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance-replica-3-green-o2vwm0",
        "Status": "AVAILABLE"
      }
    ],
    "Tasks": [
      {
        "Name": "CREATING_READ_REPLICA_OF_SOURCE",
        "Status": "COMPLETED"
      }
    ]
  }
}
```

```

    {
      "Name": "DB_ENGINE_VERSION_UPGRADE",
      "Status": "COMPLETED"
    },
    {
      "Name": "CONFIGURE_BACKUPS",
      "Status": "COMPLETED"
    },
    {
      "Name": "CREATING_TOPOLOGY_OF_SOURCE",
      "Status": "COMPLETED"
    }
  ],
  "Status": "SWITCHOVER_IN_PROGRESS",
  "CreateTime": "2022-02-25T22:33:22.225000+00:00"
}

```

有关更多信息，请参阅 Amazon RDS 用户指南中的[切换蓝/绿部署](#)。

#### 示例 2：提升 Aurora MySQL 数据库集群的蓝/绿部署

以下 `switchover-blue-green-deployment` 示例将指定的绿色环境提升为新的生产环境。

```

aws rds switchover-blue-green-deployment \
  --blue-green-deployment-identifier bgd-wi89nwzglccsfake \
  --switchover-timeout 300

```

输出：

```

{
  "BlueGreenDeployment": {
    "BlueGreenDeploymentIdentifier": "bgd-wi89nwzglccsfake",
    "BlueGreenDeploymentName": "my-blue-green-deployment",
    "Source": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster",
    "Target": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster-green-3ud8z6",
    "SwitchoverDetails": [
      {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster",

```

```
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster:my-aurora-mysql-cluster-green-3ud8z6",
        "Status": "AVAILABLE"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-1",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-1-green-bvxc73",
        "Status": "AVAILABLE"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-2",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-2-green-7wc4ie",
        "Status": "AVAILABLE"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-3",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:db:my-aurora-mysql-cluster-3-green-p4xxkz",
        "Status": "AVAILABLE"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-excluded-member-endpoint",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-excluded-member-endpoint-green-nplikl",
        "Status": "AVAILABLE"
    },
    {
        "SourceMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-reader-endpoint",
        "TargetMember": "arn:aws:rds:us-east-1:123456789012:cluster-endpoint:my-reader-endpoint-green-miszlf",
        "Status": "AVAILABLE"
    }
],
"Tasks": [
    {
        "Name": "CREATING_READ_REPLICA_OF_SOURCE",
        "Status": "COMPLETED"
    }
]
```

```
    },
    {
      "Name": "DB_ENGINE_VERSION_UPGRADE",
      "Status": "COMPLETED"
    },
    {
      "Name": "CREATE_DB_INSTANCES_FOR_CLUSTER",
      "Status": "COMPLETED"
    },
    {
      "Name": "CREATE_CUSTOM_ENDPOINTS",
      "Status": "COMPLETED"
    }
  ],
  "Status": "SWITCHOVER_IN_PROGRESS",
  "CreateTime": "2022-02-25T22:38:49.522000+00:00"
}
}
```

有关更多信息，请参阅 Amazon Aurora 用户指南中的[切换蓝/绿部署](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[SwitchoverBlueGreenDeployment](#)中的。

## 使用 Amazon RDS 数据服务示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon RDS 数据服务配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### batch-execute-statement

以下代码示例演示了如何使用 batch-execute-statement。

#### AWS CLI

##### 执行批处理 SQL 语句

以下batch-execute-statement示例使用参数集对数据数组执行批处理 SQL 语句。

```
aws rds-data batch-execute-statement \  
  --resource-arn "arn:aws:rds:us-west-2:123456789012:cluster:mydbcluster" \  
  --database "mydb" \  
  --secret-arn "arn:aws:secretsmanager:us-west-2:123456789012:secret:mysecret" \  
  --sql "insert into mytable values (:id, :val)" \  
  --parameter-sets "[[{"name": "id", "value": {"longValue": 1}}, {"name": \  
"val", "value": {"stringValue": "ValueOne"}}], \  
  [{"name": "id", "value": {"longValue": 2}}, {"name": "val", \  
"value": {"stringValue": "ValueTwo"}}], \  
  [{"name": "id", "value": {"longValue": 3}}, {"name": "val", \  
"value": {"stringValue": "ValueThree"}}]]"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon RDS 用户指南》中的[使用 Aurora Serverless 数据 API](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchExecuteStatement](#)中的。

### begin-transaction

以下代码示例演示了如何使用 begin-transaction。

#### AWS CLI

##### 启动 SQL 事务

以下begin-transaction示例启动 SQL 事务。

```
aws rds-data begin-transaction \  
  --resource-arn "arn:aws:rds:us-west-2:123456789012:cluster:mydbcluster" \  
  --database "mydb" \  
  --secret-arn "arn:aws:secretsmanager:us-west-2:123456789012:secret:mysecret"
```

输出：

```
{
  "transactionId": "ABC1234567890xyz"
}
```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[使用 Aurora Serverless 数据 API](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BeginTransaction](#)中的。

## commit-transaction

以下代码示例演示了如何使用 commit-transaction。

AWS CLI

提交 SQL 事务

以下commit-transaction示例结束指定的 SQL 事务并提交您在其中所做的更改。

```
aws rds-data commit-transaction \
  --resource-arn "arn:aws:rds:us-west-2:123456789012:cluster:mydbcluster" \
  --secret-arn "arn:aws:secretsmanager:us-west-2:123456789012:secret:mysecret" \
  --transaction-id "ABC1234567890xyz"
```

输出：

```
{
  "transactionStatus": "Transaction Committed"
}
```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[使用 Aurora Serverless 数据 API](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CommitTransaction](#)中的。

## execute-statement

以下代码示例演示了如何使用 execute-statement。

AWS CLI

示例 1：执行属于事务的 SQL 语句



以下 `execute-statement` 示例运行属于事务的 SQL 语句。

```
aws rds-data execute-statement \  
  --resource-arn "arn:aws:rds:us-west-2:123456789012:cluster:mydbcluster" \  
  --database "mydb" \  
  --secret-arn "arn:aws:secretsmanager:us-west-2:123456789012:secret:mysecret" \  
  --sql "update mytable set quantity=5 where id=201" \  
  --transaction-id "ABC1234567890xyz"
```

输出：

```
{  
  "numberOfRecordsUpdated": 1  
}
```

示例 2：执行带有参数的 SQL 语句

以下 `execute-statement` 示例运行带有参数的 SQL 语句。

```
aws rds-data execute-statement \  
  --resource-arn "arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster" \  
  --database "mydb" \  
  --secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" \  
  --sql "insert into mytable values (:id, :val)" \  
  --parameters "[{\"name\": \"id\", \"value\": {\"longValue\": 1}}, {\"name\":  
  \"val\", \"value\": {\"stringValue\": \"value1\"}}]"
```

输出：

```
{  
  "numberOfRecordsUpdated": 1  
}
```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[使用 Aurora Serverless 数据 API](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ExecuteStatement](#)中的。

## rollback-transaction

以下代码示例演示了如何使用 `rollback-transaction`。

## AWS CLI

### 回滚 SQL 事务

以下rollback-transaction示例回滚指定的 SQL 事务。

```
aws rds-data rollback-transaction \  
  --resource-arn "arn:aws:rds:us-west-2:123456789012:cluster:mydbcluster" \  
  --secret-arn "arn:aws:secretsmanager:us-west-2:123456789012:secret:mysecret" \  
  --transaction-id "ABC1234567890xyz"
```

输出：

```
{  
  "transactionStatus": "Rollback Complete"  
}
```

有关更多信息，请参阅《Amazon RDS 用户指南》中的[使用 Aurora Serverless 数据 API](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RollbackTransaction](#)中的。

## 使用 Amazon RDS 性能见解示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon RDS Performance Insights 配合使用来执行操作和实施常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### describe-dimension-keys

以下代码示例演示了如何使用 describe-dimension-keys。

## AWS CLI

### 描述维度键

此示例请求所有等待事件的名称。数据按事件名称以及指定时间段内这些事件的汇总值进行汇总。

命令:

```
aws pi describe-dimension-keys --service-type RDS --identifier db-
LKCG0BK26374TPTDFX0IWCPPM --start-time 1527026400 --end-time 1527080400 --metric
db.load.avg --group-by '{"Group":"db.wait_event"}
```

输出:

```
{
  "AlignedEndTime": 1.5270804E9,
  "AlignedStartTime": 1.5270264E9,
  "Keys": [
    {
      "Dimensions": {"db.wait_event.name": "wait/synch/mutex/innodb/
aurora_lock_thread_slot_futex"},
      "Total": 0.05906906851195666
    },
    {
      "Dimensions": {"db.wait_event.name": "wait/io/aurora_redo_log_flush"},
      "Total": 0.015824722186149193
    },
    {
      "Dimensions": {"db.wait_event.name": "CPU"},
      "Total": 0.008014396230265477
    },
    {
      "Dimensions": {"db.wait_event.name": "wait/io/
aurora_respond_to_client"},
      "Total": 0.0036361612526204477
    },
    {
      "Dimensions": {"db.wait_event.name": "wait/io/table/sql/handler"},
      "Total": 0.0019108398419382965
    },
    {
      "Dimensions": {"db.wait_event.name": "wait/synch/cond/mysys/
my_thread_var::suspend"},
      "Total": 8.533847837782684E-4
    }
  ]
}
```

```

    },
    {
      "Dimensions": {"db.wait_event.name": "wait/io/file/csv/data"},
      "Total": 6.864181956477376E-4
    },
    {
      "Dimensions": {"db.wait_event.name": "Unknown"},
      "Total": 3.895887056379051E-4
    },
    {
      "Dimensions": {"db.wait_event.name": "wait/synch/mutex/sql/
FILE_AS_TABLE::LOCK_shim_lists"},
      "Total": 3.710368625122906E-5
    },
    {
      "Dimensions": {"db.wait_event.name": "wait/lock/table/sql/handler"},
      "Total": 0
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDimensionKeys](#)中的。

## get-resource-metrics

以下代码示例演示了如何使用 `get-resource-metrics`。

### AWS CLI

#### 获取资源指标

此示例为 `db.wait_event` 维度组以及该组中的 `db.wait_event.name` 维度请求数据点。在响应中，相关的数据点按请求的维度 (`db.wait_event.name`) 进行分组。

#### 命令:

```
aws pi get-resource-metrics --service-type RDS --identifier db-
LKCG0BK26374TPTDFX0IWVCPM --start-time 1527026400 --end-time 1527080400 --period-
in-seconds 300 --metric db.load.avg --metric-queries file://metric-queries.json
```

的参数存储 `--metric-queries` 在 JSON 文件中 `metric-queries.json`。以下是该文件的内容：

```
[
  {
    "Metric": "db.load.avg",
    "GroupBy": {
      "Group": "db.wait_event"
    }
  }
]
```

输出：

```
{
  "AlignedEndTime": 1.5270804E9,
  "AlignedStartTime": 1.5270264E9,
  "Identifier": "db-LKCG0BK26374TPTDFX0IWVCPM",
  "MetricList": [
    {
      "Key": {
        "Metric": "db.load.avg"
      },
      "DataPoints": [
        {
          "Timestamp": 1527026700.0,
          "Value": 1.3533333333333333
        },
        {
          "Timestamp": 1527027000.0,
          "Value": 0.88
        },
        <...remaining output omitted...>
      ]
    },
    {
      "Key": {
        "Metric": "db.load.avg",
        "Dimensions": {
          "db.wait_event.name": "wait/synch/mutex/innodb/aurora_lock_thread_slot_futex"
        }
      },
      "DataPoints": [
        {
          "Timestamp": 1527026700.0,
```

```
        "Value": 0.8566666666666667
      },
      {
        "Timestamp": 1527027000.0,
        "Value": 0.8633333333333333
      },
      <...remaining output omitted...>
    ],
  },
  <...remaining output omitted...>
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetResourceMetrics](#)中的。

## 使用 Amazon Redshift 示例 AWS CLI

以下代码示例向您展示了如何在 Amazon Redshift 中 AWS Command Line Interface 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **accept-reserved-node-exchange**

以下代码示例演示了如何使用 accept-reserved-node-exchange。

### AWS CLI

#### 接受预留节点交换

以下 `accept-reserved-node-exchange` 示例接受将 DC1 预留节点交换 DC2 预留节点。

```
aws redshift accept-reserved-node-exchange /
  --reserved-node-id 12345678-12ab-12a1-1a2a-12ab-12a12EXAMPLE /
  --target-reserved-node-offering-id 12345678-12ab-12a1-1a2a-12ab-12a12EXAMPLE
```

输出：

```
{
  "ExchangedReservedNode": {
    "ReservedNodeId": "12345678-12ab-12a1-1a2a-12ab-12a12EXAMPLE",
    "ReservedNodeOfferingId": "12345678-12ab-12a1-1a2a-12ab-12a12EXAMPLE",
    "NodeType": "dc2.large",
    "StartTime": "2019-12-06T21:17:26Z",
    "Duration": 31536000,
    "FixedPrice": 0.0,
    "UsagePrice": 0.0,
    "CurrencyCode": "USD",
    "NodeCount": 1,
    "State": "exchanging",
    "OfferingType": "All Upfront",
    "RecurringCharges": [
      {
        "RecurringChargeAmount": 0.0,
        "RecurringChargeFrequency": "Hourly"
      }
    ],
    "ReservedNodeOfferingType": "Regular"
  }
}
```

有关更多信息，请参阅 Amazon Redshift 集群管理指南中的 [AWS 使用 CLI 升级预留节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AcceptReservedNodeExchange](#) 中的。

## **authorize-cluster-security-group-ingress**

以下代码示例演示了如何使用 `authorize-cluster-security-group-ingress`。

### AWS CLI

授予对 EC2 安全 GroupThis 示例的访问权限即授权访问指定的 Amazon EC2 安全组。命令：

```
aws redshift authorize-cluster-security-group-ingress --cluster-security-group-name
mysecuritygroup --ec2-security-group-name myec2securitygroup --ec2-security-group-
owner-id 123445677890
```

授权 CIDR 范围的访问权限此示例授权访问 CIDR 范围。命令：

```
aws redshift authorize-cluster-security-group-ingress --cluster-security-group-name
mysecuritygroup --cidrip 192.168.100.100/32
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AuthorizeClusterSecurityGroupIngress](#)中的。

## authorize-snapshot-access

以下代码示例演示了如何使用 authorize-snapshot-access。

### AWS CLI

授权 AWS 账户还原 SnapshotThis 示例授权该 AWS 账户444455556666恢复快照my-snapshot-id。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift authorize-snapshot-access --snapshot-id my-snapshot-id --account-with-
restore-access 444455556666
```

结果：

```
{
  "Snapshot": {
    "Status": "available",
    "SnapshotCreateTime": "2013-07-17T22:04:18.947Z",
    "EstimatedSecondsToCompletion": 0,
    "AvailabilityZone": "us-east-1a",
    "ClusterVersion": "1.0",
    "MasterUsername": "adminuser",
    "Encrypted": false,
    "OwnerAccount": "111122223333",
    "BackupProgressInMegabytes": 11.0,
    "ElapsedTimeInSeconds": 0,
    "DBName": "dev",
    "CurrentBackupRateInMegabytesPerSecond": 0.1534,
    "ClusterCreateTime": "2013-01-22T21:59:29.559Z",
    "ActualIncrementalBackupSizeInMegabytes"; 11.0,
```



```
"SnapshotType": "manual",
"NodeType": "dw.hs1.xlarge",
"ClusterIdentifier": "mycluster",
"TotalBackupSizeInMegabytes": 20.0,
"Port": 5439,
"NumberOfNodes": 2,
"SnapshotIdentifier": "my-snapshot-id"
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AuthorizeSnapshotAccess](#)中的。

## batch-delete-cluster-snapshots

以下代码示例演示了如何使用 batch-delete-cluster-snapshots。

### AWS CLI

#### 删除一组群集快照

以下batch-delete-cluster-snapshots示例删除了一组手动集群快照。

```
aws redshift batch-delete-cluster-snapshots \
    --identifiers SnapshotIdentifier=mycluster-2019-11-06-14-12
    SnapshotIdentifier=mycluster-2019-11-06-14-20
```

输出：

```
{
  "Resources": [
    "mycluster-2019-11-06-14-12",
    "mycluster-2019-11-06-14-20"
  ]
}
```

有关更多信息，请参阅《[亚马逊 Redshift 集群管理指南](#)》中的[亚马逊 Redshift 快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchDeleteClusterSnapshots](#)中的。

## batch-modify-cluster-snapshots

以下代码示例演示了如何使用 batch-modify-cluster-snapshots。

## AWS CLI

### 修改一组群集快照

以下batch-modify-cluster-snapshots示例修改了一组集群快照的设置。

```
aws redshift batch-modify-cluster-snapshots \  
  --snapshot-identifier-list mycluster-2019-11-06-16-31 mycluster-2019-11-06-16-32 \  
  \  
  --manual-snapshot-retention-period 30
```

输出：

```
{  
  "Resources": [  
    "mycluster-2019-11-06-16-31",  
    "mycluster-2019-11-06-16-32"  
  ],  
  "Errors": [],  
  "ResponseMetadata": {  
    "RequestId": "12345678-12ab-12a1-1a2a-12ab-12a12EXAMPLE",  
    "HTTPStatusCode": 200,  
    "HTTPHeaders": {  
      "x-amzn-requestid": "12345678-12ab-12a1-1a2a-12ab-12a12EXAMPLE",  
      "content-type": "text/xml",  
      "content-length": "480",  
      "date": "Sat, 07 Dec 2019 00:36:09 GMT",  
      "connection": "keep-alive"  
    },  
    "RetryAttempts": 0  
  }  
}
```

有关更多信息，请参阅《[亚马逊 Redshift 集群管理指南](#)》中的[亚马逊 Redshift 快照](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchModifyClusterSnapshots](#)中的。

### cancel-resize

以下代码示例演示了如何使用 cancel-resize。

## AWS CLI

### 取消调整集群大小

以下cancel-resize示例取消了集群的经典调整大小操作。

```
aws redshift cancel-resize \  
  --cluster-identifier mycluster
```

输出：

```
{  
  "TargetNodeType": "dc2.large",  
  "TargetNumberOfNodes": 2,  
  "TargetClusterType": "multi-node",  
  "Status": "CANCELLING",  
  "ResizeType": "ClassicResize",  
  "TargetEncryptionType": "NONE"  
}
```

有关更多信息，请参阅《[亚马逊 Redshift 集群管理指南](#)》中的在 [Amazon Redshift 中调整集群的大小](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelResize](#)中的。

## copy-cluster-snapshot

以下代码示例演示了如何使用 copy-cluster-snapshot。

### AWS CLI

获取所有集群的描述 VersionsThis 示例返回所有集群版本的描述。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift copy-cluster-snapshot --source-snapshot-identifier  
  cm:examplecluster-2013-01-22-19-27-58 --target-snapshot-identifier my-saved-  
  snapshot-copy
```

结果：

```
{
```

```

"Snapshot": {
  "Status": "available",
  "SnapshotCreateTime": "2013-01-22T19:27:58.931Z",
  "AvailabilityZone": "us-east-1c",
  "ClusterVersion": "1.0",
  "MasterUsername": "adminuser",
  "DBName": "dev",
  "ClusterCreateTime": "2013-01-22T19:23:59.368Z",
  "SnapshotType": "manual",
  "NodeType": "dw.hs1.xlarge",
  "ClusterIdentifier": "examplecluster",
  "Port": 5439,
  "NumberOfNodes": "2",
  "SnapshotIdentifier": "my-saved-snapshot-copy"
},
"ResponseMetadata": {
  "RequestId": "3b279691-64e3-11e2-bec0-17624ad140dd"
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CopyClusterSnapshot](#)中的。

## create-cluster-parameter-group

以下代码示例演示了如何使用 create-cluster-parameter-group。

### AWS CLI

创建集群参数 GroupThis 示例创建新的集群参数组。命令：

```

aws redshift create-cluster-parameter-group --parameter-group-name
myclusterparametergroup --parameter-group-family redshift-1.0 --description "My
first cluster parameter group"

```

结果：

```

{
  "ClusterParameterGroup": {
    "ParameterGroupFamily": "redshift-1.0",
    "Description": "My first cluster parameter group",
    "ParameterGroupName": "myclusterparametergroup"
  },
}

```

```
"ResponseMetadata": {
  "RequestId": "739448f0-64cc-11e2-8f7d-3b939af52818"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateClusterParameterGroup](#)中的。

## create-cluster-security-group

以下代码示例演示了如何使用 `create-cluster-security-group`。

### AWS CLI

创建集群安全 GroupThis 示例会创建一个新的集群安全组。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift create-cluster-security-group --cluster-security-group-name
mysecuritygroup --description "This is my cluster security group"
```

结果：

```
{
  "create_cluster_security_group_response": {
    "create_cluster_security_group_result": {
      "cluster_security_group": {
        "description": "This is my cluster security group",
        "owner_id": "300454760768",
        "cluster_security_group_name": "mysecuritygroup",
        "ec2_security_groups": \[],
        "ip_ranges": \[]
      }
    },
    "response_metadata": {
      "request_id": "5df486a0-343a-11e2-b0d8-d15d0ef48549"
    }
  }
}
```

您也可以使用 `--output text` 选项以文本格式获取相同的信息。命令：

`--output text` 选项。命令：

选项。命令：

```
aws redshift create-cluster-security-group --cluster-security-group-name
mysecuritygroup --description "This is my cluster security group" --output text
```

结果：

```
This is my cluster security group 300454760768 mysecuritygroup
a0c0bfab-343a-11e2-95d2-c3dc9fe8ab57
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateClusterSecurityGroup](#)中的。

## create-cluster-snapshot

以下代码示例演示了如何使用 create-cluster-snapshot。

AWS CLI

创建集群 SnapshotThis 示例创建了新的集群快照。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift create-cluster-snapshot --cluster-identifier mycluster --snapshot-
identifier my-snapshot-id
```

结果：

```
{
  "Snapshot": {
    "Status": "creating",
    "SnapshotCreateTime": "2013-01-22T22:20:33.548Z",
    "AvailabilityZone": "us-east-1a",
    "ClusterVersion": "1.0",
    "MasterUsername": "adminuser",
    "DBName": "dev",
    "ClusterCreateTime": "2013-01-22T21:59:29.559Z",
    "SnapshotType": "manual",
    "NodeType": "dw.hs1.xlarge",
    "ClusterIdentifier": "mycluster",
    "Port": 5439,
    "NumberOfNodes": "2",
    "SnapshotIdentifier": "my-snapshot-id"
  },
}
```

```
"ResponseMetadata": {
  "RequestId": "f024d1a5-64e1-11e2-88c5-53eb05787dfb"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateClusterSnapshot](#)中的。

## create-cluster-subnet-group

以下代码示例演示了如何使用 create-cluster-subnet-group。

### AWS CLI

创建集群子网 GroupThis 示例创建新的集群子网组。命令：

```
aws redshift create-cluster-subnet-group --cluster-subnet-group-name mysubnetgroup
--description "My subnet group" --subnet-ids subnet-763fdd1c
```

结果：

```
{
  "ClusterSubnetGroup": {
    "Subnets": [
      {
        "SubnetStatus": "Active",
        "SubnetIdentifier": "subnet-763fdd1c",
        "SubnetAvailabilityZone": {
          "Name": "us-east-1a"
        }
      }
    ],
    "VpcId": "vpc-7e3fdd14",
    "SubnetGroupStatus": "Complete",
    "Description": "My subnet group",
    "ClusterSubnetGroupName": "mysubnetgroup"
  },
  "ResponseMetadata": {
    "RequestId": "500b8ce2-698f-11e2-9790-fd67517fb6fd"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateClusterSubnetGroup](#)中的。

## create-cluster

以下代码示例演示了如何使用 create-cluster。

### AWS CLI

使用最小值创建集群 ParametersThis 示例使用最少的参数集创建集群。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift create-cluster --node-type dw.hs1.xlarge --number-of-nodes 2 --master-username adminuser --master-user-password TopSecret1 --cluster-identifier mycluster
```

结果：

```
{
  "Cluster": {
    "NodeType": "dw.hs1.xlarge",
    "ClusterVersion": "1.0",
    "PubliclyAccessible": "true",
    "MasterUsername": "adminuser",
    "ClusterParameterGroups": [
      {
        "ParameterApplyStatus": "in-sync",
        "ParameterGroupName": "default.redshift-1.0"
      }
    ],
    "ClusterSecurityGroups": [
      {
        "Status": "active",
        "ClusterSecurityGroupName": "default"
      }
    ],
    "AllowVersionUpgrade": true,
    "VpcSecurityGroups": [],
    "PreferredMaintenanceWindow": "sat:03:30-sat:04:00",
    "AutomatedSnapshotRetentionPeriod": 1,
    "ClusterStatus": "creating",
    "ClusterIdentifier": "mycluster",
    "DBName": "dev",
    "NumberOfNodes": 2,
    "PendingModifiedValues": {
      "MasterUserPassword": "\*****"
    }
  },
  "ResponseMetadata": {
```



```
    "RequestId": "7cf4bcfc-64dd-11e2-bea9-49e0ce183f07"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCluster](#)中的。

## create-event-subscription

以下代码示例演示了如何使用 create-event-subscription。

### AWS CLI

为事件创建通知订阅

以下create-event-subscription示例创建了事件通知订阅。

```
aws redshift create-event-subscription \
  --subscription-name mysubscription \
  --sns-topic-arn arn:aws:sns:us-west-2:123456789012:MySNSStopic \
  --source-type cluster \
  --source-ids mycluster
```

输出：

```
{
  "EventSubscription": {
    "CustomerAwsId": "123456789012",
    "CustSubscriptionId": "mysubscription",
    "SnsTopicArn": "arn:aws:sns:us-west-2:123456789012:MySNSStopic",
    "Status": "active",
    "SubscriptionCreationTime": "2019-12-09T20:05:19.365Z",
    "SourceType": "cluster",
    "SourceIdsList": [
      "mycluster"
    ],
    "EventCategoriesList": [],
    "Severity": "INFO",
    "Enabled": true,
    "Tags": []
  }
}
```

有关更多信息，请参阅[亚马逊 Redshift 集群管理指南中的订阅亚马逊 Redshift 事件通知](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateEventSubscription](#)中的。

## create-hsm-client-certificate

以下代码示例演示了如何使用 create-hsm-client-certificate。

### AWS CLI

#### 创建 HSM 客户端证书

以下create-hsm-client-certificate示例生成一个 HSM 客户端证书，集群可以使用该证书连接到 HSM。

```
aws redshift create-hsm-client-certificate \
  --hsm-client-certificate-identifier myhsmclientcert
```

输出：

```
{
  "HsmClientCertificate": {
    "HsmClientCertificateIdentifier": "myhsmclientcert",
    "HsmClientCertificatePublicKey": "-----BEGIN CERTIFICATE-----
MIICiEXAMPLECQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMakGA1UEBhMC
VVMxCzAJBgNVBAgTEXAMPLEEwDgYDVQHEwdTZWF0dGx1MQ8wDQYDVQKKEwZBbWF6
b24xFDASBgNVBAwTC01BTSBDb25EXAMPLEIwEAYDVQQDEw1UZXR0Q21sYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb251QGfYXpvi5jb2EXAMPLETEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMakGA1UEBhMCVVMxCzAJBgNVBEXAMPLEMRAwDgYD
EXAMPLETZWF0dGx1MQ8wDQYDVQKKEwZBbWF6b24xFDASBgNVBAwTC01BTSBDb25z
b2x1MRIwEAEXAMPLEEw1UZXR0Q21sYWMxHzAdBgkqhkiG9w0BCQEWEG5vb251QGfY
Xpvi5jb20wgZ8wDQYJKEXAMPLEAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvYSwTc2XADZ4nB+BLyGVIk6EXAMPLE3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugEXAMPLEzZswY6786m86gpE
Ibb30hjZnzcvQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEEXAMPLEEAtCu4
nUhVVxYUEXAMPLEh8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GEXAMPLE10ZxBHjJnyp3780D8uTs7fLvJx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rEXAMPLE=-----END CERTIFICATE-----\n",
    "Tags": []
  }
}
```

有关更多信息，请参阅[亚马逊 Redshift 集群管理指南中的亚马逊 Redshift API 权限参考](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateHsmClientCertificate](#)中的。

## create-hsm-configuration

以下代码示例演示了如何使用 create-hsm-configuration。

### AWS CLI

#### 创建 HSM 配置

以下create-hsm-configuration示例创建指定的 HSM 配置，其中包含集群在硬件安全模块 (HSM) 中存储和使用数据库加密密钥所需的信息。

```
aws redshift create-hsm-configuration /
  --hsm-configuration-identifier myhsmconnection
  --description "My HSM connection"
  --hsm-ip-address 192.0.2.09
  --hsm-partition-name myhsmpartition /
  --hsm-partition-password A1b2c3d4 /
  --hsm-server-public-certificate myhsmclientcert
```

输出：

```
{
  "HsmConfiguration": {
    "HsmConfigurationIdentifier": "myhsmconnection",
    "Description": "My HSM connection",
    "HsmIpAddress": "192.0.2.09",
    "HsmPartitionName": "myhsmpartition",
    "Tags": []
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateHsmConfiguration](#)中的。

## create-snapshot-copy-grant

以下代码示例演示了如何使用 create-snapshot-copy-grant。

### AWS CLI

要创建快照副本，请授予

以下 `create-snapshot-copy-grant` 示例创建快照副本授权，并对目标 AWS 区域中复制的快照进行加密。

```
aws redshift create-snapshot-copy-grant \  
  --snapshot-copy-grant-name mysnapshotcopygrantname
```

输出：

```
{  
  "SnapshotCopyGrant": {  
    "SnapshotCopyGrantName": "mysnapshotcopygrantname",  
    "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/  
bPxRfih3yCo8nvbEXAMPLEKEY",  
    "Tags": []  
  }  
}
```

有关更多信息，请参阅 [《亚马逊 Redshift 集群管理指南》](#) 中的 [亚马逊 Redshift 数据库加密](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateSnapshotCopyGrant](#) 中的。

## create-snapshot-schedule

以下代码示例演示了如何使用 `create-snapshot-schedule`。

### AWS CLI

#### 创建快照计划

以下 `create-snapshot-schedule` 示例使用指定的描述和每 12 小时的速率创建快照计划。

```
aws redshift create-snapshot-schedule \  
  --schedule-definitions "rate(12 hours)" \  
  --schedule-identifier mysnapshotschedule \  
  --schedule-description "My schedule description"
```

输出：

```
{  
  "ScheduleDefinitions": [  
    "rate(12 hours)"  
  ]  
}
```

```
    ],  
    "ScheduleIdentifier": "mysnapshotschedule",  
    "ScheduleDescription": "My schedule description",  
    "Tags": []  
  }  
}
```

有关更多信息，请参阅 Amazon Redshift 集群管理指南中的[自动快照计划](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateSnapshotSchedule](#)中的。

## create-tags

以下代码示例演示了如何使用 create-tags。

### AWS CLI

为集群创建标签

以下 create-tags 示例将指定的标签键/值对添加到指定的集群。

```
aws redshift create-tags \  
  --resource-name arn:aws:redshift:us-west-2:123456789012:cluster:mycluster \  
  --tags "Key"="mytags","Value"="tag1"
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊 Redshift [集群管理指南](#)》中的在 Amazon Redshift 中为资源添加[标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateTags](#)中的。

## delete-cluster-parameter-group

以下代码示例演示了如何使用 delete-cluster-parameter-group。

### AWS CLI

删除集群参数 GroupThis 示例删除集群参数组。命令：

```
aws redshift delete-cluster-parameter-group --parameter-group-name  
myclusterparametergroup
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteClusterParameterGroup](#)中的。

## delete-cluster-security-group

以下代码示例演示了如何使用 delete-cluster-security-group。

### AWS CLI

删除集群安全 GroupThis 示例删除集群安全组。命令：

```
aws redshift delete-cluster-security-group --cluster-security-group-name
mysecuritygroup
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteClusterSecurityGroup](#)中的。

## delete-cluster-snapshot

以下代码示例演示了如何使用 delete-cluster-snapshot。

### AWS CLI

删除集群 SnapshotThis 示例删除集群快照。命令：

```
aws redshift delete-cluster-snapshot --snapshot-identifier my-snapshot-id
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteClusterSnapshot](#)中的。

## delete-cluster-subnet-group

以下代码示例演示了如何使用 delete-cluster-subnet-group。

### AWS CLI

删除集群子网 GroupThis 示例删除集群子网组。命令：

```
aws redshift delete-cluster-subnet-group --cluster-subnet-group-name mysubnetgroup
```

结果：

```
{
```

```
"ResponseMetadata": {  
  "RequestId": "253fbffd-6993-11e2-bc3a-47431073908a"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteClusterSubnetGroup](#)中的。

## delete-cluster

以下代码示例演示了如何使用 delete-cluster。

### AWS CLI

删除没有最终集群的集群 SnapshotThis 示例删除集群，强制删除数据，因此不会创建最终的集群快照。命令：

```
aws redshift delete-cluster --cluster-identifier mycluster --skip-final-cluster-snapshot
```

删除集群，允许使用最终集群 SnapshotThis 示例删除集群，但指定了最终集群 Snapshot.Command：

```
aws redshift delete-cluster --cluster-identifier mycluster --final-cluster-snapshot-identifier myfinalsnapshot
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCluster](#)中的。

## delete-event-subscription

以下代码示例演示了如何使用 delete-event-subscription。

### AWS CLI

删除活动订阅

以下delete-event-subscription示例删除了指定的事件通知订阅。

```
aws redshift delete-event-subscription \  
  --subscription-name mysubscription
```

此命令不生成任何输出。

有关更多信息，请参阅[亚马逊 Redshift 集群管理指南中的订阅亚马逊 Redshift 事件通知](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteEventSubscription](#)中的。

## **delete-hsm-client-certificate**

以下代码示例演示了如何使用 delete-hsm-client-certificate。

### AWS CLI

删除 HSM 客户端证书

以下delete-hsm-client-certificate示例删除 HSM 客户端证书。

```
aws redshift delete-hsm-client-certificate \  
  --hsm-client-certificate-identifier myhsmclientcert
```

此命令不生成任何输出。

有关更多信息，请参阅[亚马逊 Redshift 集群管理指南中的亚马逊 Redshift API 权限参考](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteHsmClientCertificate](#)中的。

## **delete-hsm-configuration**

以下代码示例演示了如何使用 delete-hsm-configuration。

### AWS CLI

删除 HSM 配置

以下delete-hsm-configuration示例从当前 AWS 账户中删除指定的 HSM 配置。

```
aws redshift delete-hsm-configuration \  
  --hsm-configuration-identifier myhsmconnection
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteHsmConfiguration](#)中的。



## delete-scheduled-action

以下代码示例演示了如何使用 delete-scheduled-action。

### AWS CLI

删除预定操作

以下delete-scheduled-action示例删除了指定的计划操作。

```
aws redshift delete-scheduled-action \  
  --scheduled-action-name myscheduledaction
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteScheduledAction](#)中的。

## delete-snapshot-copy-grant

以下代码示例演示了如何使用 delete-snapshot-copy-grant。

### AWS CLI

要删除快照副本，请授予

以下delete-snapshot-copy-grant示例删除了指定的快照复制授权。

```
aws redshift delete-snapshot-copy-grant \  
  --snapshot-copy-grant-name mysnapshotcopygrantname
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊 Redshift 集群管理指南](#)》中的[亚马逊 Redshift 数据库加密](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSnapshotCopyGrant](#)中的。

## delete-snapshot-schedule

以下代码示例演示了如何使用 delete-snapshot-schedule。

### AWS CLI

删除快照计划

以下delete-snapshot-schedule示例删除了指定的快照计划。在删除计划之前，必须取消群集的关联。

```
aws redshift delete-snapshot-schedule \  
  --schedule-identifier mysnapshotschedule
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Redshift 集群管理指南中的[自动快照计划](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSnapshotSchedule](#)中的。

## delete-tags

以下代码示例演示了如何使用 delete-tags。

### AWS CLI

从集群中删除标签

以下delete-tags示例从指定集群中删除具有指定密钥名称的标签。

```
aws redshift delete-tags \  
  --resource-name arn:aws:redshift:us-west-2:123456789012:cluster:mycluster \  
  --tag-keys "clustertagkey" "clustertagvalue"
```

此命令不生成任何输出。

有关更多信息，请参阅《亚马逊 Redshift [集群管理指南](#)》中的在 Amazon Redshift 中为资源添加[标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTags](#)中的。

## describe-account-attributes

以下代码示例演示了如何使用 describe-account-attributes。

### AWS CLI

描述 AWS 账户的属性

以下describe-account-attributes示例显示了与主叫 AWS 账户关联的属性。

```
aws redshift describe-account-attributes
```

输出：

```
{
  "AccountAttributes": [
    {
      "AttributeName": "max-defer-maintenance-duration",
      "AttributeValues": [
        {
          "AttributeValue": "45"
        }
      ]
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAccountAttributes](#)中的。

## describe-cluster-db-revisions

以下代码示例演示了如何使用 describe-cluster-db-revisions。

AWS CLI

描述集群的数据库版本

以下describe-cluster-db-revisions示例显示了指定群集的ClusterDbRevision对象数组的详细信息。

```
aws redshift describe-cluster-db-revisions \
  --cluster-identifier mycluster
```

输出：

```
{
  "ClusterDbRevisions": [
    {
      "ClusterIdentifier": "mycluster",
      "CurrentDatabaseRevision": "11420",
```

```
        "DatabaseRevisionReleaseDate": "2019-11-22T16:43:49.597Z",
        "RevisionTargets": []
    }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeClusterDbRevisions](#)中的。

## describe-cluster-parameter-groups

以下代码示例演示了如何使用 describe-cluster-parameter-groups。

### AWS CLI

获取所有集群参数的描述 GroupsThis 示例返回账户中所有集群参数组的描述以及列标题。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift describe-cluster-parameter-groups
```

结果：

```
{
  "ParameterGroups": [
    {
      "ParameterGroupFamily": "redshift-1.0",
      "Description": "My first cluster parameter group",
      "ParameterGroupName": "myclusterparametergroup"
    }
  ],
  "ResponseMetadata": {
    "RequestId": "8ceb8f6f-64cc-11e2-bea9-49e0ce183f07"
  }
}
```

您也可以使用 --output text 选项以文本格式获取相同的信息。命令：

--output text 选项。命令：

选项。命令：

```
aws redshift describe-cluster-parameter-groups --output text
```

结果：

```
redshift-1.0      My first cluster parameter group      myclusterparametergroup
RESPONSEMETADATA  9e665a36-64cc-11e2-8f7d-3b939af52818
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeClusterParameterGroups](#)中的。

## describe-cluster-parameters

以下代码示例演示了如何使用 describe-cluster-parameters。

### AWS CLI

检索指定集群参数的参数 GroupThis 示例检索指定参数组的参数。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift describe-cluster-parameters --parameter-group-name
myclusterparametergroup
```

结果：

```
{
  "Parameters": [
    {
      "Description": "Sets the display format for date and time values.",
      "DataType": "string",
      "IsModifiable": true,
      "Source": "engine-default",
      "ParameterValue": "ISO, MDY",
      "ParameterName": "datestyle"
    },
    {
      "Description": "Sets the number of digits displayed for floating-point
values",
      "DataType": "integer",
      "IsModifiable": true,
      "AllowedValues": "-15-2",
      "Source": "engine-default",
      "ParameterValue": "0",
      "ParameterName": "extra_float_digits"
    }
  ],
}
```

```
(...remaining output omitted...)
]
}
```

您也可以使用 `--output text` 选项以文本格式获取相同的信息。命令：

`--output text` 选项。命令：

选项。命令：

```
aws redshift describe-cluster-parameters --parameter-group-name
myclusterparametergroup --output text
```

结果：

```
RESPONSEMETADATA    cdac40aa-64cc-11e2-9e70-918437dd236d
Sets the display format for date and time values.    string True    engine-default
ISO, MDY        datestyle
Sets the number of digits displayed for floating-point values    integer True
-15-2    engine-default 0        extra_float_digits
This parameter applies a user-defined label to a group of queries that are run
during the same session..    string True    engine-default default query_group
require ssl for all databaseconnections    boolean True    true,false    engine-
default false    require_ssl
Sets the schema search order for names that are not schema-qualified.    string
True    engine-default $user, public    search_path
Aborts any statement that takes over the specified number of milliseconds.    integer
True    engine-default 0        statement_timeout
wlm json configuration    string True    engine-default
\ [{"query_concurrency":5}]    wlm_json_configuration
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeClusterParameters](#)中的。

## describe-cluster-security-groups

以下代码示例演示了如何使用 `describe-cluster-security-groups`。

### AWS CLI

获取所有集群安全的描述 `GroupsThis` 示例返回该账户的所有集群安全组的描述。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift describe-cluster-security-groups
```

结果：

```
{
  "ClusterSecurityGroups": [
    {
      "OwnerId": "100447751468",
      "Description": "default",
      "ClusterSecurityGroupName": "default",
      "EC2SecurityGroups": \[],
      "IPRanges": [
        {
          "Status": "authorized",
          "CIDRIP": "0.0.0.0/0"
        }
      ]
    },
    {
      "OwnerId": "100447751468",
      "Description": "This is my cluster security group",
      "ClusterSecurityGroupName": "mysecuritygroup",
      "EC2SecurityGroups": \[],
      "IPRanges": \[]
    },
    (...remaining output omitted...)
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeClusterSecurityGroups](#)中的。

## describe-cluster-snapshots

以下代码示例演示了如何使用 describe-cluster-snapshots。

### AWS CLI

获取所有集群的描述 SnapshotsThis 示例返回该账户的所有集群快照的描述。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift describe-cluster-snapshots
```

结果：

```
{
  "Snapshots": [
    {
      "Status": "available",
      "SnapshotCreateTime": "2013-07-17T22:02:22.852Z",
      "EstimatedSecondsToCompletion": -1,
      "AvailabilityZone": "us-east-1a",
      "ClusterVersion": "1.0",
      "MasterUsername": "adminuser",
      "Encrypted": false,
      "OwnerAccount": "111122223333",
      "BackupProgressInMegabytes": 20.0,
      "ElapsedTimeInSeconds": 0,
      "DBName": "dev",
      "CurrentBackupRateInMegabytesPerSecond": 0.0,
      "ClusterCreateTime": "2013-01-22T21:59:29.559Z",
      "ActualIncrementalBackupSizeInMegabytes"; 20.0
      "SnapshotType": "automated",
      "NodeType": "dw.hs1.xlarge",
      "ClusterIdentifier": "mycluster",
      "Port": 5439,
      "TotalBackupSizeInMegabytes": 20.0,
      "NumberOfNodes": "2",
      "SnapshotIdentifier": "cm:mycluster-2013-01-22-22-04-18"
    },
    {
      "EstimatedSecondsToCompletion": 0,
      "OwnerAccount": "111122223333",
      "CurrentBackupRateInMegabytesPerSecond": 0.1534,
      "ActualIncrementalBackupSizeInMegabytes"; 11.0,
      "NumberOfNodes": "2",
      "Status": "available",
      "ClusterVersion": "1.0",
      "MasterUsername": "adminuser",
      "AccountsWithRestoreAccess": [
        {
          "AccountID": "444455556666"
        }
      ],
      "TotalBackupSizeInMegabytes": 20.0,
      "DBName": "dev",
      "BackupProgressInMegabytes": 11.0,
      "ClusterCreateTime": "2013-01-22T21:59:29.559Z",
```



```
    "ElapsedTimeInSeconds": 0,
    "ClusterIdentifier": "mycluster",
    "SnapshotCreateTime": "2013-07-17T22:04:18.947Z",
    "AvailabilityZone": "us-east-1a",
    "NodeType": "dw.hs1.xlarge",
    "Encrypted": false,
    "SnapshotType": "manual",
    "Port": 5439,
    "SnapshotIdentifier": "my-snapshot-id"
  } ]
}
(...remaining output omitted...)
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeClusterSnapshots](#)中的。

## describe-cluster-subnet-groups

以下代码示例演示了如何使用 describe-cluster-subnet-groups。

### AWS CLI

获取所有集群子网的描述 GroupsThis 示例返回所有集群子网组的描述。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift describe-cluster-subnet-groups
```

结果：

```
{
  "ClusterSubnetGroups": [
    {
      "Subnets": [
        {
          "SubnetStatus": "Active",
          "SubnetIdentifier": "subnet-763fdd1c",
          "SubnetAvailabilityZone": {
            "Name": "us-east-1a"
          }
        }
      ],
      "VpcId": "vpc-7e3fdd14",
      "SubnetGroupStatus": "Complete",
    }
  ]
}
```

```
        "Description": "My subnet group",
        "ClusterSubnetGroupName": "mysubnetgroup"
    }
  ],
  "ResponseMetadata": {
    "RequestId": "37fa8c89-6990-11e2-8f75-ab4018764c77"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeClusterSubnetGroups](#)中的。

## describe-cluster-tracks

以下代码示例演示了如何使用 describe-cluster-tracks。

### AWS CLI

#### 描述集群轨道

以下describe-cluster-tracks示例显示了可用维护轨道的详细信息。

```
aws redshift describe-cluster-tracks \
  --maintenance-track-name current
```

输出：

```
{
  "MaintenanceTracks": [
    {
      "MaintenanceTrackName": "current",
      "DatabaseVersion": "1.0.11420",
      "UpdateTargets": [
        {
          "MaintenanceTrackName": "preview_features",
          "DatabaseVersion": "1.0.11746",
          "SupportedOperations": [
            {
              "OperationName": "restore-from-cluster-snapshot"
            }
          ]
        }
      ]
    },
    {
```

```

    "MaintenanceTrackName": "trailing",
    "DatabaseVersion": "1.0.11116",
    "SupportedOperations": [
      {
        "OperationName": "restore-from-cluster-snapshot"
      },
      {
        "OperationName": "modify-cluster"
      }
    ]
  }
]
}

```

有关更多信息，请参阅 Amazon Redshift 集群管理指南中的选择集群[维护轨道](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeClusterTracks](#)中的。

## describe-cluster-versions

以下代码示例演示了如何使用 describe-cluster-versions。

### AWS CLI

获取所有集群的描述 VersionsThis 示例返回所有集群版本的描述。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift describe-cluster-versions
```

结果：

```

{
  "ClusterVersions": [
    {
      "ClusterVersion": "1.0",
      "Description": "Initial release",
      "ClusterParameterGroupFamily": "redshift-1.0"
    } ],
  "ResponseMetadata": {
    "RequestId": "16a53de3-64cc-11e2-bec0-17624ad140dd"
  }
}

```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeClusterVersions](#)中的。

## describe-clusters

以下代码示例演示了如何使用 describe-clusters。

### AWS CLI

获取全部描述 ClustersThis 示例返回该账户所有集群的描述。默认情况下，输出采用 JSON 格式。

命令：

```
aws redshift describe-clusters
```

结果：

```
{
  "Clusters": [
    {
      "NodeType": "dw.hs1.xlarge",
      "Endpoint": {
        "Port": 5439,
        "Address": "mycluster.coqoarplqhsn.us-east-1.redshift.amazonaws.com"
      },
      "ClusterVersion": "1.0",
      "PubliclyAccessible": "true",
      "MasterUsername": "adminuser",
      "ClusterParameterGroups": [
        {
          "ParameterApplyStatus": "in-sync",
          "ParameterGroupName": "default.redshift-1.0"
        }
      ],
      "ClusterSecurityGroups": [
        {
          "Status": "active",
          "ClusterSecurityGroupName": "default"
        }
      ],
      "AllowVersionUpgrade": true,
      "VpcSecurityGroups": [],
      "AvailabilityZone": "us-east-1a",
      "ClusterCreateTime": "2013-01-22T21:59:29.559Z",
```

```

    "PreferredMaintenanceWindow": "sat:03:30-sat:04:00",
    "AutomatedSnapshotRetentionPeriod": 1,
    "ClusterStatus": "available",
    "ClusterIdentifier": "mycluster",
    "DBName": "dev",
    "NumberOfNodes": 2,
    "PendingModifiedValues": {}
  } ],
  "ResponseMetadata": {
    "RequestId": "65b71cac-64df-11e2-8f5b-e90bd6c77476"
  }
}

```

您也可以使用 `--output text` 选项以文本格式获取相同的信息。命令：

`--output text` 选项。命令：

选项。命令：

```
aws redshift describe-clusters --output text
```

结果：

```

dw.hs1.xlarge      1.0      true      adminuser      True      us-east-1a
2013-01-22T21:59:29.559Z      sat:03:30-sat:04:00      1      available
mycluster      dev      2
ENDPOINT      5439      mycluster.coqoarplqhsn.us-east-1.redshift.amazonaws.com
in-sync      default.redshift-1.0
active      default
PENDINGMODIFIEDVALUES
RESPONSEMETADATA      934281a8-64df-11e2-b07c-f7fbdd006c67

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeClusters](#)中的。

## describe-default-cluster-parameters

以下代码示例演示了如何使用 `describe-default-cluster-parameters`。

### AWS CLI

获取默认集群描述 `ParametersThis` 示例返回该 `redshift-1.0` 系列的默认集群参数的描述。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift describe-default-cluster-parameters --parameter-group-family
redshift-1.0
```

结果：

```
{
  "DefaultClusterParameters": {
    "ParameterGroupFamily": "redshift-1.0",
    "Parameters": [
      {
        "Description": "Sets the display format for date and time values.",
        "DataType": "string",
        "IsModifiable": true,
        "Source": "engine-default",
        "ParameterValue": "ISO, MDY",
        "ParameterName": "datestyle"
      },
      {
        "Description": "Sets the number of digits displayed for floating-point
values",
        "DataType": "integer",
        "IsModifiable": true,
        "AllowedValues": "-15-2",
        "Source": "engine-default",
        "ParameterValue": "0",
        "ParameterName": "extra_float_digits"
      },
      (...remaining output omitted...)
    ]
  }
}
```

要查看有效参数组系列的列表，请使用 `describe-cluster-parameter-groups` 命令。

`describe-cluster-parameter-groups` 命令。

命令。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDefaultClusterParameters](#) 中的。

## describe-event-categories

以下代码示例演示了如何使用 `describe-event-categories`。

## AWS CLI

### 描述集群的事件类别

以下describe-event-categories示例显示了集群的事件类别的详细信息。

```
aws redshift describe-event-categories \  
  --source-type cluster
```

输出：

```
{  
  "EventCategoriesMapList": [  
    {  
      "SourceType": "cluster",  
      "Events": [  
        {  
          "EventId": "REDSHIFT-EVENT-2000",  
          "EventCategories": [  
            "management"  
          ],  
          "EventDescription": "Cluster <cluster name> created at <time in  
UTC>.",  
          "Severity": "INFO"  
        },  
        {  
          "EventId": "REDSHIFT-EVENT-2001",  
          "EventCategories": [  
            "management"  
          ],  
          "EventDescription": "Cluster <cluster name> deleted at <time in  
UTC>.",  
          "Severity": "INFO"  
        },  
        {  
          "EventId": "REDSHIFT-EVENT-3625",  
          "EventCategories": [  
            "monitoring"  
          ],  
          "EventDescription": "The cluster <cluster name> can't be resumed  
with its previous elastic network interface <ENI id>. We will allocate a new  
elastic network interface and associate it with the cluster node.",  
          "Severity": "INFO"  
        }  
      ]  
    }  
  ]  
}
```

```
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEventCategories](#)中的。

## describe-event-subscriptions

以下代码示例演示了如何使用 describe-event-subscriptions。

### AWS CLI

#### 描述活动订阅

以下describe-event-subscriptions示例显示了指定订阅的事件通知订阅。

```
aws redshift describe-event-subscriptions \
  --subscription-name mysubscription
```

#### 输出：

```
{
  "EventSubscriptionsList": [
    {
      "CustomerAwsId": "123456789012",
      "CustSubscriptionId": "mysubscription",
      "SnsTopicArn": "arn:aws:sns:us-west-2:123456789012:MySNSTopic",
      "Status": "active",
      "SubscriptionCreationTime": "2019-12-09T21:50:21.332Z",
      "SourceIdsList": [],
      "EventCategoriesList": [
        "management"
      ],
      "Severity": "ERROR",
      "Enabled": true,
      "Tags": []
    }
  ]
}
```



有关更多信息，请参阅亚马逊 [Redshift 集群管理指南](#) 中的 [订阅亚马逊 Redshift 事件通知](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEventSubscriptions](#) 中的。

## describe-events

以下代码示例演示了如何使用 describe-events。

### AWS CLI

描述所有事件此示例返回所有事件。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift describe-events
```

结果：

```
{
  "Events": [
    {
      "Date": "2013-01-22T19:17:03.640Z",
      "SourceIdentifier": "myclusterparametergroup",
      "Message": "Cluster parameter group myclusterparametergroup has been
created.",
      "SourceType": "cluster-parameter-group"
    } ],
  "ResponseMetadata": {
    "RequestId": "9f056111-64c9-11e2-9390-ff04f2c1e638"
  }
}
```

您也可以使用 --output text 选项以文本格式获取相同的信息。命令：

--output text 选项。命令：

选项。命令：

```
aws redshift describe-events --output text
```

结果：

```
2013-01-22T19:17:03.640Z    myclusterparametergroup Cluster parameter group
myclusterparametergroup has been created.    cluster-parameter-group
```

```
RESPONSEMETADATA      8e5fe765-64c9-11e2-bce3-e56f52c50e17
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEvents](#)中的。

## describe-hsm-client-certificates

以下代码示例演示了如何使用 describe-hsm-client-certificates。

### AWS CLI

描述 HSM 客户端证书

以下describe-hsm-client-certificates示例显示了指定 HSM 客户端证书的详细信息。

```
aws redshift describe-hsm-client-certificates \
  --hsm-client-certificate-identifier myhsmclientcert
```

输出：

```
{
  "HsmClientCertificates": [
    {
      "HsmClientCertificateIdentifier": "myhsmclientcert",
      "HsmClientCertificatePublicKey": "-----BEGIN CERTIFICATE-----\
EXAMPLECAfICCQD6m7oRw0uX0jANBgqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAEXAMPLERAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBA5TC01BTSBDb25zEXAMPLEwEAYDVQQDEwLUZXN0Q2lsYWxhZAd
BgqhkiG9w0BCQEWEG5vb25lQGfYXpvi5jb20wHhEXAMPLEDI1MjA0EXAMPLEN
EXAMPLE0MjA0NTIxWjCBiDELMAkGA1UEBhMCMVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGEXAMPLEQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC01BTSBDb25z
b2x1MRIwEAYDVQQDEwLUZXN0Q2lsEXAMPLEdBgqhkiG9w0BCQEWEG5vb25lQGfY
Xpvi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIEXAMPLEMaK0dn+a4GmWIWJ
21uUSfwfEvySwTC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY67EXAMPLEE
EXAMPLEZnzcvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9EXAMPLE6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEEXAMPLEBHjJnyp3780D8uTs7fLvJx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rEXAMPLE=-----END CERTIFICATE-----\n",
      "Tags": []
    }
  ]
}
```

有关更多信息，请参阅[亚马逊 Redshift 集群管理指南中的亚马逊 Redshift API 权限参考](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeHsmClientCertificates](#)中的。

## describe-hsm-configurations

以下代码示例演示了如何使用 describe-hsm-configurations。

### AWS CLI

#### 描述 HSM 配置

以下describe-hsm-configurations示例显示了主叫 AWS 账户可用 HSM 配置的详细信息。

```
aws redshift describe-hsm-configurations /
  --hsm-configuration-identifier myhsmconnection
```

输出：

```
{
  "HsmConfigurations": [
    {
      "HsmConfigurationIdentifier": "myhsmconnection",
      "Description": "My HSM connection",
      "HsmIpAddress": "192.0.2.09",
      "HsmPartitionName": "myhsmpartition",
      "Tags": []
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeHsmConfigurations](#)中的。

## describe-logging-status

以下代码示例演示了如何使用 describe-logging-status。

### AWS CLI

#### 描述集群的日志状态

以下describe-logging-status示例显示是否正在记录集群的查询和连接尝试次数等信息。

```
aws redshift describe-logging-status \  
  --cluster-identifier mycluster
```

输出：

```
{  
  "LoggingEnabled": false  
}
```

有关更多信息，请参阅 Amazon Redshift 集群管理指南中的[数据库审计日志](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeLoggingStatus](#)中的。

## describe-node-configuration-options

以下代码示例演示了如何使用 describe-node-configuration-options。

AWS CLI

描述节点配置选项

以下describe-node-configuration-options示例显示了可能的节点配置的属性，例如指定集群快照的节点类型、节点数和磁盘使用情况。

```
aws redshift describe-node-configuration-options \  
  --action-type restore-cluster \  
  --snapshot-identifier rs:mycluster-2019-12-09-16-42-43
```

输出：

```
{  
  "NodeConfigurationOptionList": [  
    {  
      "NodeType": "dc2.large",  
      "NumberOfNodes": 2,  
      "EstimatedDiskUtilizationPercent": 19.61  
    },  
    {  
      "NodeType": "dc2.large",  
      "NumberOfNodes": 4,  
      "EstimatedDiskUtilizationPercent": 9.96  
    }  
  ]  
}
```

```
    },
    {
      "NodeType": "ds2.xlarge",
      "NumberOfNodes": 2,
      "EstimatedDiskUtilizationPercent": 1.53
    },
    {
      "NodeType": "ds2.xlarge",
      "NumberOfNodes": 4,
      "EstimatedDiskUtilizationPercent": 0.78
    }
  ]
}
```

有关更多信息，请参阅[亚马逊 Redshift 集群管理指南中的购买亚马逊 Red shift 预留节点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeNodeConfigurationOptions](#)中的。

## describe-orderable-cluster-options

以下代码示例演示了如何使用 describe-orderable-cluster-options。

### AWS CLI

描述所有可排序集群 OptionsThis 示例返回所有可订购集群选项的描述。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift describe-orderable-cluster-options
```

结果：

```
{
  "OrderableClusterOptions": [
    {
      "NodeType": "dw.hs1.8xlarge",
      "AvailabilityZones": [
        { "Name": "us-east-1a" },
        { "Name": "us-east-1b" },
        { "Name": "us-east-1c" } ],
      "ClusterVersion": "1.0",
      "ClusterType": "multi-node"
    },
    {
```

```

    "NodeType": "dw.hs1.xlarge",
    "AvailabilityZones": [
      { "Name": "us-east-1a" },
      { "Name": "us-east-1b" },
      { "Name": "us-east-1c" } ],
    "ClusterVersion": "1.0",
    "ClusterType": "multi-node"
  },
  {
    "NodeType": "dw.hs1.xlarge",
    "AvailabilityZones": [
      { "Name": "us-east-1a" },
      { "Name": "us-east-1b" },
      { "Name": "us-east-1c" } ],
    "ClusterVersion": "1.0",
    "ClusterType": "single-node"
  } ],
  "ResponseMetadata": {
    "RequestId": "f6000035-64cb-11e2-9135-ff82df53a51a"
  }
}

```

您也可以使用 `--output text` 选项以文本格式获取相同的信息。命令：

`--output text` 选项。命令：

选项。命令：

```
aws redshift describe-orderable-cluster-options --output text
```

结果：

```

dw.hs1.8xlarge      1.0      multi-node
us-east-1a
us-east-1b
us-east-1c
dw.hs1.xlarge      1.0      multi-node
us-east-1a
us-east-1b
us-east-1c
dw.hs1.xlarge      1.0      single-node
us-east-1a
us-east-1b

```

```
us-east-1c
RESPONSEMETADATA    e648696b-64cb-11e2-bec0-17624ad140dd
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeOrderableClusterOptions](#)中的。

## describe-reserved-node-offerings

以下代码示例演示了如何使用 `describe-reserved-node-offerings`。

### AWS CLI

描述预留节点 OfferingsThis 示例显示了所有可供购买的预留节点产品。命令：

```
aws redshift describe-reserved-node-offerings
```

结果：

```
{
  "ReservedNodeOfferings": [
    {
      "OfferingType": "Heavy Utilization",
      "FixedPrice": "",
      "NodeType": "dw.hs1.xlarge",
      "UsagePrice": "",
      "RecurringCharges": [
        {
          "RecurringChargeAmount": "",
          "RecurringChargeFrequency": "Hourly"
        }
      ],
      "Duration": 31536000,
      "ReservedNodeOfferingId": "ceb6a579-cf4c-4343-be8b-d832c45ab51c"
    },
    {
      "OfferingType": "Heavy Utilization",
      "FixedPrice": "",
      "NodeType": "dw.hs1.8xlarge",
      "UsagePrice": "",
      "RecurringCharges": [
        {
          "RecurringChargeAmount": "",
          "RecurringChargeFrequency": "Hourly"
        }
      ],
    }
  ]
}
```

```

        "Duration": 31536000,
        "ReservedNodeOfferingId": "e5a2ff3b-352d-4a9c-ad7d-373c4cab5dd2"
    },
    ...remaining output omitted...
],
"ResponseMetadata": {
    "RequestId": "8b1a1a43-75ff-11e2-9666-e142fe91ddd1"
}
}

```

如果您想购买预留节点产品，则可以使用有效的预留节点产品 `purchase-reserved-node-offering` 进行调用 `ReservedNodeOfferingId`。

`purchase-reserved-node-offering` 使用有效的 `ReservedNodeOfferingId`。

使用有效的 `ReservedNodeOfferingId`。

`ReservedNodeOfferingId`。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeReservedNodeOfferings](#) 中的。

## describe-reserved-nodes

以下代码示例演示了如何使用 `describe-reserved-nodes`。

### AWS CLI

Describe Reserved Nodes This 示例显示了已购买的预留节点产品。命令：

```
aws redshift describe-reserved-nodes
```

结果：

```

{
  "ResponseMetadata": {
    "RequestId": "bc29ce2e-7600-11e2-9949-4b361e7420b7"
  },
  "ReservedNodes": [
    {
      "OfferingType": "Heavy Utilization",
      "FixedPrice": "",

```



```

    "NodeType": "dw.hs1.xlarge",
    "ReservedNodeId": "1ba8e2e3-bc01-4d65-b35d-a4a3e931547e",
    "UsagePrice": "",
    "RecurringCharges": [
      {
        "RecurringChargeAmount": "",
        "RecurringChargeFrequency": "Hourly"
      }
    ],
    "NodeCount": 1,
    "State": "payment-pending",
    "StartTime": "2013-02-13T17:08:39.051Z",
    "Duration": 31536000,
    "ReservedNodeOfferingId": "ceb6a579-cf4c-4343-be8b-d832c45ab51c"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeReservedNodes](#)中的。

## describe-resize

以下代码示例演示了如何使用 describe-resize。

### AWS CLI

描述 ResizeThis 示例描述了集群的最新大小。请求的是类型为 3 个节点dw.hs1.8xlarge。Command:

```
aws redshift describe-resize --cluster-identifier mycluster
```

结果：

```

{
  "Status": "NONE",
  "TargetClusterType": "multi-node",
  "TargetNodeType": "dw.hs1.8xlarge",
  "ResponseMetadata": {
    "RequestId": "9f52b0b4-7733-11e2-aa9b-318b2909bd27"
  },
  "TargetNumberOfNodes": "3"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeResize](#)中的。

## describe-scheduled-actions

以下代码示例演示了如何使用 describe-scheduled-actions。

### AWS CLI

#### 描述计划操作

以下describe-scheduled-actions示例显示了所有当前计划操作的详细信息。

```
aws redshift describe-scheduled-actions
```

输出：

```
{
  "ScheduledActions": [
    {
      "ScheduledActionName": "resizecluster",
      "TargetAction": {
        "ResizeCluster": {
          "ClusterIdentifier": "mycluster",
          "NumberOfNodes": 4,
          "Classic": false
        }
      },
      "Schedule": "at(2019-12-10T00:07:00)",
      "IamRole": "arn:aws:iam::123456789012:role/myRedshiftRole",
      "State": "ACTIVE",
      "NextInvocations": [
        "2019-12-10T00:07:00Z"
      ]
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeScheduledActions](#)中的。

## describe-snapshot-copy-grants

以下代码示例演示了如何使用 describe-snapshot-copy-grants。

## AWS CLI

### 描述快照副本授权

以下describe-snapshot-copy-grants示例显示了指定集群快照复制授予的详细信息。

```
aws redshift describe-snapshot-copy-grants \  
  --snapshot-copy-grant-name mysnapshotcopygrantname
```

输出：

```
{  
  "SnapshotCopyGrants": [  
    {  
      "SnapshotCopyGrantName": "mysnapshotcopygrantname",  
      "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/  
bPxRfih3yCo8nvbEXAMPLEKEY",  
      "Tags": []  
    }  
  ]  
}
```

有关更多信息，请参阅《[亚马逊 Redshift 集群管理指南](#)》中的[亚马逊 Redshift 数据库加密](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSnapshotCopyGrants](#)中的。

## describe-snapshot-schedules

以下代码示例演示了如何使用 describe-snapshot-schedules。

### AWS CLI

#### 描述快照计划

以下describe-snapshot-schedules示例显示了指定集群快照计划的详细信息。

```
aws redshift describe-snapshot-schedules \  
  --cluster-identifier mycluster \  
  --schedule-identifier mysnapshotschedule
```

输出：

```
{
```

```
"SnapshotSchedules": [
  {
    "ScheduleDefinitions": [
      "rate(12 hours)"
    ],
    "ScheduleIdentifier": "mysnapshotschedule",
    "ScheduleDescription": "My schedule description",
    "Tags": [],
    "AssociatedClusterCount": 1,
    "AssociatedClusters": [
      {
        "ClusterIdentifier": "mycluster",
        "ScheduleAssociationState": "ACTIVE"
      }
    ]
  }
]
```

有关更多信息，请参阅 Amazon Redshift 集群管理指南中的[自动快照计划](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeSnapshotSchedules](#)中的。

## describe-storage

以下代码示例演示了如何使用 describe-storage。

### AWS CLI

#### 描述存储

以下 describe-storage 示例显示了有关该账户的备份存储空间和临时存储大小的详细信息。

```
aws redshift describe-storage
```

输出：

```
{
  "TotalBackupSizeInMegaBytes": 193149.0,
  "TotalProvisionedStorageInMegaBytes": 655360.0
}
```

有关更多信息，请参阅 Amazon Redshift 集群管理指南中的管理[快照存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStorage](#)中的。

## describe-table-restore-status

以下代码示例演示了如何使用 describe-table-restore-status。

### AWS CLI

描述来自集群快照的表还原请求的状态

以下describe-table-restore-status示例显示了为指定集群发出的表还原请求的详细信息。

```
aws redshift describe-table-restore-status /
  --cluster-identifier mycluster
```

输出：

```
{
  "TableRestoreStatusDetails": [
    {
      "TableRestoreRequestId": "z1116630-0e80-46f4-ba86-bd9670411ebd",
      "Status": "IN_PROGRESS",
      "RequestTime": "2019-12-27T18:22:12.257Z",
      "ClusterIdentifier": "mycluster",
      "SnapshotIdentifier": "mysnapshotid",
      "SourceDatabaseName": "dev",
      "SourceSchemaName": "public",
      "SourceTableName": "mytable",
      "TargetDatabaseName": "dev",
      "TargetSchemaName": "public",
      "NewTableName": "mytable-clone"
    }
  ]
}
```

有关更多信息，请参阅 Amazon Redshift 集群管理[指南中的从快照恢复表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTableRestoreStatus](#)中的。

## describe-tags

以下代码示例演示了如何使用 describe-tags。

## AWS CLI

### 描述标签

以下describe-tags示例显示了与指定标签名称和值关联的指定群集的资源。

```
aws redshift describe-tags \  
  --resource-name arn:aws:redshift:us-west-2:123456789012:cluster:mycluster \  
  --tag-keys clustertagkey \  
  --tag-values clustertagvalue
```

输出：

```
{  
  "TaggedResources": [  
    {  
      "Tag": {  
        "Key": "clustertagkey",  
        "Value": "clustertagvalue"  
      },  
      "ResourceName": "arn:aws:redshift:us-  
west-2:123456789012:cluster:mycluster",  
      "ResourceType": "cluster"  
    }  
  ]  
}
```

有关更多信息，请参阅《[亚马逊 Redshift 集群管理指南](#)》中的[在 Amazon Redshift 中为资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTags](#)中的。

## disable-snapshot-copy

以下代码示例演示了如何使用 disable-snapshot-copy。

### AWS CLI

#### 禁用集群的快照复制

以下disable-snapshot-copy示例禁用指定集群的自动快照复制。

```
aws redshift disable-snapshot-copy \  
--cluster-identifier mycluster
```

输出：

```
{  
  "Cluster": {  
    "ClusterIdentifier": "mycluster",  
    "NodeType": "dc2.large",  
    "ClusterStatus": "available",  
    "ClusterAvailabilityStatus": "Available",  
    "MasterUsername": "adminuser",  
    "DBName": "dev",  
    "Endpoint": {  
      "Address": "mycluster.cmeaswqeuae.us-west-2.redshift.amazonaws.com",  
      "Port": 5439  
    },  
    "ClusterCreateTime": "2019-12-05T18:44:36.991Z",  
    "AutomatedSnapshotRetentionPeriod": 3,  
    "ManualSnapshotRetentionPeriod": -1,  
    "ClusterSecurityGroups": [],  
    "VpcSecurityGroups": [  
      {  
        "VpcSecurityGroupId": "sh-i9b431cd",  
        "Status": "active"  
      }  
    ],  
    "ClusterParameterGroups": [  
      {  
        "ParameterGroupName": "default.redshift-1.0",  
        "ParameterApplyStatus": "in-sync"  
      }  
    ],  
    "ClusterSubnetGroupName": "default",  
    "VpcId": "vpc-b1fel7t9",  
    "AvailabilityZone": "us-west-2f",  
    "PreferredMaintenanceWindow": "sat:16:00-sat:16:30",  
    "PendingModifiedValues": {  
      "NodeType": "dc2.large",  
      "NumberOfNodes": 2,  
      "ClusterType": "multi-node"  
    },  
    "ClusterVersion": "1.0",
```

```
    "AllowVersionUpgrade": true,
    "NumberOfNodes": 4,
    "PubliclyAccessible": false,
    "Encrypted": false,
    "Tags": [
      {
        "Key": "mytags",
        "Value": "tag1"
      }
    ],
    "EnhancedVpcRouting": false,
    "IamRoles": [
      {
        "IamRoleArn": "arn:aws:iam::123456789012:role/myRedshiftRole",
        "ApplyStatus": "in-sync"
      }
    ],
    "MaintenanceTrackName": "current",
    "DeferredMaintenanceWindows": [],
    "ExpectedNextSnapshotScheduleTime": "2019-12-10T04:42:43.390Z",
    "ExpectedNextSnapshotScheduleTimeStatus": "OnTrack",
    "NextMaintenanceWindowStartTime": "2019-12-14T16:00:00Z"
  }
}
```

有关更多信息，请参阅 Amazon Redshift 集群管理指南中的[将快照复制到其他 AWS 区域](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DisableSnapshotCopy](#)中的。

## enable-snapshot-copy

以下代码示例演示了如何使用 enable-snapshot-copy。

### AWS CLI

为群集启用快照复制

以下 enable-snapshot-copy 示例为指定群集启用快照的自动复制。

```
aws redshift enable-snapshot-copy \
  --cluster-identifier mycluster \
  --destination-region us-west-1
```



输出：

```
{
  "Cluster": {
    "ClusterIdentifier": "mycluster",
    "NodeType": "dc2.large",
    "ClusterStatus": "available",
    "ClusterAvailabilityStatus": "Available",
    "MasterUsername": "adminuser",
    "DBName": "dev",
    "Endpoint": {
      "Address": "mycluster.cmeaswqeuae.us-west-2.redshift.amazonaws.com",
      "Port": 5439
    },
    "ClusterCreateTime": "2019-12-05T18:44:36.991Z",
    "AutomatedSnapshotRetentionPeriod": 3,
    "ManualSnapshotRetentionPeriod": -1,
    "ClusterSecurityGroups": [],
    "VpcSecurityGroups": [
      {
        "VpcSecurityGroupId": "sh-f4c731cd",
        "Status": "active"
      }
    ],
    "ClusterParameterGroups": [
      {
        "ParameterGroupName": "default.redshift-1.0",
        "ParameterApplyStatus": "in-sync"
      }
    ],
    "ClusterSubnetGroupName": "default",
    "VpcId": "vpc-b1ael7t9",
    "AvailabilityZone": "us-west-2f",
    "PreferredMaintenanceWindow": "sat:16:00-sat:16:30",
    "PendingModifiedValues": {
      "NodeType": "dc2.large",
      "NumberOfNodes": 2,
      "ClusterType": "multi-node"
    },
    "ClusterVersion": "1.0",
    "AllowVersionUpgrade": true,
    "NumberOfNodes": 4,
    "PubliclyAccessible": false,
    "Encrypted": false,
  }
}
```

```
    "ClusterSnapshotCopyStatus": {
      "DestinationRegion": "us-west-1",
      "RetentionPeriod": 7,
      "ManualSnapshotRetentionPeriod": -1
    },
    "Tags": [
      {
        "Key": "mytags",
        "Value": "tag1"
      }
    ],
    "EnhancedVpcRouting": false,
    "IamRoles": [
      {
        "IamRoleArn": "arn:aws:iam::123456789012:role/myRedshiftRole",
        "ApplyStatus": "in-sync"
      }
    ],
    "MaintenanceTrackName": "current",
    "DeferredMaintenanceWindows": [],
    "ExpectedNextSnapshotScheduleTime": "2019-12-10T04:42:43.390Z",
    "ExpectedNextSnapshotScheduleTimeStatus": "OnTrack",
    "NextMaintenanceWindowStartTime": "2019-12-14T16:00:00Z"
  }
}
```

有关更多信息，请参阅 Amazon Redshift 集群管理指南中的[将快照复制到其他 AWS 区域](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[EnableSnapshotCopy](#)中的。

## get-cluster-credentials

以下代码示例演示了如何使用 `get-cluster-credentials`。

### AWS CLI

获取 AWS 账户的集群凭证

以下 `get-cluster-credentials` 示例检索允许访问 Amazon Redshift 数据库的临时证书。

```
aws redshift get-cluster-credentials \  
  --db-user adminuser --db-name dev \  
  --cluster-identifier mycluster
```

输出：

```
{
  "DbUser": "IAM:adminuser",
  "DbPassword": "AMAFUyyuros/QjxPTtgzcsuQsqzIasdzJEN04aCtWDzXx109d6UmpkBtvEeqFly/
EXAMPLE==",
  "Expiration": "2019-12-10T17:25:05.770Z"
}
```

有关更多信息，请参阅亚马逊 Redshift 集群管理指南中的[使用亚马逊 Redshift CLI 或 API 生成 IAM 数据库证书](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetClusterCredentials](#)中的。

## get-reserved-node-exchange-offerings

以下代码示例演示了如何使用 get-reserved-node-exchange-offerings。

### AWS CLI

获取预留节点交换产品

以下get-reserved-node-exchange-offerings示例检索与指定DC1预留节点DC2ReservedNodeOfferings匹配的数组。

```
aws redshift get-reserved-node-exchange-offerings \
  --reserved-node-id 12345678-12ab-12a1-1a2a-12ab-12a12EXAMPLE
```

输出：

```
{
  "ReservedNodeOfferings": [
    {
      "ReservedNodeOfferingId": "12345678-12ab-12a1-1a2a-12ab-12a12EXAMPLE",
      "NodeType": "dc2.large",
      "Duration": 31536000,
      "FixedPrice": 0.0,
      "UsagePrice": 0.0,
      "CurrencyCode": "USD",
      "OfferingType": "All Upfront",
      "RecurringCharges": [
        {
```

```

        "RecurringChargeAmount": 0.0,
        "RecurringChargeFrequency": "Hourly"
    }
  ],
  "ReservedNodeOfferingType": "Regular"
}
]
}

```

有关更多信息，请参阅 Amazon Redshift 集群管理指南中的 [AWS 使用 CLI 升级预留节点](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetReservedNodeExchangeOfferings](#) 中的。

## modify-cluster-iam-roles

以下代码示例演示了如何使用 `modify-cluster-iam-roles`。

### AWS CLI

#### 修改集群的 IAM 角色

以下 `modify-cluster-iam-roles` 示例从指定集群中移除指定 AWS 的 IAM 角色。

```

aws redshift modify-cluster-iam-roles \
  --cluster-identifier mycluster \
  --remove-iam-roles arn:aws:iam::123456789012:role/myRedshiftRole

```

输出：

```

{
  "Cluster": {
    "ClusterIdentifier": "mycluster",
    "NodeType": "dc2.large",
    "ClusterStatus": "available",
    "ClusterAvailabilityStatus": "Available",
    "MasterUsername": "adminuser",
    "DBName": "dev",
    "Endpoint": {
      "Address": "mycluster.cmeaswqeuae.us-west-2.redshift.amazonaws.com",
      "Port": 5439
    },
    "ClusterCreateTime": "2019-12-05T18:44:36.991Z",
    "AutomatedSnapshotRetentionPeriod": 3,
  }
}

```

```
"ManualSnapshotRetentionPeriod": -1,
"ClusterSecurityGroups": [],
"VpcSecurityGroups": [
  {
    "VpcSecurityGroupId": "sh-f9b731sd",
    "Status": "active"
  }
],
"ClusterParameterGroups": [
  {
    "ParameterGroupName": "default.redshift-1.0",
    "ParameterApplyStatus": "in-sync"
  }
],
"ClusterSubnetGroupName": "default",
"VpcId": "vpc-b2fal7t9",
"AvailabilityZone": "us-west-2f",
"PreferredMaintenanceWindow": "sat:16:00-sat:16:30",
"PendingModifiedValues": {
  "NodeType": "dc2.large",
  "NumberOfNodes": 2,
  "ClusterType": "multi-node"
},
"ClusterVersion": "1.0",
"AllowVersionUpgrade": true,
"NumberOfNodes": 4,
"PubliclyAccessible": false,
"Encrypted": false,
"ClusterSnapshotCopyStatus": {
  "DestinationRegion": "us-west-1",
  "RetentionPeriod": 7,
  "ManualSnapshotRetentionPeriod": -1
},
"Tags": [
  {
    "Key": "mytags",
    "Value": "tag1"
  }
],
"EnhancedVpcRouting": false,
"IamRoles": [],
"MaintenanceTrackName": "current",
"DeferredMaintenanceWindows": [],
"ExpectedNextSnapshotScheduleTime": "2019-12-11T04:42:55.631Z",
```

```
    "ExpectedNextSnapshotScheduleTimeStatus": "OnTrack",
    "NextMaintenanceWindowStartTime": "2019-12-14T16:00:00Z"
  }
}
```

有关更多信息，请参阅《亚马逊 Redshift 集群管理指南》中的[为亚马逊 Redshift 使用基于身份的策略 \(IAM 策略\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyClusterIamRoles](#)中的。

## modify-cluster-maintenance

以下代码示例演示了如何使用 modify-cluster-maintenance。

### AWS CLI

#### 修改集群维护

以下modify-cluster-maintenance示例将指定集群的维护推迟 30 天。

```
aws redshift modify-cluster-maintenance \
  --cluster-identifier mycluster \
  --defer-maintenance \
  --defer-maintenance-duration 30
```

输出：

```
{
  "Cluster": {
    "ClusterIdentifier": "mycluster",
    "NodeType": "dc2.large",
    "ClusterStatus": "available",
    "ClusterAvailabilityStatus": "Available",
    "MasterUsername": "adminuser",
    "DBName": "dev",
    "Endpoint": {
      "Address": "mycluster.cmeaswquae.us-west-2.redshift.amazonaws.com",
      "Port": 5439
    },
    "ClusterCreateTime": "2019-12-05T18:44:36.991Z",
    "AutomatedSnapshotRetentionPeriod": 3,
    "ManualSnapshotRetentionPeriod": -1,
    "ClusterSecurityGroups": [],
  }
}
```

```
"VpcSecurityGroups": [
  {
    "VpcSecurityGroupId": "sh-a1a123ab",
    "Status": "active"
  }
],
"ClusterParameterGroups": [
  {
    "ParameterGroupName": "default.redshift-1.0",
    "ParameterApplyStatus": "in-sync"
  }
],
"ClusterSubnetGroupName": "default",
"VpcId": "vpc-b1ael7t9",
"AvailabilityZone": "us-west-2f",
"PreferredMaintenanceWindow": "sat:16:00-sat:16:30",
"PendingModifiedValues": {
  "NodeType": "dc2.large",
  "NumberOfNodes": 2,
  "ClusterType": "multi-node"
},
"ClusterVersion": "1.0",
"AllowVersionUpgrade": true,
"NumberOfNodes": 4,
"PubliclyAccessible": false,
"Encrypted": false,
"ClusterSnapshotCopyStatus": {
  "DestinationRegion": "us-west-1",
  "RetentionPeriod": 7,
  "ManualSnapshotRetentionPeriod": -1
},
"Tags": [
  {
    "Key": "mytags",
    "Value": "tag1"
  }
],
"EnhancedVpcRouting": false,
"IamRoles": [],
"MaintenanceTrackName": "current",
"DeferredMaintenanceWindows": [
  {
    "DeferMaintenanceIdentifier": "dfm-mUdVIFcT1B4SGhw6fyF",
    "DeferMaintenanceStartTime": "2019-12-10T18:18:39.354Z",
```

```

        "DeferMaintenanceEndTime": "2020-01-09T18:18:39.354Z"
      }
    ],
    "ExpectedNextSnapshotScheduleTime": "2019-12-11T04:42:55.631Z",
    "ExpectedNextSnapshotScheduleTimeStatus": "OnTrack",
    "NextMaintenanceWindowStartTime": "2020-01-11T16:00:00Z"
  }
}

```

有关更多信息，请参阅 Amazon Redshift 集群管理指南中的 [集群维护](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyClusterMaintenance](#) 中的。

## modify-cluster-parameter-group

以下代码示例演示了如何使用 modify-cluster-parameter-group。

### AWS CLI

#### 修改参数组中的参数

以下 modify-cluster-parameter-group 示例修改了工作负载管理的 wlm\_json\_configuration 参数。它接受来自包含下面所示的 JSON 内容的文件中的参数。

```

aws redshift modify-cluster-parameter-group \
  --parameter-group-name myclusterparametergroup \
  --parameters file://modify_pg.json

```

modify\_pg.json 的内容：

```

[
  {
    "ParameterName": "wlm_json_configuration",
    "ParameterValue": "[{\"user_group\": \"example_user_group1\", \"query_group\": \"example_query_group1\", \"query_concurrency\": 7}, {\"query_concurrency\": 5}]"
  }
]

```

输出：

```

{

```



```
"ParameterGroupStatus": "Your parameter group has been updated but changes won't
get applied until you reboot the associated Clusters.",
"ParameterGroupName": "myclusterparametergroup",
"ResponseMetadata": {
  "RequestId": "09974cc0-64cd-11e2-bea9-49e0ce183f07"
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyClusterParameterGroup](#)中的。

## modify-cluster-snapshot-schedule

以下代码示例演示了如何使用 `modify-cluster-snapshot-schedule`。

### AWS CLI

#### 修改集群快照时间表

以下`modify-cluster-snapshot-schedule`示例将指定的快照时间表从指定集群中移除。

```
aws redshift modify-cluster-snapshot-schedule \
  --cluster-identifier mycluster \
  --schedule-identifier mysnapshotschedule \
  --disassociate-schedule
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Redshift 集群管理指南中的[自动快照计划](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyClusterSnapshotSchedule](#)中的。

## modify-cluster-snapshot

以下代码示例演示了如何使用 `modify-cluster-snapshot`。

### AWS CLI

#### 修改集群快照

以下`modify-cluster-snapshot`示例将指定集群快照的手动保留期设置为 10 天。

```
aws redshift modify-cluster-snapshot \
```

```
--snapshot-identifier mycluster-2019-11-06-16-32 \  
--manual-snapshot-retention-period 10
```

输出：

```
{  
  "Snapshot": {  
    "SnapshotIdentifier": "mycluster-2019-11-06-16-32",  
    "ClusterIdentifier": "mycluster",  
    "SnapshotCreateTime": "2019-12-07T00:34:05.633Z",  
    "Status": "available",  
    "Port": 5439,  
    "AvailabilityZone": "us-west-2f",  
    "ClusterCreateTime": "2019-12-05T18:44:36.991Z",  
    "MasterUsername": "adminuser",  
    "ClusterVersion": "1.0",  
    "SnapshotType": "manual",  
    "NodeType": "dc2.large",  
    "NumberOfNodes": 2,  
    "DBName": "dev",  
    "VpcId": "vpc-b1cel7t9",  
    "Encrypted": false,  
    "EncryptedWithHSM": false,  
    "OwnerAccount": "123456789012",  
    "TotalBackupSizeInMegaBytes": 64384.0,  
    "ActualIncrementalBackupSizeInMegaBytes": 24.0,  
    "BackupProgressInMegaBytes": 24.0,  
    "CurrentBackupRateInMegaBytesPerSecond": 13.0011,  
    "EstimatedSecondsToCompletion": 0,  
    "ElapsedTimeInSeconds": 1,  
    "Tags": [  
      {  
        "Key": "mytagkey",  
        "Value": "mytagvalue"  
      }  
    ],  
    "EnhancedVpcRouting": false,  
    "MaintenanceTrackName": "current",  
    "ManualSnapshotRetentionPeriod": 10,  
    "ManualSnapshotRemainingDays": 6,  
    "SnapshotRetentionStartTime": "2019-12-07T00:34:07.479Z"  
  }  
}
```

有关更多信息，请参阅《[亚马逊 Redshift 集群管理指南](#)》中的[亚马逊 Redshift 快照](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyClusterSnapshot](#) 中的。

## modify-cluster-subnet-group

以下代码示例演示了如何使用 `modify-cluster-subnet-group`。

### AWS CLI

修改集群子网中的子网 GroupThis 示例显示了如何修改缓存子网组中的子网列表。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift modify-cluster-subnet-group --cluster-subnet-group-name mysubnetgroup
--subnet-ids subnet-763fdd1 subnet-ac830e9
```

结果：

```
{
  "ClusterSubnetGroup":
  {
    "Subnets": [
      {
        "SubnetStatus": "Active",
        "SubnetIdentifier": "subnet-763fdd1c",
        "SubnetAvailabilityZone":
          { "Name": "us-east-1a" }
      },
      {
        "SubnetStatus": "Active",
        "SubnetIdentifier": "subnet-ac830e9",
        "SubnetAvailabilityZone":
          { "Name": "us-east-1b" }
      }
    ],
    "VpcId": "vpc-7e3fdd14",
    "SubnetGroupStatus": "Complete",
    "Description": "My subnet group",
    "ClusterSubnetGroupName": "mysubnetgroup"
  },
  "ResponseMetadata": {
    "RequestId": "8da93e89-8372-f936-93a8-873918938197a"
  }
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyClusterSubnetGroup](#)中的。

## modify-cluster

以下代码示例演示了如何使用 `modify-cluster`。

### AWS CLI

将安全组与关联 ClusterThis 示例说明如何将群集安全组与指定的集群相关联。命令：

```
aws redshift modify-cluster --cluster-identifier mycluster --cluster-security-groups mysecuritygroup
```

修改维护时段 ClusterThis 显示了如何将群集的每周首选维护时段更改为最少四小时的时段，从周日晚上 11:15 开始，到周一凌晨 3:15 结束。命令：

```
aws redshift modify-cluster --cluster-identifier mycluster --preferred-maintenance-window Sun:23:15-Mon:03:15
```

更改主密码 ClusterThis 示例显示了如何更改集群的主密码。命令：

```
aws redshift modify-cluster --cluster-identifier mycluster --master-user-password A1b2c3d4
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyCluster](#)中的。

## modify-event-subscription

以下代码示例演示了如何使用 `modify-event-subscription`。

### AWS CLI

修改活动订阅

以下`modify-event-subscription`示例禁用了指定的事件通知订阅。

```
aws redshift modify-event-subscription \  
  --subscription-name mysubscription \  
  --no-enabled
```

输出：

```
{
  "EventSubscription": {
    "CustomerAwsId": "123456789012",
    "CustSubscriptionId": "mysubscription",
    "SnsTopicArn": "arn:aws:sns:us-west-2:123456789012:MySNSStopic",
    "Status": "active",
    "SubscriptionCreationTime": "2019-12-09T21:50:21.332Z",
    "SourceIdsList": [],
    "EventCategoriesList": [
      "management"
    ],
    "Severity": "ERROR",
    "Enabled": false,
    "Tags": []
  }
}
```

有关更多信息，请参阅亚马逊 [Redshift 集群管理指南中的订阅亚马逊 Redshift 事件通知](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyEventSubscription](#)中的。

## modify-scheduled-action

以下代码示例演示了如何使用 modify-scheduled-action。

### AWS CLI

#### 修改计划操作

以下modify-scheduled-action示例为指定的现有计划操作添加了描述。

```
aws redshift modify-scheduled-action \
  --scheduled-action-name myscheduledaction \
  --scheduled-action-description "My scheduled action"
```

输出：

```
{
  "ScheduledActionName": "myscheduledaction",
  "TargetAction": {
    "ResizeCluster": {
```

```
        "ClusterIdentifier": "mycluster",
        "NumberOfNodes": 2,
        "Classic": false
    }
},
"Schedule": "at(2019-12-25T00:00:00)",
"IamRole": "arn:aws:iam::123456789012:role/myRedshiftRole",
"ScheduledActionDescription": "My scheduled action",
"State": "ACTIVE",
"NextInvocations": [
    "2019-12-25T00:00:00Z"
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyScheduledAction](#)中的。

## modify-snapshot-copy-retention-period

以下代码示例演示了如何使用 modify-snapshot-copy-retention-period。

### AWS CLI

#### 修改快照副本保留期

以下modify-snapshot-copy-retention-period示例修改了从源 AWS 区域复制指定集群的快照后在目标 AWS 区域中保留快照的天数。

```
aws redshift modify-snapshot-copy-retention-period \
  --cluster-identifier mycluster \
  --retention-period 15
```

输出：

```
{
  "Cluster": {
    "ClusterIdentifier": "mycluster",
    "NodeType": "dc2.large",
    "ClusterStatus": "available",
    "ClusterAvailabilityStatus": "Available",
    "MasterUsername": "adminuser",
    "DBName": "dev",
    "Endpoint": {
```

```
    "Address": "mycluster.cmeaswqeuae.us-west-2.redshift.amazonaws.com",
    "Port": 5439
  },
  "ClusterCreateTime": "2019-12-05T18:44:36.991Z",
  "AutomatedSnapshotRetentionPeriod": 3,
  "ManualSnapshotRetentionPeriod": -1,
  "ClusterSecurityGroups": [],
  "VpcSecurityGroups": [
    {
      "VpcSecurityGroupId": "sh-a1a123ab",
      "Status": "active"
    }
  ],
  "ClusterParameterGroups": [
    {
      "ParameterGroupName": "default.redshift-1.0",
      "ParameterApplyStatus": "in-sync"
    }
  ],
  "ClusterSubnetGroupName": "default",
  "VpcId": "vpc-b1fet7t9",
  "AvailabilityZone": "us-west-2f",
  "PreferredMaintenanceWindow": "sat:16:00-sat:16:30",
  "PendingModifiedValues": {
    "NodeType": "dc2.large",
    "NumberOfNodes": 2,
    "ClusterType": "multi-node"
  },
  "ClusterVersion": "1.0",
  "AllowVersionUpgrade": true,
  "NumberOfNodes": 4,
  "PubliclyAccessible": false,
  "Encrypted": false,
  "ClusterSnapshotCopyStatus": {
    "DestinationRegion": "us-west-1",
    "RetentionPeriod": 15,
    "ManualSnapshotRetentionPeriod": -1
  },
  "Tags": [
    {
      "Key": "mytags",
      "Value": "tag1"
    }
  ],
],
```

```

    "EnhancedVpcRouting": false,
    "IamRoles": [],
    "MaintenanceTrackName": "current",
    "DeferredMaintenanceWindows": [
      {
        "DeferMaintenanceIdentifier": "dfm-mUdVSfDcT1F4SGhw6fyF",
        "DeferMaintenanceStartTime": "2019-12-10T18:18:39.354Z",
        "DeferMaintenanceEndTime": "2020-01-09T18:18:39.354Z"
      }
    ],
    "NextMaintenanceWindowStartTime": "2020-01-11T16:00:00Z"
  }
}

```

有关更多信息，请参阅 Amazon Redshift 集群管理指南中的[快照计划格式](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ModifySnapshotCopyRetentionPeriod](#)中的。

## modify-snapshot-schedule

以下代码示例演示了如何使用 modify-snapshot-schedule。

### AWS CLI

#### 修改快照时间表

以下 modify-snapshot-schedule 示例将指定快照计划的速率修改为每 10 小时一次。

```

aws redshift modify-snapshot-schedule \
  --schedule-identifier mysnapshotschedule \
  --schedule-definitions "rate(10 hours)"

```

输出：

```

{
  "ScheduleDefinitions": [
    "rate(10 hours)"
  ],
  "ScheduleIdentifier": "mysnapshotschedule",
  "ScheduleDescription": "My schedule description",
  "Tags": []
}

```



有关更多信息，请参阅 Amazon Redshift 集群管理指南中的[快照计划格式](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ModifySnapshotSchedule](#)中的。

## **purchase-reserved-node-offering**

以下代码示例演示了如何使用 `purchase-reserved-node-offering`。

### AWS CLI

购买预留节点 NodeThis 示例显示了如何购买预留节点产品。可通过调用 `describe-reserved-node-offerings` `reserved-node-offering-id` Command 获得：

```
aws redshift purchase-reserved-node-offering --reserved-node-offering-id ceb6a579-cf4c-4343-be8b-d832c45ab51c
```

结果：

```
{
  "ReservedNode": {
    "OfferingType": "Heavy Utilization",
    "FixedPrice": "",
    "NodeType": "dw.hs1.xlarge",
    "ReservedNodeId": "1ba8e2e3-bc01-4d65-b35d-a4a3e931547e",
    "UsagePrice": "",
    "RecurringCharges": [
      {
        "RecurringChargeAmount": "",
        "RecurringChargeFrequency": "Hourly"
      }
    ],
    "NodeCount": 1,
    "State": "payment-pending",
    "StartTime": "2013-02-13T17:08:39.051Z",
    "Duration": 31536000,
    "ReservedNodeOfferingId": "ceb6a579-cf4c-4343-be8b-d832c45ab51c"
  },
  "ResponseMetadata": {
    "RequestId": "01bda7bf-7600-11e2-b605-2568d7396e7f"
  }
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[PurchaseReservedNodeOffering](#)中的。

## reboot-cluster

以下代码示例演示了如何使用 `reboot-cluster`。

### AWS CLI

重启 ClusterThis 示例重启集群。默认情况下，输出采用 JSON 格式。命令：

```
aws redshift reboot-cluster --cluster-identifier mycluster
```

结果：

```
{
  "Cluster": {
    "NodeType": "dw.hs1.xlarge",
    "Endpoint": {
      "Port": 5439,
      "Address": "mycluster.coqoarplqhsn.us-east-1.redshift.amazonaws.com"
    },
    "ClusterVersion": "1.0",
    "PubliclyAccessible": "true",
    "MasterUsername": "adminuser",
    "ClusterParameterGroups": [
      {
        "ParameterApplyStatus": "in-sync",
        "ParameterGroupName": "default.redshift-1.0"
      }
    ],
    "ClusterSecurityGroups": [
      {
        "Status": "active",
        "ClusterSecurityGroupName": "default"
      }
    ],
    "AllowVersionUpgrade": true,
    "VpcSecurityGroups": [],
    "AvailabilityZone": "us-east-1a",
    "ClusterCreateTime": "2013-01-22T21:59:29.559Z",
    "PreferredMaintenanceWindow": "sun:23:15-mon:03:15",
    "AutomatedSnapshotRetentionPeriod": 1,
    "ClusterStatus": "rebooting",
    "ClusterIdentifier": "mycluster",
    "DBName": "dev",
```

```
    "NumberOfNodes": 2,
    "PendingModifiedValues": {}
  },
  "ResponseMetadata": {
    "RequestId": "61c8b564-64e8-11e2-8f7d-3b939af52818"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RebootCluster](#)中的。

## reset-cluster-parameter-group

以下代码示例演示了如何使用 `reset-cluster-parameter-group`。

### AWS CLI

参数中的重置参数 GroupThis 示例说明了如何重置参数组中的所有参数。Command :

```
aws redshift reset-cluster-parameter-group --parameter-group-name
myclusterparametergroup --reset-all-parameters
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ResetClusterParameterGroup](#)中的。

## resize-cluster

以下代码示例演示了如何使用 `resize-cluster`。

### AWS CLI

调整集群大小

以下`resize-cluster`示例调整了指定集群的大小。

```
aws redshift resize-cluster \
  --cluster-identifier mycluster \
  --cluster-type multi-node \
  --node-type dc2.large \
  --number-of-nodes 6 \
  --classic
```

输出：

```
{
  "Cluster": {
    "ClusterIdentifier": "mycluster",
    "NodeType": "dc2.large",
    "ClusterStatus": "resizing",
    "ClusterAvailabilityStatus": "Modifying",
    "MasterUsername": "adminuser",
    "DBName": "dev",
    "Endpoint": {
      "Address": "mycluster.cmeaswqeuae.us-west-2.redshift.amazonaws.com",
      "Port": 5439
    },
    "ClusterCreateTime": "2019-12-05T18:44:36.991Z",
    "AutomatedSnapshotRetentionPeriod": 3,
    "ManualSnapshotRetentionPeriod": -1,
    "ClusterSecurityGroups": [],
    "VpcSecurityGroups": [
      {
        "VpcSecurityGroupId": "sh-a1a123ab",
        "Status": "active"
      }
    ],
    "ClusterParameterGroups": [
      {
        "ParameterGroupName": "default.redshift-1.0",
        "ParameterApplyStatus": "in-sync"
      }
    ],
    "ClusterSubnetGroupName": "default",
    "VpcId": "vpc-a1abc1a1",
    "AvailabilityZone": "us-west-2f",
    "PreferredMaintenanceWindow": "sat:16:00-sat:16:30",
    "PendingModifiedValues": {
      "NodeType": "dc2.large",
      "NumberOfNodes": 6,
      "ClusterType": "multi-node"
    },
    "ClusterVersion": "1.0",
    "AllowVersionUpgrade": true,
    "NumberOfNodes": 4,
    "PubliclyAccessible": false,
    "Encrypted": false,
    "ClusterSnapshotCopyStatus": {
```

```

        "DestinationRegion": "us-west-1",
        "RetentionPeriod": 15,
        "ManualSnapshotRetentionPeriod": -1
    },
    "Tags": [
        {
            "Key": "mytags",
            "Value": "tag1"
        }
    ],
    "EnhancedVpcRouting": false,
    "IamRoles": [],
    "MaintenanceTrackName": "current",
    "DeferredMaintenanceWindows": [
        {
            "DeferMaintenanceIdentifier": "dfm-mUdVCfDcT1B4SGhw6fyF",
            "DeferMaintenanceStartTime": "2019-12-10T18:18:39.354Z",
            "DeferMaintenanceEndTime": "2020-01-09T18:18:39.354Z"
        }
    ],
    "NextMaintenanceWindowStartTime": "2020-01-11T16:00:00Z",
    "ResizeInfo": {
        "ResizeType": "ClassicResize",
        "AllowCancelResize": true
    }
}
}

```

有关更多信息，请参阅 Amazon Red [shift 集群管理指南中的调整集群大小](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ResizeCluster](#) 中的。

## restore-from-cluster-snapshot

以下代码示例演示了如何使用 `restore-from-cluster-snapshot`。

### AWS CLI

从 SnapshotThis 示例中恢复集群从快照恢复集群。命令：

```
aws redshift restore-from-cluster-snapshot --cluster-identifier mycluster-clone --
snapshot-identifier my-snapshot-id
```

结果：

```
{
  "Cluster": {
    "NodeType": "dw.hs1.xlarge",
    "ClusterVersion": "1.0",
    "PubliclyAccessible": "true",
    "MasterUsername": "adminuser",
    "ClusterParameterGroups": [
      {
        "ParameterApplyStatus": "in-sync",
        "ParameterGroupName": "default.redshift-1.0"
      }
    ],
    "ClusterSecurityGroups": [
      {
        "Status": "active",
        "ClusterSecurityGroupName": "default"
      }
    ],
    "AllowVersionUpgrade": true,
    "VpcSecurityGroups": \[],
    "PreferredMaintenanceWindow": "sun:23:15-mon:03:15",
    "AutomatedSnapshotRetentionPeriod": 1,
    "ClusterStatus": "creating",
    "ClusterIdentifier": "mycluster-clone",
    "DBName": "dev",
    "NumberOfNodes": 2,
    "PendingModifiedValues": {}
  },
  "ResponseMetadata": {
    "RequestId": "77fd512b-64e3-11e2-8f5b-e90bd6c77476"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RestoreFromClusterSnapshot](#)中的。

## restore-table-from-cluster-snapshot

以下代码示例演示了如何使用 `restore-table-from-cluster-snapshot`。

## AWS CLI

### 从集群快照还原表

以下 `restore-table-from-cluster-snapshot` 示例根据指定集群快照中的指定表创建一个新表。

```
aws redshift restore-table-from-cluster-snapshot /
  --cluster-identifier mycluster /
  --snapshot-identifier mycluster-2019-11-19-16-17 /
  --source-database-name dev /
  --source-schema-name public /
  --source-table-name mytable /
  --target-database-name dev /
  --target-schema-name public /
  --new-table-name mytable-clone
```

输出：

```
{
  "TableRestoreStatus": {
    "TableRestoreRequestId": "a123a12b-abc1-1a1a-a123-a1234ab12345",
    "Status": "PENDING",
    "RequestTime": "2019-12-20T00:20:16.402Z",
    "ClusterIdentifier": "mycluster",
    "SnapshotIdentifier": "mycluster-2019-11-19-16-17",
    "SourceDatabaseName": "dev",
    "SourceSchemaName": "public",
    "SourceTableName": "mytable",
    "TargetDatabaseName": "dev",
    "TargetSchemaName": "public",
    "NewTableName": "mytable-clone"
  }
}
```

有关更多信息，请参阅 Amazon Redshift [集群管理指南中的从快照恢复表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RestoreTableFromClusterSnapshot](#) 中的。

### **revoke-cluster-security-group-ingress**

以下代码示例演示了如何使用 `revoke-cluster-security-group-ingress`。

## AWS CLI

撤消 EC2 安全 GroupThis 示例的访问权限撤消对指定的 Amazon EC2 安全组的访问权限。命令：

```
aws redshift revoke-cluster-security-group-ingress --cluster-security-group-name
mysecuritygroup --ec2-security-group-name myec2securitygroup --ec2-security-group-
owner-id 123445677890
```

撤消对 CIDR 范围的访问权限此示例撤消对 CIDR 范围的访问权限。命令：

```
aws redshift revoke-cluster-security-group-ingress --cluster-security-group-name
mysecuritygroup --cidrip 192.168.100.100/32
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RevokeClusterSecurityGroupIngress](#)中的。

## revoke-snapshot-access

以下代码示例演示了如何使用 revoke-snapshot-access。

## AWS CLI

撤销 AWS 账户还原授权 SnapshotThis 示例撤销该 AWS 账户444455556666恢复快照的授权。my-snapshot-id默认情况下，输出采用 JSON 格式。命令：

```
aws redshift revoke-snapshot-access --snapshot-id my-snapshot-id --account-with-
restore-access 444455556666
```

结果：

```
{
  "Snapshot": {
    "Status": "available",
    "SnapshotCreateTime": "2013-07-17T22:04:18.947Z",
    "EstimatedSecondsToCompletion": 0,
    "AvailabilityZone": "us-east-1a",
    "ClusterVersion": "1.0",
    "MasterUsername": "adminuser",
    "Encrypted": false,
    "OwnerAccount": "111122223333",
    "BackupProgressInMegabytes": 11.0,
    "ElapsedTimeInSeconds": 0,
    "DBName": "dev",
```



```

    "CurrentBackupRateInMegabytesPerSecond": 0.1534,
    "ClusterCreateTime": "2013-01-22T21:59:29.559Z",
    "ActualIncrementalBackupSizeInMegabytes"; 11.0,
    "SnapshotType": "manual",
    "NodeType": "dw.hs1.xlarge",
    "ClusterIdentifier": "mycluster",
    "TotalBackupSizeInMegabytes": 20.0,
    "Port": 5439,
    "NumberOfNodes": 2,
    "SnapshotIdentifier": "my-snapshot-id"
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RevokeSnapshotAccess](#)中的。

## rotate-encryption-key

以下代码示例演示了如何使用 rotate-encryption-key。

### AWS CLI

轮换集群的加密密钥

以下rotate-encryption-key示例轮换指定集群的加密密钥。

```

aws redshift rotate-encryption-key \
  --cluster-identifier mycluster

```

输出：

```

{
  "Cluster": {
    "ClusterIdentifier": "mycluster",
    "NodeType": "dc2.large",
    "ClusterStatus": "rotating-keys",
    "ClusterAvailabilityStatus": "Modifying",
    "MasterUsername": "adminuser",
    "DBName": "dev",
    "Endpoint": {
      "Address": "mycluster.cmeaswqeuae.us-west-2.redshift.amazonaws.com",
      "Port": 5439
    },
    "ClusterCreateTime": "2019-12-10T19:25:45.886Z",

```

```

    "AutomatedSnapshotRetentionPeriod": 30,
    "ManualSnapshotRetentionPeriod": -1,
    "ClusterSecurityGroups": [],
    "VpcSecurityGroups": [
      {
        "VpcSecurityGroupId": "sh-a1a123ab",
        "Status": "active"
      }
    ],
    "ClusterParameterGroups": [
      {
        "ParameterGroupName": "default.redshift-1.0",
        "ParameterApplyStatus": "in-sync"
      }
    ],
    "ClusterSubnetGroupName": "default",
    "VpcId": "vpc-a1abc1a1",
    "AvailabilityZone": "us-west-2a",
    "PreferredMaintenanceWindow": "sat:16:00-sat:16:30",
    "PendingModifiedValues": {},
    "ClusterVersion": "1.0",
    "AllowVersionUpgrade": true,
    "NumberOfNodes": 2,
    "PubliclyAccessible": false,
    "Encrypted": true,
    "Tags": [],
    "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/
bPxRfih3yCo8nvbEXAMPLEKEY",
    "EnhancedVpcRouting": false,
    "IamRoles": [
      {
        "IamRoleArn": "arn:aws:iam::123456789012:role/myRedshiftRole",
        "ApplyStatus": "in-sync"
      }
    ],
    "MaintenanceTrackName": "current",
    "DeferredMaintenanceWindows": [],
    "NextMaintenanceWindowStartTime": "2019-12-14T16:00:00Z"
  }
}

```

有关更多信息，请参阅 [《亚马逊 Redshift 集群管理指南》](#) 中的 [亚马逊 Redshift 数据库加密](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RotateEncryptionKey](#) 中的。

## 使用 Amazon Rekognition 的示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon Rekognition 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **compare-faces**

以下代码示例演示了如何使用 compare-faces。

有关更多信息，请参阅[比较图像中的人脸](#)。

### AWS CLI

#### 比较两张图像中的人脸

以下 compare-faces 命令将比较存储在 Amazon S3 存储桶中的两张图像中的人脸。

```
aws rekognition compare-faces \  
  --source-image '{"S3Object":{"Bucket":"MyImageS3Bucket","Name":"source.jpg"}}' \  
  --target-image '{"S3Object":{"Bucket":"MyImageS3Bucket","Name":"target.jpg"}}'
```

#### 输出：

```
{  
  "UnmatchedFaces": [],  
  "FaceMatches": [  
    {  
      "Face": {  
        "BoundingBox": {
```

```
        "Width": 0.12368916720151901,
        "Top": 0.16007372736930847,
        "Left": 0.5901257991790771,
        "Height": 0.25140416622161865
    },
    "Confidence": 100.0,
    "Pose": {
        "Yaw": -3.7351467609405518,
        "Roll": -0.10309021919965744,
        "Pitch": 0.8637830018997192
    },
    "Quality": {
        "Sharpness": 95.51618957519531,
        "Brightness": 65.29893493652344
    },
    "Landmarks": [
        {
            "Y": 0.26721030473709106,
            "X": 0.6204193830490112,
            "Type": "eyeLeft"
        },
        {
            "Y": 0.26831310987472534,
            "X": 0.6776827573776245,
            "Type": "eyeRight"
        },
        {
            "Y": 0.3514654338359833,
            "X": 0.6241428852081299,
            "Type": "mouthLeft"
        },
        {
            "Y": 0.35258132219314575,
            "X": 0.6713621020317078,
            "Type": "mouthRight"
        },
        {
            "Y": 0.3140771687030792,
            "X": 0.6428444981575012,
            "Type": "nose"
        }
    ]
},
"Similarity": 100.0
```

```
    }
  ],
  "SourceImageFace": {
    "BoundingBox": {
      "Width": 0.12368916720151901,
      "Top": 0.16007372736930847,
      "Left": 0.5901257991790771,
      "Height": 0.25140416622161865
    },
    "Confidence": 100.0
  }
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[比较图像中的人脸](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CompareFaces](#)中的。

## create-collection

以下代码示例演示了如何使用 create-collection。

有关更多信息，请参阅[创建集合](#)。

### AWS CLI

#### 创建集合

以下 create-collection 命令创建具有指定名称的集合。

```
aws rekognition create-collection \
  --collection-id "MyCollection"
```

输出：

```
{
  "CollectionArn": "aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
  "FaceModelVersion": "4.0",
  "StatusCode": 200
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[创建集合](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCollection](#)中的。

## create-stream-processor

以下代码示例演示了如何使用 create-stream-processor。

### AWS CLI

#### 创建新的流处理器

以下create-stream-processor示例使用指定配置创建新的流处理器。

```
aws rekognition create-stream-processor --name my-stream-processor\  
  --input '{"KinesisVideoStream":{"Arn":"arn:aws:kinesisvideo:us-  
west-2:123456789012:stream/macwebcam/1530559711205"}}'\  
  --stream-processor-output '{"KinesisDataStream":{"Arn":"arn:aws:kinesis:us-  
west-2:123456789012:stream/AmazonRekognitionRekStream"}}'\  
  --role-arn arn:aws:iam::123456789012:role/AmazonRekognitionDetect\  
  --settings '{"FaceSearch":  
{ "CollectionId": "MyCollection", "FaceMatchThreshold": 85.5 } }'
```

输出：

```
{  
  "StreamProcessorArn": "arn:aws:rekognition:us-  
west-2:123456789012:streamprocessor/my-stream-processor"  
}
```

有关更多信息，[请参阅亚马逊 Rekognition 开发者指南中的使用流媒体视频](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateStreamProcessor](#)中的。

## delete-collection

以下代码示例演示了如何使用 delete-collection。

有关更多信息，请参阅[删除集合](#)。

### AWS CLI

#### 删除集合

以下 `delete-collection` 命令将删除指定的集合。

```
aws rekognition delete-collection \  
  --collection-id MyCollection
```

输出：

```
{  
  "StatusCode": 200  
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[删除集合](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCollection](#)中的。

## delete-faces

以下代码示例演示了如何使用 `delete-faces`。

有关更多信息，请参阅[从集中删除人脸](#)。

### AWS CLI

从集合中删除人脸

以下 `delete-faces` 命令将从集合中删除指定的人脸。

```
aws rekognition delete-faces \  
  --collection-id MyCollection  
  --face-ids '["0040279c-0178-436e-b70a-e61b074e96b0"]'
```

输出：

```
{  
  "DeletedFaces": [  
    "0040279c-0178-436e-b70a-e61b074e96b0"  
  ]  
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[从集中删除人脸](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFaces](#)中的。

## delete-stream-processor

以下代码示例演示了如何使用 delete-stream-processor。

### AWS CLI

#### 删除流处理器

以下delete-stream-processor命令删除指定的流处理器。

```
aws rekognition delete-stream-processor \  
  --name my-stream-processor
```

此命令不生成任何输出。

有关更多信息，[请参阅亚马逊 Rekognition 开发者指南中的使用流媒体视频](#)。

- 有关 API 的详细信息，[请参阅AWS CLI 命令参考DeleteStreamProcessor](#)中的。

## describe-collection

以下代码示例演示了如何使用 describe-collection。

有关更多信息，[请参阅描述集合](#)。

### AWS CLI

#### 描述集合

以下 describe-collection 示例显示有关指定集合的详细信息。

```
aws rekognition describe-collection \  
  --collection-id MyCollection
```

输出：

```
{  
  "FaceCount": 200,  
  "CreationTimestamp": 1569444828.274,  
  "CollectionARN": "arn:aws:rekognition:us-west-2:123456789012:collection/  
MyCollection",  
  "FaceModelVersion": "4.0"  
}
```



有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[描述集合](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCollection](#)中的。

## describe-stream-processor

以下代码示例演示了如何使用 describe-stream-processor。

### AWS CLI

获取有关流处理器的信息

以下describe-stream-processor命令显示有关指定流处理器的详细信息。

```
aws rekognition describe-stream-processor \  
  --name my-stream-processor
```

输出：

```
{  
  "Status": "STOPPED",  
  "Name": "my-stream-processor",  
  "LastUpdateTimestamp": 1532449292.712,  
  "Settings": {  
    "FaceSearch": {  
      "FaceMatchThreshold": 80.0,  
      "CollectionId": "my-collection"  
    }  
  },  
  "RoleArn": "arn:aws:iam::123456789012:role/AmazonRekognitionDetectStream",  
  "StreamProcessorArn": "arn:aws:rekognition:us-west-2:123456789012:streamprocessor/my-stream-processpr",  
  "Output": {  
    "KinesisDataStream": {  
      "Arn": "arn:aws:kinesis:us-west-2:123456789012:stream/  
AmazonRekognitionRekStream"  
    }  
  },  
  "Input": {  
    "KinesisVideoStream": {  
      "Arn": "arn:aws:kinesisvideo:us-west-2:123456789012:stream/  
macwebcam/123456789012"  
    }  
  }  
}
```

```
  },
  "CreationTimestamp": 1532449292.712
}
```

有关更多信息，[请参阅亚马逊 Rekognition 开发者指南中的使用流媒体视频](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStreamProcessor](#)中的。

## detect-faces

以下代码示例演示了如何使用 detect-faces。

有关更多信息，请参阅[检测图像中的人脸](#)。

### AWS CLI

检测图像中的人脸

以下 detect-faces 命令将检测存储在 Amazon S3 存储桶中的指定图像中的人脸。

```
aws rekognition detect-faces \
  --image '{"S3Object":{"Bucket":"MyImageS3Bucket","Name":"MyFriend.jpg"}}' \
  --attributes "ALL"
```

输出：

```
{
  "FaceDetails": [
    {
      "Confidence": 100.0,
      "Eyeglasses": {
        "Confidence": 98.91107940673828,
        "Value": false
      },
      "Sunglasses": {
        "Confidence": 99.7966537475586,
        "Value": false
      },
      "Gender": {
        "Confidence": 99.56611633300781,
        "Value": "Male"
      },
      "Landmarks": [
```

```
{
  "Y": 0.26721030473709106,
  "X": 0.6204193830490112,
  "Type": "eyeLeft"
},
{
  "Y": 0.26831310987472534,
  "X": 0.6776827573776245,
  "Type": "eyeRight"
},
{
  "Y": 0.3514654338359833,
  "X": 0.6241428852081299,
  "Type": "mouthLeft"
},
{
  "Y": 0.35258132219314575,
  "X": 0.6713621020317078,
  "Type": "mouthRight"
},
{
  "Y": 0.3140771687030792,
  "X": 0.6428444981575012,
  "Type": "nose"
},
{
  "Y": 0.24662546813488007,
  "X": 0.6001564860343933,
  "Type": "leftEyeBrowLeft"
},
{
  "Y": 0.24326619505882263,
  "X": 0.6303644776344299,
  "Type": "leftEyeBrowRight"
},
{
  "Y": 0.23818562924861908,
  "X": 0.6146903038024902,
  "Type": "leftEyeBrowUp"
},
{
  "Y": 0.24373626708984375,
  "X": 0.6640064716339111,
  "Type": "rightEyeBrowLeft"
}
```

```
    },
    {
      "Y": 0.24877218902111053,
      "X": 0.7025929093360901,
      "Type": "rightEyeBrowRight"
    },
    {
      "Y": 0.23938551545143127,
      "X": 0.6823262572288513,
      "Type": "rightEyeBrowUp"
    },
    {
      "Y": 0.265746533870697,
      "X": 0.6112898588180542,
      "Type": "leftEyeLeft"
    },
    {
      "Y": 0.2676128149032593,
      "X": 0.6317071914672852,
      "Type": "leftEyeRight"
    },
    {
      "Y": 0.262735515832901,
      "X": 0.6201658248901367,
      "Type": "leftEyeUp"
    },
    {
      "Y": 0.27025148272514343,
      "X": 0.6206279993057251,
      "Type": "leftEyeDown"
    },
    {
      "Y": 0.268223375082016,
      "X": 0.6658390760421753,
      "Type": "rightEyeLeft"
    },
    {
      "Y": 0.2672517001628876,
      "X": 0.687832236289978,
      "Type": "rightEyeRight"
    },
    {
      "Y": 0.26383838057518005,
      "X": 0.6769183874130249,
```

```
    "Type": "rightEyeUp"
  },
  {
    "Y": 0.27138751745224,
    "X": 0.676596462726593,
    "Type": "rightEyeDown"
  },
  {
    "Y": 0.32283174991607666,
    "X": 0.6350004076957703,
    "Type": "noseLeft"
  },
  {
    "Y": 0.3219289481639862,
    "X": 0.6567046642303467,
    "Type": "noseRight"
  },
  {
    "Y": 0.3420318365097046,
    "X": 0.6450609564781189,
    "Type": "mouthUp"
  },
  {
    "Y": 0.3664324879646301,
    "X": 0.6455618143081665,
    "Type": "mouthDown"
  },
  {
    "Y": 0.26721030473709106,
    "X": 0.6204193830490112,
    "Type": "leftPupil"
  },
  {
    "Y": 0.26831310987472534,
    "X": 0.6776827573776245,
    "Type": "rightPupil"
  },
  {
    "Y": 0.26343393325805664,
    "X": 0.5946047306060791,
    "Type": "upperJawlineLeft"
  },
  {
    "Y": 0.3543180525302887,
```

```
        "X": 0.6044883728027344,
        "Type": "midJawlineLeft"
    },
    {
        "Y": 0.4084877669811249,
        "X": 0.6477024555206299,
        "Type": "chinBottom"
    },
    {
        "Y": 0.3562754988670349,
        "X": 0.707981526851654,
        "Type": "midJawlineRight"
    },
    {
        "Y": 0.26580461859703064,
        "X": 0.7234612107276917,
        "Type": "upperJawlineRight"
    }
],
"Pose": {
    "Yaw": -3.7351467609405518,
    "Roll": -0.10309021919965744,
    "Pitch": 0.8637830018997192
},
"Emotions": [
    {
        "Confidence": 8.74203109741211,
        "Type": "SURPRISED"
    },
    {
        "Confidence": 2.501944065093994,
        "Type": "ANGRY"
    },
    {
        "Confidence": 0.7378743290901184,
        "Type": "DISGUSTED"
    },
    {
        "Confidence": 3.5296201705932617,
        "Type": "HAPPY"
    },
    {
        "Confidence": 1.7162904739379883,
        "Type": "SAD"
    }
]
```

```
    },
    {
      "Confidence": 9.518536567687988,
      "Type": "CONFUSED"
    },
    {
      "Confidence": 0.45474427938461304,
      "Type": "FEAR"
    },
    {
      "Confidence": 72.79895782470703,
      "Type": "CALM"
    }
  ],
  "AgeRange": {
    "High": 48,
    "Low": 32
  },
  "EyesOpen": {
    "Confidence": 98.93987274169922,
    "Value": true
  },
  "BoundingBox": {
    "Width": 0.12368916720151901,
    "Top": 0.16007372736930847,
    "Left": 0.5901257991790771,
    "Height": 0.25140416622161865
  },
  "Smile": {
    "Confidence": 93.4493179321289,
    "Value": false
  },
  "MouthOpen": {
    "Confidence": 90.53053283691406,
    "Value": false
  },
  "Quality": {
    "Sharpness": 95.51618957519531,
    "Brightness": 65.29893493652344
  },
  "Mustache": {
    "Confidence": 89.85221099853516,
    "Value": false
  },
},
```

```
        "Beard": {
            "Confidence": 86.1991195678711,
            "Value": true
        }
    ]
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[检测图像中的人脸](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetectFaces](#)中的。

## detect-labels

以下代码示例演示了如何使用 detect-labels。

有关更多信息，请参阅[检测图像中的标签](#)。

### AWS CLI

#### 检测图像中的标签

以下 detect-labels 示例将检测存储在 Amazon S3 存储桶中的图像中的场景和对象。

```
aws rekognition detect-labels \
  --image '{"S3Object":{"Bucket":"bucket","Name":"image"}}'
```

输出：

```
{
  "Labels": [
    {
      "Instances": [],
      "Confidence": 99.15271759033203,
      "Parents": [
        {
          "Name": "Vehicle"
        },
        {
          "Name": "Transportation"
        }
      ],
      "Name": "Automobile"
    }
  ]
}
```



```
  },
  {
    "Instances": [],
    "Confidence": 99.15271759033203,
    "Parents": [
      {
        "Name": "Transportation"
      }
    ],
    "Name": "Vehicle"
  },
  {
    "Instances": [],
    "Confidence": 99.15271759033203,
    "Parents": [],
    "Name": "Transportation"
  },
  {
    "Instances": [
      {
        "BoundingBox": {
          "Width": 0.10616336017847061,
          "Top": 0.5039216876029968,
          "Left": 0.0037978808395564556,
          "Height": 0.18528179824352264
        },
        "Confidence": 99.15271759033203
      },
      {
        "BoundingBox": {
          "Width": 0.2429988533258438,
          "Top": 0.5251884460449219,
          "Left": 0.7309805154800415,
          "Height": 0.21577216684818268
        },
        "Confidence": 99.1286392211914
      },
      {
        "BoundingBox": {
          "Width": 0.14233611524105072,
          "Top": 0.5333095788955688,
          "Left": 0.6494812965393066,
          "Height": 0.15528248250484467
        }
      },
    ]
  },
```

```
    "Confidence": 98.48368072509766
  },
  {
    "BoundingBox": {
      "Width": 0.11086395382881165,
      "Top": 0.5354844927787781,
      "Left": 0.10355594009160995,
      "Height": 0.10271988064050674
    },
    "Confidence": 96.45606231689453
  },
  {
    "BoundingBox": {
      "Width": 0.06254628300666809,
      "Top": 0.5573825240135193,
      "Left": 0.46083059906959534,
      "Height": 0.053911514580249786
    },
    "Confidence": 93.65448760986328
  },
  {
    "BoundingBox": {
      "Width": 0.10105438530445099,
      "Top": 0.534368634223938,
      "Left": 0.5743985772132874,
      "Height": 0.12226245552301407
    },
    "Confidence": 93.06217193603516
  },
  {
    "BoundingBox": {
      "Width": 0.056389667093753815,
      "Top": 0.5235804319381714,
      "Left": 0.9427769780158997,
      "Height": 0.17163699865341187
    },
    "Confidence": 92.6864013671875
  },
  {
    "BoundingBox": {
      "Width": 0.06003860384225845,
      "Top": 0.5441341400146484,
      "Left": 0.22409997880458832,
      "Height": 0.06737709045410156
    }
  }
}
```

```
    },
    "Confidence": 90.4227066040039
  },
  {
    "BoundingBox": {
      "Width": 0.02848697081208229,
      "Top": 0.5107086896896362,
      "Left": 0,
      "Height": 0.19150497019290924
    },
    "Confidence": 86.65286254882812
  },
  {
    "BoundingBox": {
      "Width": 0.04067881405353546,
      "Top": 0.5566273927688599,
      "Left": 0.316415935754776,
      "Height": 0.03428703173995018
    },
    "Confidence": 85.36471557617188
  },
  {
    "BoundingBox": {
      "Width": 0.043411049991846085,
      "Top": 0.5394920110702515,
      "Left": 0.18293385207653046,
      "Height": 0.0893595889210701
    },
    "Confidence": 82.21705627441406
  },
  {
    "BoundingBox": {
      "Width": 0.031183116137981415,
      "Top": 0.5579366683959961,
      "Left": 0.2853088080883026,
      "Height": 0.03989990055561066
    },
    "Confidence": 81.0157470703125
  },
  {
    "BoundingBox": {
      "Width": 0.031113790348172188,
      "Top": 0.5504819750785828,
      "Left": 0.2580395042896271,
```

```
        "Height": 0.056484755128622055
      },
      "Confidence": 56.13441467285156
    },
    {
      "BoundingBox": {
        "Width": 0.08586374670267105,
        "Top": 0.5438792705535889,
        "Left": 0.5128012895584106,
        "Height": 0.08550430089235306
      },
      "Confidence": 52.37760925292969
    }
  ],
  "Confidence": 99.15271759033203,
  "Parents": [
    {
      "Name": "Vehicle"
    },
    {
      "Name": "Transportation"
    }
  ],
  "Name": "Car"
},
{
  "Instances": [],
  "Confidence": 98.9914321899414,
  "Parents": [],
  "Name": "Human"
},
{
  "Instances": [
    {
      "BoundingBox": {
        "Width": 0.19360728561878204,
        "Top": 0.35072067379951477,
        "Left": 0.43734854459762573,
        "Height": 0.2742200493812561
      },
      "Confidence": 98.9914321899414
    },
    {
      "BoundingBox": {
```

```
        "Width": 0.03801717236638069,
        "Top": 0.5010883808135986,
        "Left": 0.9155802130699158,
        "Height": 0.06597328186035156
      },
      "Confidence": 85.02790832519531
    }
  ],
  "Confidence": 98.9914321899414,
  "Parents": [],
  "Name": "Person"
},
{
  "Instances": [],
  "Confidence": 93.24951934814453,
  "Parents": [],
  "Name": "Machine"
},
{
  "Instances": [
    {
      "BoundingBox": {
        "Width": 0.03561960905790329,
        "Top": 0.6468243598937988,
        "Left": 0.7850857377052307,
        "Height": 0.08878646790981293
      },
      "Confidence": 93.24951934814453
    },
    {
      "BoundingBox": {
        "Width": 0.02217046171426773,
        "Top": 0.6149078607559204,
        "Left": 0.04757237061858177,
        "Height": 0.07136218994855881
      },
      "Confidence": 91.5025863647461
    },
    {
      "BoundingBox": {
        "Width": 0.016197510063648224,
        "Top": 0.6274210214614868,
        "Left": 0.6472989320755005,
        "Height": 0.04955997318029404
      }
    }
  ]
}
```

```
    },
    "Confidence": 85.14686584472656
  },
  {
    "BoundingBox": {
      "Width": 0.020207518711686134,
      "Top": 0.6348286867141724,
      "Left": 0.7295016646385193,
      "Height": 0.07059963047504425
    },
    "Confidence": 83.34547424316406
  },
  {
    "BoundingBox": {
      "Width": 0.020280985161662102,
      "Top": 0.6171894669532776,
      "Left": 0.08744934946298599,
      "Height": 0.05297485366463661
    },
    "Confidence": 79.9981460571289
  },
  {
    "BoundingBox": {
      "Width": 0.018318990245461464,
      "Top": 0.623889148235321,
      "Left": 0.6836880445480347,
      "Height": 0.06730121374130249
    },
    "Confidence": 78.87144470214844
  },
  {
    "BoundingBox": {
      "Width": 0.021310249343514442,
      "Top": 0.6167286038398743,
      "Left": 0.004064912907779217,
      "Height": 0.08317798376083374
    },
    "Confidence": 75.89361572265625
  },
  {
    "BoundingBox": {
      "Width": 0.03604431077837944,
      "Top": 0.7030032277107239,
      "Left": 0.9254803657531738,
```

```
        "Height": 0.04569442570209503
      },
      "Confidence": 64.402587890625
    },
    {
      "BoundingBox": {
        "Width": 0.009834849275648594,
        "Top": 0.5821820497512817,
        "Left": 0.28094568848609924,
        "Height": 0.01964157074689865
      },
      "Confidence": 62.79907989501953
    },
    {
      "BoundingBox": {
        "Width": 0.01475677452981472,
        "Top": 0.6137543320655823,
        "Left": 0.5950819253921509,
        "Height": 0.039063986390829086
      },
      "Confidence": 59.40483474731445
    }
  ],
  "Confidence": 93.24951934814453,
  "Parents": [
    {
      "Name": "Machine"
    }
  ],
  "Name": "Wheel"
},
{
  "Instances": [],
  "Confidence": 92.61514282226562,
  "Parents": [],
  "Name": "Road"
},
{
  "Instances": [],
  "Confidence": 92.37877655029297,
  "Parents": [
    {
      "Name": "Person"
    }
  ]
}
```

```
    ],
    "Name": "Sport"
  },
  {
    "Instances": [],
    "Confidence": 92.37877655029297,
    "Parents": [
      {
        "Name": "Person"
      }
    ],
    "Name": "Sports"
  },
  {
    "Instances": [
      {
        "BoundingBox": {
          "Width": 0.12326609343290329,
          "Top": 0.6332163214683533,
          "Left": 0.44815489649772644,
          "Height": 0.058117982000112534
        },
        "Confidence": 92.37877655029297
      }
    ],
    "Confidence": 92.37877655029297,
    "Parents": [
      {
        "Name": "Person"
      },
      {
        "Name": "Sport"
      }
    ],
    "Name": "Skateboard"
  },
  {
    "Instances": [],
    "Confidence": 90.62931060791016,
    "Parents": [
      {
        "Name": "Person"
      }
    ]
  },
],
```



```
    "Name": "Pedestrian"
  },
  {
    "Instances": [],
    "Confidence": 88.81334686279297,
    "Parents": [],
    "Name": "Asphalt"
  },
  {
    "Instances": [],
    "Confidence": 88.81334686279297,
    "Parents": [],
    "Name": "Tarmac"
  },
  {
    "Instances": [],
    "Confidence": 88.23201751708984,
    "Parents": [],
    "Name": "Path"
  },
  {
    "Instances": [],
    "Confidence": 80.26520538330078,
    "Parents": [],
    "Name": "Urban"
  },
  {
    "Instances": [],
    "Confidence": 80.26520538330078,
    "Parents": [
      {
        "Name": "Building"
      },
      {
        "Name": "Urban"
      }
    ],
    "Name": "Town"
  },
  {
    "Instances": [],
    "Confidence": 80.26520538330078,
    "Parents": [],
    "Name": "Building"
```

```
  },
  {
    "Instances": [],
    "Confidence": 80.26520538330078,
    "Parents": [
      {
        "Name": "Building"
      },
      {
        "Name": "Urban"
      }
    ],
    "Name": "City"
  },
  {
    "Instances": [],
    "Confidence": 78.37934875488281,
    "Parents": [
      {
        "Name": "Car"
      },
      {
        "Name": "Vehicle"
      },
      {
        "Name": "Transportation"
      }
    ],
    "Name": "Parking Lot"
  },
  {
    "Instances": [],
    "Confidence": 78.37934875488281,
    "Parents": [
      {
        "Name": "Car"
      },
      {
        "Name": "Vehicle"
      },
      {
        "Name": "Transportation"
      }
    ],
  },
```

```
    "Name": "Parking"
  },
  {
    "Instances": [],
    "Confidence": 74.37590026855469,
    "Parents": [
      {
        "Name": "Building"
      },
      {
        "Name": "Urban"
      },
      {
        "Name": "City"
      }
    ],
    "Name": "Downtown"
  },
  {
    "Instances": [],
    "Confidence": 69.84622955322266,
    "Parents": [
      {
        "Name": "Road"
      }
    ],
    "Name": "Intersection"
  },
  {
    "Instances": [],
    "Confidence": 57.68518829345703,
    "Parents": [
      {
        "Name": "Sports Car"
      },
      {
        "Name": "Car"
      },
      {
        "Name": "Vehicle"
      },
      {
        "Name": "Transportation"
      }
    ]
  }
```

```
    ],
    "Name": "Coupe"
  },
  {
    "Instances": [],
    "Confidence": 57.68518829345703,
    "Parents": [
      {
        "Name": "Car"
      },
      {
        "Name": "Vehicle"
      },
      {
        "Name": "Transportation"
      }
    ],
    "Name": "Sports Car"
  },
  {
    "Instances": [],
    "Confidence": 56.59492111206055,
    "Parents": [
      {
        "Name": "Path"
      }
    ],
    "Name": "Sidewalk"
  },
  {
    "Instances": [],
    "Confidence": 56.59492111206055,
    "Parents": [
      {
        "Name": "Path"
      }
    ],
    "Name": "Pavement"
  },
  {
    "Instances": [],
    "Confidence": 55.58770751953125,
    "Parents": [
      {
```

```
        "Name": "Building"
      },
      {
        "Name": "Urban"
      }
    ],
    "Name": "Neighborhood"
  }
],
"LabelModelVersion": "2.0"
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[检测图像中的标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetectLabels](#)中的。

## detect-moderation-labels

以下代码示例演示了如何使用 detect-moderation-labels。

有关更多信息，请参阅[检测不适宜的图像](#)。

### AWS CLI

检测图像中不安全的内容

以下 detect-moderation-labels 命令将检测存储在 Amazon S3 存储桶中的指定图像中不安全的内容。

```
aws rekognition detect-moderation-labels \
  --image "S3Object={Bucket=MyImageS3Bucket,Name=gun.jpg}"
```

输出：

```
{
  "ModerationModelVersion": "3.0",
  "ModerationLabels": [
    {
      "Confidence": 97.29618072509766,
      "ParentName": "Violence",
      "Name": "Weapon Violence"
    },
    {
```

```
        "Confidence": 97.29618072509766,  
        "ParentName": "",  
        "Name": "Violence"  
    }  
]  
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[检测不安全的图像](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DetectModerationLabels](#) 中的。

## detect-text

以下代码示例演示了如何使用 detect-text。

有关更多信息，请参阅[检测图像中的文本](#)。

### AWS CLI

检测图像中的文本

以下 detect-text 命令将检测指定图像中的文本。

```
aws rekognition detect-text \  
  --image '{"S3Object":{"Bucket":"MyImageS3Bucket","Name":"ExamplePicture.jpg"}}'
```

输出：

```
{  
  "TextDetections": [  
    {  
      "Geometry": {  
        "BoundingBox": {  
          "Width": 0.24624845385551453,  
          "Top": 0.28288066387176514,  
          "Left": 0.391388863325119,  
          "Height": 0.022687450051307678  
        },  
        "Polygon": [  
          {  
            "Y": 0.28288066387176514,  
            "X": 0.391388863325119  
          },  
          {  
            "Y": 0.28288066387176514,  
            "X": 0.391388863325119  
          }  
        ]  
      }  
    }  
  ]  
}
```

```
        {
            "Y": 0.2826388478279114,
            "X": 0.6376373171806335
        },
        {
            "Y": 0.30532628297805786,
            "X": 0.637677013874054
        },
        {
            "Y": 0.305568128824234,
            "X": 0.39142853021621704
        }
    ]
},
"Confidence": 94.35709381103516,
"DetectedText": "ESTD 1882",
"Type": "LINE",
"Id": 0
},
{
    "Geometry": {
        "BoundingBox": {
            "Width": 0.33933889865875244,
            "Top": 0.32603850960731506,
            "Left": 0.34534579515457153,
            "Height": 0.07126858830451965
        },
        "Polygon": [
            {
                "Y": 0.32603850960731506,
                "X": 0.34534579515457153
            },
            {
                "Y": 0.32633158564567566,
                "X": 0.684684693813324
            },
            {
                "Y": 0.3976001739501953,
                "X": 0.684575080871582
            },
            {
                "Y": 0.3973070979118347,
                "X": 0.345236212015152
            }
        ]
    }
}
```

```
    ]
  },
  "Confidence": 99.95779418945312,
  "DetectedText": "BRAINS",
  "Type": "LINE",
  "Id": 1
},
{
  "Confidence": 97.22098541259766,
  "Geometry": {
    "BoundingBox": {
      "Width": 0.061079490929841995,
      "Top": 0.2843210697174072,
      "Left": 0.391391396522522,
      "Height": 0.021029088646173477
    },
    "Polygon": [
      {
        "Y": 0.2843210697174072,
        "X": 0.391391396522522
      },
      {
        "Y": 0.2828207015991211,
        "X": 0.4524524509906769
      },
      {
        "Y": 0.3038259446620941,
        "X": 0.4534534513950348
      },
      {
        "Y": 0.30532634258270264,
        "X": 0.3923923969268799
      }
    ]
  },
  "DetectedText": "ESTD",
  "ParentId": 0,
  "Type": "WORD",
  "Id": 2
},
{
  "Confidence": 91.49320983886719,
  "Geometry": {
    "BoundingBox": {
```



```
        "Width": 0.07007007300853729,
        "Top": 0.2828207015991211,
        "Left": 0.5675675868988037,
        "Height": 0.02250562608242035
    },
    "Polygon": [
        {
            "Y": 0.2828207015991211,
            "X": 0.5675675868988037
        },
        {
            "Y": 0.2828207015991211,
            "X": 0.6376376152038574
        },
        {
            "Y": 0.30532634258270264,
            "X": 0.6376376152038574
        },
        {
            "Y": 0.30532634258270264,
            "X": 0.5675675868988037
        }
    ]
},
"DetectedText": "1882",
"ParentId": 0,
"Type": "WORD",
"Id": 3
},
{
    "Confidence": 99.95779418945312,
    "Geometry": {
        "BoundingBox": {
            "Width": 0.33933934569358826,
            "Top": 0.32633158564567566,
            "Left": 0.3453453481197357,
            "Height": 0.07127484679222107
        },
        "Polygon": [
            {
                "Y": 0.32633158564567566,
                "X": 0.3453453481197357
            },
            {
```

```

        "Y": 0.32633158564567566,
        "X": 0.684684693813324
      },
      {
        "Y": 0.39759939908981323,
        "X": 0.6836836934089661
      },
      {
        "Y": 0.39684921503067017,
        "X": 0.3453453481197357
      }
    ]
  },
  "DetectedText": "BRAINS",
  "ParentId": 1,
  "Type": "WORD",
  "Id": 4
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetectText](#)中的。

## disassociate-faces

以下代码示例演示了如何使用 disassociate-faces。

### AWS CLI

```
aws rekognition disassociate-faces --face-ids list-of-face-ids
--user-id user-id --collection-id collection-name --region region-name
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateFaces](#)中的。

## get-celebrity-info

以下代码示例演示了如何使用 get-celebrity-info。

### AWS CLI

获取有关名人的信息

以下 `get-celebrity-info` 命令显示有关指定名人的信息：`id` 参数来自先前对 `recognize-celebrities` 的调用。

```
aws rekognition get-celebrity-info --id nnnnnnn
```

输出：

```
{
  "Name": "Celeb A",
  "Urls": [
    "www.imdb.com/name/aaaaaaaaa"
  ]
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[获取有关名人的信息](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetCelebrityInfo](#) 中的。

## get-celebrity-recognition

以下代码示例演示了如何使用 `get-celebrity-recognition`。

### AWS CLI

为了获得名人认可操作的结果

以下 `get-celebrity-recognition` 命令显示您之前通过调用 `start-celebrity-recognition` 启动的名人识别操作的结果。

```
aws rekognition get-celebrity-recognition \
  --job-id 1234567890abcdef1234567890abcdef1234567890abcdef
```

输出：

```
{
  "NextToken": "3D01Clx1CiT31VsRDkA03IybLb/h5AtDWSGuhYi
+N1FIJwwPtAkuKzDhL2rV3GcwmNt77+12",
  "Celebrities": [
    {
      "Timestamp": 0,
      "Celebrity": {
        "Confidence": 96.0,

```

```
"Face": {
  "BoundingBox": {
    "Width": 0.70333331823349,
    "Top": 0.16750000417232513,
    "Left": 0.19555555284023285,
    "Height": 0.3956249952316284
  },
  "Landmarks": [
    {
      "Y": 0.31031012535095215,
      "X": 0.441436767578125,
      "Type": "eyeLeft"
    },
    {
      "Y": 0.3081788718700409,
      "X": 0.6437258720397949,
      "Type": "eyeRight"
    },
    {
      "Y": 0.39542075991630554,
      "X": 0.5572493076324463,
      "Type": "nose"
    },
    {
      "Y": 0.4597957134246826,
      "X": 0.4579732120037079,
      "Type": "mouthLeft"
    },
    {
      "Y": 0.45688048005104065,
      "X": 0.6349081993103027,
      "Type": "mouthRight"
    }
  ],
  "Pose": {
    "Yaw": 8.943398475646973,
    "Roll": -2.0309247970581055,
    "Pitch": -0.5674862861633301
  },
  "Quality": {
    "Sharpness": 99.40211486816406,
    "Brightness": 89.47132110595703
  },
  "Confidence": 99.99861145019531
}
```

```
    },
    "Name": "CelebrityA",
    "Urls": [
      "www.imdb.com/name/111111111"
    ],
    "Id": "nnnnnn"
  }
},
{
  "Timestamp": 467,
  "Celebrity": {
    "Confidence": 99.0,
    "Face": {
      "BoundingBox": {
        "Width": 0.6877777576446533,
        "Top": 0.18437500298023224,
        "Left": 0.20555555820465088,
        "Height": 0.3868750035762787
      },
      "Landmarks": [
        {
          "Y": 0.31895750761032104,
          "X": 0.4411413371562958,
          "Type": "eyeLeft"
        },
        {
          "Y": 0.3140959143638611,
          "X": 0.6523157954216003,
          "Type": "eyeRight"
        },
        {
          "Y": 0.4016456604003906,
          "X": 0.5682755708694458,
          "Type": "nose"
        },
        {
          "Y": 0.46894142031669617,
          "X": 0.4597797095775604,
          "Type": "mouthLeft"
        },
        {
          "Y": 0.46971091628074646,
          "X": 0.6286435127258301,
          "Type": "mouthRight"
        }
      ]
    }
  }
}
```

```
    }
    ],
    "Pose": {
      "Yaw": 10.433465957641602,
      "Roll": -3.347442388534546,
      "Pitch": 1.3709543943405151
    },
    "Quality": {
      "Sharpness": 99.5531005859375,
      "Brightness": 88.5764389038086
    },
    "Confidence": 99.99148559570312
  },
  "Name": "Jane Celebrity",
  "Urls": [
    "www.imdb.com/name/111111111"
  ],
  "Id": "nnnnnn"
}
]
"JobStatus": "SUCCEEDED",
"VideoMetadata": {
  "Format": "QuickTime / MOV",
  "FrameRate": 29.978118896484375,
  "Codec": "h264",
  "DurationMillis": 4570,
  "FrameHeight": 1920,
  "FrameWidth": 1080
}
}
```

有关更多信息，请参阅亚马逊 Rekognition 开发者指南中的识别[存储视频中的名人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCelebrityRecognition](#)中的。

## get-content-moderation

以下代码示例演示了如何使用 get-content-moderation。

### AWS CLI

#### 获取不安全内容操作的结果

以下`get-content-moderation`命令显示您之前通过调用启动的不安全内容操作的结果`start-content-moderation`。

```
aws rekognition get-content-moderation \  
  --job-id 1234567890abcdef1234567890abcdef1234567890abcdef
```

输出：

```
{  
  "NextToken": "dlhcKMHMzpCBGFukz6I03JMcWiJAamCVhXHt3r6b4b5Tfbyw3q7o+Jeezt  
+Zpgf0nW9FCCgQ",  
  "ModerationLabels": [  
    {  
      "Timestamp": 0,  
      "ModerationLabel": {  
        "Confidence": 97.39583587646484,  
        "ParentName": "",  
        "Name": "Violence"  
      }  
    },  
    {  
      "Timestamp": 0,  
      "ModerationLabel": {  
        "Confidence": 97.39583587646484,  
        "ParentName": "Violence",  
        "Name": "Weapon Violence"  
      }  
    }  
  ],  
  "JobStatus": "SUCCEEDED",  
  "VideoMetadata": {  
    "Format": "QuickTime / MOV",  
    "FrameRate": 29.97515869140625,  
    "Codec": "h264",  
    "DurationMillis": 6039,  
    "FrameHeight": 1920,  
    "FrameWidth": 1080  
  }  
}
```

有关更多信息，请参阅《亚马逊 Rekognition 开发者指南》中的“[检测存储的不安全视频](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetContentModeration](#)中的。

## get-face-detection

以下代码示例演示了如何使用 `get-face-detection`。

### AWS CLI

获取人脸检测操作的结果

以下 `get-face-detection` 命令显示您之前通过调用启动的人脸检测操作的结果 `start-face-detection`。

```
aws rekognition get-face-detection \  
  --job-id 1234567890abcdef1234567890abcdef1234567890abcdef
```

输出：

```
{  
  "Faces": [  
    {  
      "Timestamp": 467,  
      "Face": {  
        "BoundingBox": {  
          "Width": 0.1560753583908081,  
          "Top": 0.13555361330509186,  
          "Left": -0.0952017530798912,  
          "Height": 0.6934483051300049  
        },  
        "Landmarks": [  
          {  
            "Y": 0.4013825058937073,  
            "X": -0.041750285774469376,  
            "Type": "eyeLeft"  
          },  
          {  
            "Y": 0.41695496439933777,  
            "X": 0.027979329228401184,  
            "Type": "eyeRight"  
          },  
          {  
            "Y": 0.6375303268432617,  
            "X": -0.04034662991762161,  
            "Type": "mouthLeft"  
          }  
        ]  
      }  
    ]  
}
```



```
        {
            "Y": 0.6497718691825867,
            "X": 0.013960429467260838,
            "Type": "mouthRight"
        },
        {
            "Y": 0.5238034129142761,
            "X": 0.008022055961191654,
            "Type": "nose"
        }
    ],
    "Pose": {
        "Yaw": -58.07863998413086,
        "Roll": 1.9384294748306274,
        "Pitch": -24.66305160522461
    },
    "Quality": {
        "Sharpness": 83.14741516113281,
        "Brightness": 25.75942611694336
    },
    "Confidence": 87.7622299194336
}
},
{
    "Timestamp": 967,
    "Face": {
        "BoundingBox": {
            "Width": 0.28559377789497375,
            "Top": 0.19436298310756683,
            "Left": 0.024553587660193443,
            "Height": 0.7216082215309143
        },
        "Landmarks": [
            {
                "Y": 0.4650231599807739,
                "X": 0.16269078850746155,
                "Type": "eyeLeft"
            },
            {
                "Y": 0.4843238294124603,
                "X": 0.2782580852508545,
                "Type": "eyeRight"
            }
        ]
    }
}
```

```
        "Y": 0.71530681848526,
        "X": 0.1741468608379364,
        "Type": "mouthLeft"
      },
      {
        "Y": 0.7310671210289001,
        "X": 0.26857468485832214,
        "Type": "mouthRight"
      },
      {
        "Y": 0.582602322101593,
        "X": 0.2566150426864624,
        "Type": "nose"
      }
    ],
    "Pose": {
      "Yaw": 11.487052917480469,
      "Roll": 5.074230670928955,
      "Pitch": 15.396159172058105
    },
    "Quality": {
      "Sharpness": 73.32209777832031,
      "Brightness": 54.96497344970703
    },
    "Confidence": 99.99998474121094
  }
}
],
"NextToken":
"0zL223pDKy91160/02KXRqFIEAwxy4PkgYcm3hSo0rdysbXg5Ex0eFgTGEj0ADEac6S037U",
"JobStatus": "SUCCEEDED",
"VideoMetadata": {
  "Format": "QuickTime / MOV",
  "FrameRate": 29.970617294311523,
  "Codec": "h264",
  "DurationMillis": 6806,
  "FrameHeight": 1080,
  "FrameWidth": 1920
}
}
```

有关更多信息，请参阅亚马逊 Rekognition [开发者指南中的检测存储视频中的人脸](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFaceDetection](#)中的。

## get-face-search

以下代码示例演示了如何使用 `get-face-search`。

### AWS CLI

获取人脸搜索操作的结果

以下 `get-face-search` 命令显示您之前通过调用启动的人脸搜索操作的结果 `start-face-search`。

```
aws rekognition get-face-search \
  --job-id 1234567890abcdef1234567890abcdef1234567890abcdef
```

输出：

```
{
  "Persons": [
    {
      "Timestamp": 467,
      "FaceMatches": [],
      "Person": {
        "Index": 0,
        "Face": {
          "BoundingBox": {
            "Width": 0.1560753583908081,
            "Top": 0.13555361330509186,
            "Left": -0.0952017530798912,
            "Height": 0.6934483051300049
          },
          "Landmarks": [
            {
              "Y": 0.4013825058937073,
              "X": -0.041750285774469376,
              "Type": "eyeLeft"
            },
            {
              "Y": 0.41695496439933777,
              "X": 0.027979329228401184,
              "Type": "eyeRight"
            },
            {
              "Y": 0.6375303268432617,
```

```
        "X": -0.04034662991762161,  
        "Type": "mouthLeft"  
    },  
    {  
        "Y": 0.6497718691825867,  
        "X": 0.013960429467260838,  
        "Type": "mouthRight"  
    },  
    {  
        "Y": 0.5238034129142761,  
        "X": 0.008022055961191654,  
        "Type": "nose"  
    }  
],  
"Pose": {  
    "Yaw": -58.07863998413086,  
    "Roll": 1.9384294748306274,  
    "Pitch": -24.66305160522461  
},  
"Quality": {  
    "Sharpness": 83.14741516113281,  
    "Brightness": 25.75942611694336  
},  
"Confidence": 87.7622299194336  
}  
}  
},  
{  
    "Timestamp": 967,  
    "FaceMatches": [  
        {  
            "Face": {  
                "BoundingBox": {  
                    "Width": 0.12368900328874588,  
                    "Top": 0.16007399559020996,  
                    "Left": 0.5901259779930115,  
                    "Height": 0.2514039874076843  
                },  
                "FaceId": "056a95fa-2060-4159-9cab-7ed4daa030fa",  
                "ExternalImageId": "image3.jpg",  
                "Confidence": 100.0,  
                "ImageId": "08f8a078-8929-37fd-8e8f-aadf690e8232"  
            },  
            "Similarity": 98.44476318359375  
        }  
    ]  
}
```

```
    }
  ],
  "Person": {
    "Index": 1,
    "Face": {
      "BoundingBox": {
        "Width": 0.28559377789497375,
        "Top": 0.19436298310756683,
        "Left": 0.024553587660193443,
        "Height": 0.7216082215309143
      },
      "Landmarks": [
        {
          "Y": 0.4650231599807739,
          "X": 0.16269078850746155,
          "Type": "eyeLeft"
        },
        {
          "Y": 0.4843238294124603,
          "X": 0.2782580852508545,
          "Type": "eyeRight"
        },
        {
          "Y": 0.71530681848526,
          "X": 0.1741468608379364,
          "Type": "mouthLeft"
        },
        {
          "Y": 0.7310671210289001,
          "X": 0.26857468485832214,
          "Type": "mouthRight"
        },
        {
          "Y": 0.582602322101593,
          "X": 0.2566150426864624,
          "Type": "nose"
        }
      ],
      "Pose": {
        "Yaw": 11.487052917480469,
        "Roll": 5.074230670928955,
        "Pitch": 15.396159172058105
      },
      "Quality": {
```

```

        "Sharpness": 73.32209777832031,
        "Brightness": 54.96497344970703
    },
    "Confidence": 99.99998474121094
}
}
}
],
"NextToken": "5bkgcezyuaqhtWk3C80TW6cjRghrwV9XDMivm5B3MXm+Lv6G+L+GejyFHPhoNa/
ldXIC4c/d",
"JobStatus": "SUCCEEDED",
"VideoMetadata": {
    "Format": "QuickTime / MOV",
    "FrameRate": 29.970617294311523,
    "Codec": "h264",
    "DurationMillis": 6806,
    "FrameHeight": 1080,
    "FrameWidth": 1920
}
}
}

```

有关更多信息，请参阅亚马逊 Rekognition [开发者指南中的在存储的视频中搜索人脸](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFaceSearch](#)中的。

## get-label-detection

以下代码示例演示了如何使用 get-label-detection。

### AWS CLI

获取物体和场景检测操作的结果

以下get-label-detection命令显示您之前通过调用启动的对象和场景检测操作的结果start-label-detection。

```
aws rekognition get-label-detection \
  --job-id 1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef
```

输出：

```
{
  "Labels": [
```

```
{
  "Timestamp": 0,
  "Label": {
    "Instances": [],
    "Confidence": 50.19071578979492,
    "Parents": [
      {
        "Name": "Person"
      },
      {
        "Name": "Crowd"
      }
    ],
    "Name": "Audience"
  }
},
{
  "Timestamp": 0,
  "Label": {
    "Instances": [],
    "Confidence": 55.74115753173828,
    "Parents": [
      {
        "Name": "Room"
      },
      {
        "Name": "Indoors"
      },
      {
        "Name": "School"
      }
    ],
    "Name": "Classroom"
  }
}
],
"JobStatus": "SUCCEEDED",
"LabelModelVersion": "2.0",
"VideoMetadata": {
  "Format": "QuickTime / MOV",
  "FrameRate": 29.970617294311523,
  "Codec": "h264",
  "DurationMillis": 6806,
  "FrameHeight": 1080,
```

```
    "FrameWidth": 1920
  },
  "NextToken": "BMugzAi4L72IERzQdbpyMQuEFBsjl05W0Yx3mfG+sR9mm98E1/
Cp0benspRfs/5FBQFs4X7G"
}
```

有关更多信息，请参阅亚马逊 Rekognition 开发者指南 [中的检测视频中的标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetLabelDetection](#) 中的。

## get-person-tracking

以下代码示例演示了如何使用 get-person-tracking。

### AWS CLI

获取人员路径分析操作的结果

以下 get-person-tracking 命令显示您之前通过调用 start-person-tracking 启动的人员路径分析操作的结果。

```
aws rekognition get-person-tracking \
  --job-id 1234567890abcdef1234567890abcdef1234567890abcdef
```

输出：

```
{
  "Persons": [
    {
      "Timestamp": 500,
      "Person": {
        "BoundingBox": {
          "Width": 0.4151041805744171,
          "Top": 0.07870370149612427,
          "Left": 0.0,
          "Height": 0.9212962985038757
        },
        "Index": 0
      }
    },
    {
      "Timestamp": 567,
```



```

    "Person": {
      "BoundingBox": {
        "Width": 0.4755208194255829,
        "Top": 0.077777778059244156,
        "Left": 0.0,
        "Height": 0.91944444417953491
      },
      "Index": 0
    }
  ],
  "NextToken": "D/vRIYNyhG79ugdta3f+8cRg9oSRO
+HigG0uxRiYpTn0ExnqTi1CJektVAc4HrAXDv25eHYk",
  "JobStatus": "SUCCEEDED",
  "VideoMetadata": {
    "Format": "QuickTime / MOV",
    "FrameRate": 29.970617294311523,
    "Codec": "h264",
    "DurationMillis": 6806,
    "FrameHeight": 1080,
    "FrameWidth": 1920
  }
}

```

有关更多信息，请参阅亚马逊 Rekognition 开发者指南中的[人员路径](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetPersonTracking](#)中的。

## index-faces

以下代码示例演示了如何使用 `index-faces`。

有关更多信息，请参阅[将人脸添加到集合中](#)。

### AWS CLI

#### 将人脸添加到集合

以下 `index-faces` 命令将在图像中找到的人脸添加到指定的集合中。

```

aws rekognition index-faces \
  --image '{"S3Object":{"Bucket":"MyVideoS3Bucket","Name":"MyPicture.jpg"}}' \
  --collection-id MyCollection \

```

```
--max-faces 1 \  
--quality-filter "AUTO" \  
--detection-attributes "ALL" \  
--external-image-id "MyPicture.jpg"
```

输出：

```
{  
  "FaceRecords": [  
    {  
      "FaceDetail": {  
        "Confidence": 99.993408203125,  
        "Eyeglasses": {  
          "Confidence": 99.11750030517578,  
          "Value": false  
        },  
        "Sunglasses": {  
          "Confidence": 99.98249053955078,  
          "Value": false  
        },  
        "Gender": {  
          "Confidence": 99.92769622802734,  
          "Value": "Male"  
        },  
        "Landmarks": [  
          {  
            "Y": 0.26750367879867554,  
            "X": 0.6202793717384338,  
            "Type": "eyeLeft"  
          },  
          {  
            "Y": 0.26642778515815735,  
            "X": 0.6787431836128235,  
            "Type": "eyeRight"  
          },  
          {  
            "Y": 0.31361380219459534,  
            "X": 0.6421601176261902,  
            "Type": "nose"  
          },  
          {  
            "Y": 0.3495299220085144,  
            "X": 0.6216195225715637,  
            "Type": "mouthLeft"  
          }  
        ]  
      }  
    }  
  ]  
}
```

```
    "Type": "mouthLeft"
  },
  {
    "Y": 0.35194727778434753,
    "X": 0.669899046421051,
    "Type": "mouthRight"
  },
  {
    "Y": 0.26844894886016846,
    "X": 0.6210268139839172,
    "Type": "leftPupil"
  },
  {
    "Y": 0.26707562804222107,
    "X": 0.6817160844802856,
    "Type": "rightPupil"
  },
  {
    "Y": 0.24834522604942322,
    "X": 0.6018546223640442,
    "Type": "leftEyeBrowLeft"
  },
  {
    "Y": 0.24397172033786774,
    "X": 0.6172008514404297,
    "Type": "leftEyeBrowUp"
  },
  {
    "Y": 0.24677404761314392,
    "X": 0.6339119076728821,
    "Type": "leftEyeBrowRight"
  },
  {
    "Y": 0.24582654237747192,
    "X": 0.6619398593902588,
    "Type": "rightEyeBrowLeft"
  },
  {
    "Y": 0.23973053693771362,
    "X": 0.6804757118225098,
    "Type": "rightEyeBrowUp"
  },
  {
    "Y": 0.24441994726657867,
```

```
        "X": 0.6978968977928162,  
        "Type": "rightEyeBrowRight"  
    },  
    {  
        "Y": 0.2695908546447754,  
        "X": 0.6085202693939209,  
        "Type": "leftEyeLeft"  
    },  
    {  
        "Y": 0.26716896891593933,  
        "X": 0.6315826177597046,  
        "Type": "leftEyeRight"  
    },  
    {  
        "Y": 0.26289820671081543,  
        "X": 0.6202316880226135,  
        "Type": "leftEyeUp"  
    },  
    {  
        "Y": 0.27123287320137024,  
        "X": 0.6205548048019409,  
        "Type": "leftEyeDown"  
    },  
    {  
        "Y": 0.2668408751487732,  
        "X": 0.6663622260093689,  
        "Type": "rightEyeLeft"  
    },  
    {  
        "Y": 0.26741549372673035,  
        "X": 0.6910083889961243,  
        "Type": "rightEyeRight"  
    },  
    {  
        "Y": 0.2614026665687561,  
        "X": 0.6785826086997986,  
        "Type": "rightEyeUp"  
    },  
    {  
        "Y": 0.27075251936912537,  
        "X": 0.6789616942405701,  
        "Type": "rightEyeDown"  
    },  
    {
```

```
        "Y": 0.3211299479007721,  
        "X": 0.6324167847633362,  
        "Type": "noseLeft"  
    },  
    {  
        "Y": 0.32276326417922974,  
        "X": 0.6558475494384766,  
        "Type": "noseRight"  
    },  
    {  
        "Y": 0.34385165572166443,  
        "X": 0.6444970965385437,  
        "Type": "mouthUp"  
    },  
    {  
        "Y": 0.3671635091304779,  
        "X": 0.6459195017814636,  
        "Type": "mouthDown"  
    }  
],  
"Pose": {  
    "Yaw": -9.54541015625,  
    "Roll": -0.5709401965141296,  
    "Pitch": 0.6045494675636292  
},  
"Emotions": [  
    {  
        "Confidence": 39.90074157714844,  
        "Type": "HAPPY"  
    },  
    {  
        "Confidence": 23.38753890991211,  
        "Type": "CALM"  
    },  
    {  
        "Confidence": 5.840933322906494,  
        "Type": "CONFUSED"  
    }  
],  
"AgeRange": {  
    "High": 63,  
    "Low": 45  
},  
"EyesOpen": {
```

```
        "Confidence": 99.80887603759766,  
        "Value": true  
    },  
    "BoundingBox": {  
        "Width": 0.18562500178813934,  
        "Top": 0.1618015021085739,  
        "Left": 0.5575000047683716,  
        "Height": 0.24770642817020416  
    },  
    "Smile": {  
        "Confidence": 99.69740295410156,  
        "Value": false  
    },  
    "MouthOpen": {  
        "Confidence": 99.97393798828125,  
        "Value": false  
    },  
    "Quality": {  
        "Sharpness": 95.54405975341797,  
        "Brightness": 63.867706298828125  
    },  
    "Mustache": {  
        "Confidence": 97.05007934570312,  
        "Value": false  
    },  
    "Beard": {  
        "Confidence": 87.34505462646484,  
        "Value": false  
    }  
},  
"Face": {  
    "BoundingBox": {  
        "Width": 0.18562500178813934,  
        "Top": 0.1618015021085739,  
        "Left": 0.5575000047683716,  
        "Height": 0.24770642817020416  
    },  
    "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",  
    "ExternalImageId": "example-image.jpg",  
    "Confidence": 99.993408203125,  
    "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"  
}  
}  
],
```

```
"UnindexedFaces": [],
"FaceModelVersion": "3.0",
"OrientationCorrection": "ROTATE_0"
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[将人脸添加到集合中](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[IndexFaces](#)中的。

## list-collections

以下代码示例演示了如何使用 list-collections。

有关更多信息，请参阅[列出集合](#)。

### AWS CLI

列出可用的集合

以下list-collections命令列出了 AWS 账户中的可用集合。

```
aws rekognition list-collections
```

输出：

```
{
  "FaceModelVersions": [
    "2.0",
    "3.0",
    "3.0",
    "3.0",
    "4.0",
    "1.0",
    "3.0",
    "4.0",
    "4.0",
    "4.0"
  ],
  "CollectionIds": [
    "MyCollection1",
    "MyCollection2",
    "MyCollection3",
    "MyCollection4",
  ]
}
```

```
    "MyCollection5",  
    "MyCollection6",  
    "MyCollection7",  
    "MyCollection8",  
    "MyCollection9",  
    "MyCollection10"  
  ]  
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[列出集合](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCollections](#)中的。

## list-faces

以下代码示例演示了如何使用 list-faces。

有关更多信息，请参阅[列出集合中的人脸](#)。

### AWS CLI

列出集合中的人脸

以下 list-faces 命令将列出指定集合中的人脸。

```
aws rekognition list-faces \  
  --collection-id MyCollection
```

输出：

```
{  
  "FaceModelVersion": "3.0",  
  "Faces": [  
    {  
      "BoundingBox": {  
        "Width": 0.5216310024261475,  
        "Top": 0.3256250023841858,  
        "Left": 0.13394300639629364,  
        "Height": 0.3918749988079071  
      },  
      "FaceId": "0040279c-0178-436e-b70a-e61b074e96b0",  
      "ExternalImageId": "image1.jpg",  
      "Confidence": 100.0,  
    }  
  ]  
}
```



```
    "ImageId": "f976e487-3719-5e2d-be8b-ea2724c26991"
  },
  {
    "BoundingBox": {
      "Width": 0.5074880123138428,
      "Top": 0.3774999976158142,
      "Left": 0.18302799761295319,
      "Height": 0.3812499940395355
    },
    "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
    "ExternalImageId": "image2.jpg",
    "Confidence": 99.99930572509766,
    "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
  },
  {
    "BoundingBox": {
      "Width": 0.5574039816856384,
      "Top": 0.37187498807907104,
      "Left": 0.14559100568294525,
      "Height": 0.4181250035762787
    },
    "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
    "ExternalImageId": "image3.jpg",
    "Confidence": 99.99960327148438,
    "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
  },
  {
    "BoundingBox": {
      "Width": 0.18562500178813934,
      "Top": 0.1618019938468933,
      "Left": 0.5575000047683716,
      "Height": 0.24770599603652954
    },
    "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
    "ExternalImageId": "image4.jpg",
    "Confidence": 99.99340057373047,
    "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
  },
  {
    "BoundingBox": {
      "Width": 0.5307819843292236,
      "Top": 0.2862499952316284,
      "Left": 0.1564060002565384,
      "Height": 0.3987500071525574
    }
  }
}
```

```
    },
    "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
    "ExternalImageId": "image5.jpg",
    "Confidence": 99.99970245361328,
    "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
  },
  {
    "BoundingBox": {
      "Width": 0.5773710012435913,
      "Top": 0.34437501430511475,
      "Left": 0.12396000325679779,
      "Height": 0.4337500035762787
    },
    "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
    "ExternalImageId": "image6.jpg",
    "Confidence": 100.0,
    "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
  },
  {
    "BoundingBox": {
      "Width": 0.5349419713020325,
      "Top": 0.29124999046325684,
      "Left": 0.16389399766921997,
      "Height": 0.40187498927116394
    },
    "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
    "ExternalImageId": "image7.jpg",
    "Confidence": 99.99979400634766,
    "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
  },
  {
    "BoundingBox": {
      "Width": 0.41499999165534973,
      "Top": 0.09187500178813934,
      "Left": 0.28083300590515137,
      "Height": 0.3112500011920929
    },
    "FaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
    "ExternalImageId": "image8.jpg",
    "Confidence": 99.99769592285156,
    "ImageId": "a294da46-2cb1-5cc4-9045-61d7ca567662"
  },
  {
    "BoundingBox": {
```

```
        "Width": 0.48166701197624207,
        "Top": 0.20999999344348907,
        "Left": 0.21250000596046448,
        "Height": 0.36125001311302185
    },
    "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
    "ExternalImageId": "image9.jpg",
    "Confidence": 99.99949645996094,
    "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
},
{
    "BoundingBox": {
        "Width": 0.18562500178813934,
        "Top": 0.1618019938468933,
        "Left": 0.5575000047683716,
        "Height": 0.24770599603652954
    },
    "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
    "ExternalImageId": "image10.jpg",
    "Confidence": 99.99340057373047,
    "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
}
]
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[列出集合中的人脸](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFaces](#)中的。

## list-stream-processors

以下代码示例演示了如何使用 list-stream-processors。

### AWS CLI

列出您账户中的流媒体处理器

以下list-stream-processors命令列出了您账户中的流处理器以及每个流处理器的状态。

```
aws rekognition list-stream-processors
```

输出：

```
{
  "StreamProcessors": [
    {
      "Status": "STOPPED",
      "Name": "my-stream-processor"
    }
  ]
}
```

有关更多信息，[请参阅亚马逊 Rekognition 开发者指南中的使用流媒体视频](#)。

- 有关 API 的详细信息，[请参阅AWS CLI 命令参考ListStreamProcessors](#)中的。

## recognize-celebrities

以下代码示例演示了如何使用 recognize-celebrities。

有关更多信息，[请参阅识别图像中的名人](#)。

### AWS CLI

识别图像中的名人

以下 recognize-celebrities 命令将识别存储在 Amazon S3 存储桶中的指定图像中的名人。

```
aws rekognition recognize-celebrities \
  --image "S3Object={Bucket=MyImageS3Bucket,Name=moviestars.jpg}"
```

输出：

```
{
  "UnrecognizedFaces": [
    {
      "BoundingBox": {
        "Width": 0.14416666328907013,
        "Top": 0.077777778059244156,
        "Left": 0.625,
        "Height": 0.2746031880378723
      },
      "Confidence": 99.9990234375,
      "Pose": {
        "Yaw": 10.80408763885498,
        "Roll": -12.761146545410156,

```

```
    "Pitch": 10.96889877319336
  },
  "Quality": {
    "Sharpness": 94.1185531616211,
    "Brightness": 79.18367004394531
  },
  "Landmarks": [
    {
      "Y": 0.18220913410186768,
      "X": 0.6702951788902283,
      "Type": "eyeLeft"
    },
    {
      "Y": 0.16337193548679352,
      "X": 0.7188183665275574,
      "Type": "eyeRight"
    },
    {
      "Y": 0.20739148557186127,
      "X": 0.7055801749229431,
      "Type": "nose"
    },
    {
      "Y": 0.2889308035373688,
      "X": 0.687512218952179,
      "Type": "mouthLeft"
    },
    {
      "Y": 0.2706988751888275,
      "X": 0.7250053286552429,
      "Type": "mouthRight"
    }
  ]
},
"CelebrityFaces": [
  {
    "MatchConfidence": 100.0,
    "Face": {
      "BoundingBox": {
        "Width": 0.14000000059604645,
        "Top": 0.1190476194024086,
        "Left": 0.82833331823349,
        "Height": 0.2666666805744171
      }
    }
  }
]
```

```
    },
    "Confidence": 99.99359130859375,
    "Pose": {
      "Yaw": -10.509642601013184,
      "Roll": -14.51749324798584,
      "Pitch": 13.799399375915527
    },
    "Quality": {
      "Sharpness": 78.74752044677734,
      "Brightness": 42.201324462890625
    },
    "Landmarks": [
      {
        "Y": 0.2290833294391632,
        "X": 0.8709492087364197,
        "Type": "eyeLeft"
      },
      {
        "Y": 0.20639978349208832,
        "X": 0.9153988361358643,
        "Type": "eyeRight"
      },
      {
        "Y": 0.25417643785476685,
        "X": 0.8907724022865295,
        "Type": "nose"
      },
      {
        "Y": 0.32729196548461914,
        "X": 0.8876466155052185,
        "Type": "mouthLeft"
      },
      {
        "Y": 0.3115464746952057,
        "X": 0.9238573312759399,
        "Type": "mouthRight"
      }
    ]
  },
  "Name": "Celeb A",
  "Urls": [
    "www.imdb.com/name/aaaaaaaa"
  ],
  "Id": "1111111"
```

```
  },
  {
    "MatchConfidence": 97.0,
    "Face": {
      "BoundingBox": {
        "Width": 0.13333334028720856,
        "Top": 0.24920634925365448,
        "Left": 0.4449999928474426,
        "Height": 0.2539682686328888
      },
      "Confidence": 99.99979400634766,
      "Pose": {
        "Yaw": 6.557040691375732,
        "Roll": -7.316643714904785,
        "Pitch": 9.272967338562012
      },
      "Quality": {
        "Sharpness": 83.23492431640625,
        "Brightness": 78.83267974853516
      },
      "Landmarks": [
        {
          "Y": 0.3625510632991791,
          "X": 0.48898839950561523,
          "Type": "eyeLeft"
        },
        {
          "Y": 0.35366007685661316,
          "X": 0.5313721299171448,
          "Type": "eyeRight"
        },
        {
          "Y": 0.3894785940647125,
          "X": 0.5173314809799194,
          "Type": "nose"
        },
        {
          "Y": 0.44889405369758606,
          "X": 0.5020005702972412,
          "Type": "mouthLeft"
        },
        {
          "Y": 0.4408611059188843,
          "X": 0.5351271629333496,
```

```
        "Type": "mouthRight"
      }
    ]
  },
  "Name": "Celeb B",
  "Urls": [
    "www.imdb.com/name/bbbbbbbbbb"
  ],
  "Id": "2222222"
},
{
  "MatchConfidence": 100.0,
  "Face": {
    "BoundingBox": {
      "Width": 0.12416666746139526,
      "Top": 0.2968254089355469,
      "Left": 0.2150000035762787,
      "Height": 0.23650793731212616
    },
    "Confidence": 99.99958801269531,
    "Pose": {
      "Yaw": 7.801797866821289,
      "Roll": -8.326810836791992,
      "Pitch": 7.844768047332764
    },
    "Quality": {
      "Sharpness": 86.93206024169922,
      "Brightness": 79.81291198730469
    },
    "Landmarks": [
      {
        "Y": 0.4027804136276245,
        "X": 0.2575301229953766,
        "Type": "eyeLeft"
      },
      {
        "Y": 0.3934555947780609,
        "X": 0.2956969439983368,
        "Type": "eyeRight"
      },
      {
        "Y": 0.4309830069541931,
        "X": 0.2837020754814148,
        "Type": "nose"
      }
    ]
  }
}
```



```
    },
    {
      "Y": 0.48186683654785156,
      "X": 0.26812544465065,
      "Type": "mouthLeft"
    },
    {
      "Y": 0.47338807582855225,
      "X": 0.29905644059181213,
      "Type": "mouthRight"
    }
  ]
},
"Name": "Celeb C",
"Urls": [
  "www.imdb.com/name/ccccccccc"
],
"Id": "3333333"
},
{
  "MatchConfidence": 97.0,
  "Face": {
    "BoundingBox": {
      "Width": 0.11916666477918625,
      "Top": 0.3698412775993347,
      "Left": 0.008333333767950535,
      "Height": 0.22698412835597992
    },
    "Confidence": 99.99999237060547,
    "Pose": {
      "Yaw": 16.38478660583496,
      "Roll": -1.0260354280471802,
      "Pitch": 5.975185394287109
    },
    "Quality": {
      "Sharpness": 83.23492431640625,
      "Brightness": 61.408443450927734
    },
    "Landmarks": [
      {
        "Y": 0.4632347822189331,
        "X": 0.049406956881284714,
        "Type": "eyeLeft"
      },

```

```
    {
      "Y": 0.46388113498687744,
      "X": 0.08722897619009018,
      "Type": "eyeRight"
    },
    {
      "Y": 0.5020678639411926,
      "X": 0.0758260041475296,
      "Type": "nose"
    },
    {
      "Y": 0.544157862663269,
      "X": 0.054029736667871475,
      "Type": "mouthLeft"
    },
    {
      "Y": 0.5463630557060242,
      "X": 0.08464983850717545,
      "Type": "mouthRight"
    }
  ]
},
"Name": "Celeb D",
"Urls": [
  "www.imdb.com/name/ddddddddd"
],
"Id": "44444444"
}
]
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[识别图像中的名人](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RecognizeCelebrities](#) 中的。

## search-faces-by-image

以下代码示例演示了如何使用 search-faces-by-image。

有关更多信息，请参阅[搜索人脸 \(图像\)](#)。

### AWS CLI

搜索集合中与图像中最大人脸匹配的人脸。

以下 `search-faces-by-image` 命令将搜索集中与指定图像中最大人脸相匹配的人脸。

```
aws rekognition search-faces-by-image \  
  --image '{"S3Object":{"Bucket":"MyImageS3Bucket","Name":"ExamplePerson.jpg"}}' \  
  --collection-id MyFaceImageCollection  
  
{  
  "SearchedFaceBoundingBox": {  
    "Width": 0.18562500178813934,  
    "Top": 0.1618015021085739,  
    "Left": 0.5575000047683716,  
    "Height": 0.24770642817020416  
  },  
  "SearchedFaceConfidence": 99.993408203125,  
  "FaceMatches": [  
    {  
      "Face": {  
        "BoundingBox": {  
          "Width": 0.18562500178813934,  
          "Top": 0.1618019938468933,  
          "Left": 0.5575000047683716,  
          "Height": 0.24770599603652954  
        },  
        "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",  
        "ExternalImageId": "example-image.jpg",  
        "Confidence": 99.99340057373047,  
        "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"  
      },  
      "Similarity": 99.97913360595703  
    },  
    {  
      "Face": {  
        "BoundingBox": {  
          "Width": 0.18562500178813934,  
          "Top": 0.1618019938468933,  
          "Left": 0.5575000047683716,  
          "Height": 0.24770599603652954  
        },  
        "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",  
        "ExternalImageId": "image3.jpg",  
        "Confidence": 99.99340057373047,  
        "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"  
      },  
      "Similarity": 99.97913360595703  
    }  
  ]  
}
```

```
  },
  {
    "Face": {
      "BoundingBox": {
        "Width": 0.41499999165534973,
        "Top": 0.09187500178813934,
        "Left": 0.28083300590515137,
        "Height": 0.3112500011920929
      },
      "FaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
      "ExternalImageId": "image2.jpg",
      "Confidence": 99.99769592285156,
      "ImageId": "a294da46-2cb1-5cc4-9045-61d7ca567662"
    },
    "Similarity": 99.18069458007812
  },
  {
    "Face": {
      "BoundingBox": {
        "Width": 0.48166701197624207,
        "Top": 0.20999999344348907,
        "Left": 0.21250000596046448,
        "Height": 0.36125001311302185
      },
      "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
      "ExternalImageId": "image1.jpg",
      "Confidence": 99.99949645996094,
      "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
    },
    "Similarity": 98.66607666015625
  },
  {
    "Face": {
      "BoundingBox": {
        "Width": 0.5349419713020325,
        "Top": 0.29124999046325684,
        "Left": 0.16389399766921997,
        "Height": 0.40187498927116394
      },
      "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
      "ExternalImageId": "image9.jpg",
      "Confidence": 99.99979400634766,
      "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
    },
  },
```

```
    "Similarity": 98.24278259277344
  },
  {
    "Face": {
      "BoundingBox": {
        "Width": 0.5307819843292236,
        "Top": 0.2862499952316284,
        "Left": 0.1564060002565384,
        "Height": 0.3987500071525574
      },
      "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
      "ExternalImageId": "image10.jpg",
      "Confidence": 99.99970245361328,
      "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
    },
    "Similarity": 98.10665893554688
  },
  {
    "Face": {
      "BoundingBox": {
        "Width": 0.5074880123138428,
        "Top": 0.3774999976158142,
        "Left": 0.18302799761295319,
        "Height": 0.3812499940395355
      },
      "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
      "ExternalImageId": "image6.jpg",
      "Confidence": 99.99930572509766,
      "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
    },
    "Similarity": 98.10526275634766
  },
  {
    "Face": {
      "BoundingBox": {
        "Width": 0.5574039816856384,
        "Top": 0.37187498807907104,
        "Left": 0.14559100568294525,
        "Height": 0.4181250035762787
      },
      "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
      "ExternalImageId": "image5.jpg",
      "Confidence": 99.99960327148438,
      "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
```

```
    },
    "Similarity": 97.94659423828125
  },
  {
    "Face": {
      "BoundingBox": {
        "Width": 0.5773710012435913,
        "Top": 0.34437501430511475,
        "Left": 0.12396000325679779,
        "Height": 0.4337500035762787
      },
      "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
      "ExternalImageId": "image8.jpg",
      "Confidence": 100.0,
      "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
    },
    "Similarity": 97.93476867675781
  }
],
"FaceModelVersion": "3.0"
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[使用图像搜索人脸](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SearchFacesByImage](#) 中的。

## search-faces

以下代码示例演示了如何使用 search-faces。

有关更多信息，请参阅[搜索人脸 \(面容 ID\)](#)。

### AWS CLI

搜索集合中与人脸 ID 匹配的人脸

以下 search-faces 命令将搜索集合中与指定人脸 ID 相匹配的人脸。

```
aws rekognition search-faces \
  --face-id 8d3cfc70-4ba8-4b36-9644-90fba29c2dac \
  --collection-id MyCollection
```

输出：

```
{
  "SearchedFaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
  "FaceModelVersion": "3.0",
  "FaceMatches": [
    {
      "Face": {
        "BoundingBox": {
          "Width": 0.48166701197624207,
          "Top": 0.20999999344348907,
          "Left": 0.21250000596046448,
          "Height": 0.36125001311302185
        },
        "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
        "ExternalImageId": "image1.jpg",
        "Confidence": 99.99949645996094,
        "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
      },
      "Similarity": 99.30997467041016
    },
    {
      "Face": {
        "BoundingBox": {
          "Width": 0.18562500178813934,
          "Top": 0.1618019938468933,
          "Left": 0.5575000047683716,
          "Height": 0.24770599603652954
        },
        "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
        "ExternalImageId": "example-image.jpg",
        "Confidence": 99.99340057373047,
        "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
      },
      "Similarity": 99.24862670898438
    },
    {
      "Face": {
        "BoundingBox": {
          "Width": 0.18562500178813934,
          "Top": 0.1618019938468933,
          "Left": 0.5575000047683716,
          "Height": 0.24770599603652954
        },
        "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
```

```
        "ExternalImageId": "image3.jpg",
        "Confidence": 99.99340057373047,
        "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
    },
    "Similarity": 99.24862670898438
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.5349419713020325,
            "Top": 0.29124999046325684,
            "Left": 0.16389399766921997,
            "Height": 0.40187498927116394
        },
        "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
        "ExternalImageId": "image9.jpg",
        "Confidence": 99.99979400634766,
        "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
    },
    "Similarity": 96.73158264160156
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.5307819843292236,
            "Top": 0.2862499952316284,
            "Left": 0.1564060002565384,
            "Height": 0.3987500071525574
        },
        "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
        "ExternalImageId": "image10.jpg",
        "Confidence": 99.99970245361328,
        "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
    },
    "Similarity": 96.48291015625
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.5074880123138428,
            "Top": 0.3774999976158142,
            "Left": 0.18302799761295319,
            "Height": 0.3812499940395355
        },
```



```
        "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
        "ExternalImageId": "image6.jpg",
        "Confidence": 99.99930572509766,
        "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
    },
    "Similarity": 96.43287658691406
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.5574039816856384,
            "Top": 0.37187498807907104,
            "Left": 0.14559100568294525,
            "Height": 0.4181250035762787
        },
        "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
        "ExternalImageId": "image5.jpg",
        "Confidence": 99.99960327148438,
        "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
    },
    "Similarity": 95.25305938720703
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.5773710012435913,
            "Top": 0.34437501430511475,
            "Left": 0.12396000325679779,
            "Height": 0.4337500035762787
        },
        "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
        "ExternalImageId": "image8.jpg",
        "Confidence": 100.0,
        "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
    },
    "Similarity": 95.22837829589844
}
]
}
```

有关更多信息，请参阅《Amazon Rekognition 开发人员指南》中的[使用人脸 ID 搜索人脸](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SearchFaces](#)中的。

## start-celebrity-recognition

以下代码示例演示了如何使用 start-celebrity-recognition。

### AWS CLI

开始在存储的视频中识别名人

以下start-celebrity-recognition命令启动一项任务，在存储在 Amazon S3 存储桶中的指定视频文件中寻找名人。

```
aws rekognition start-celebrity-recognition \  
  --video "S3Object={Bucket=MyVideoS3Bucket,Name=MyVideoFile.mpg}"
```

输出：

```
{  
  "JobId": "1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef"  
}
```

有关更多信息，请参阅亚马逊 Rekognition 开发者指南中的识别[存储视频中的名人](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartCelebrityRecognition](#)中的。

## start-content-moderation

以下代码示例演示了如何使用 start-content-moderation。

### AWS CLI

开始识别存储视频中的不安全内容

以下start-content-moderation命令启动一项任务，以检测存储在 Amazon S3 存储桶中的指定视频文件中的不安全内容。

```
aws rekognition start-content-moderation \  
  --video "S3Object={Bucket=MyVideoS3Bucket,Name=MyVideoFile.mpg}"
```

输出：

```
{
```

```
"JobId": "1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef"
}
```

有关更多信息，请参阅《亚马逊 Rekognition 开发者指南》中的“[检测存储的不安全视频](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartContentModeration](#)中的。

## start-face-detection

以下代码示例演示了如何使用 start-face-detection。

### AWS CLI

检测视频中的人脸

以下start-face-detection命令启动一项任务，以检测存储在 Amazon S3 存储桶中的指定视频文件中的人脸。

```
aws rekognition start-face-detection
  --video "S3Object={Bucket=MyVideoS3Bucket,Name=MyVideoFile.mpg}"
```

输出：

```
{
  "JobId": "1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef"
}
```

有关更多信息，请参阅亚马逊 Rekognition 开发者指南中的[检测存储视频中的人脸](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartFaceDetection](#)中的。

## start-face-search

以下代码示例演示了如何使用 start-face-search。

### AWS CLI

在集合中搜索与视频中检测到的人脸相匹配的人脸

以下start-face-search命令启动一项任务，在集合中搜索与 Amazon S3 存储桶中指定视频文件中检测到的人脸相匹配的人脸。

```
aws rekognition start-face-search \  
  --video "S3Object={Bucket=MyVideoS3Bucket,Name=MyVideoFile.mpg}" \  
  --collection-id collection
```

输出：

```
{  
  "JobId": "1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef"  
}
```

有关更多信息，请参阅亚马逊 Rekognition [开发者指南中的在存储的视频中搜索人脸](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartFaceSearch](#)中的。

## start-label-detection

以下代码示例演示了如何使用 start-label-detection。

### AWS CLI

检测视频中的物体和场景

以下start-label-detection命令启动一项任务，以检测存储在 Amazon S3 存储桶中的指定视频文件中的对象和场景。

```
aws rekognition start-label-detection \  
  --video "S3Object={Bucket=MyVideoS3Bucket,Name=MyVideoFile.mpg}"
```

输出：

```
{  
  "JobId": "1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef"  
}
```

有关更多信息，请参阅亚马逊 Rekognition 开发者指南[中的检测视频中的标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartLabelDetection](#)中的。

## start-person-tracking

以下代码示例演示了如何使用 start-person-tracking。

## AWS CLI

在存储的视频中开始人物路径

以下 `start-person-tracking` 命令启动一项任务，以跟踪人们在 Amazon S3 存储桶中存储的指定视频文件中的路径。：

```
aws rekognition start-person-tracking \  
  --video "S3Object={Bucket=MyVideoS3Bucket,Name=MyVideoFile.mpg}"
```

输出：

```
{  
  "JobId": "1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef"  
}
```

有关更多信息，请参阅亚马逊 Rekognition 开发者指南中的 [人员路径](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartPersonTracking](#) 中的。

## start-stream-processor

以下代码示例演示了如何使用 `start-stream-processor`。

### AWS CLI

启动流处理器

以下 `start-stream-processor` 命令启动指定的视频流处理器。

```
aws rekognition start-stream-processor \  
  --name my-stream-processor
```

此命令不生成任何输出。

有关更多信息，[请参阅亚马逊 Rekognition 开发者指南中的使用流媒体视频](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartStreamProcessor](#) 中的。

## stop-stream-processor

以下代码示例演示了如何使用 `stop-stream-processor`。

## AWS CLI

### 停止正在运行的流处理器

以下 `stop-stream-processor` 命令停止指定的正在运行的流处理器。

```
aws rekognition stop-stream-processor \  
  --name my-stream-processor
```

此命令不生成任何输出。

有关更多信息，[请参阅亚马逊 Rekognition 开发者指南中的使用流媒体视频](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StopStreamProcessor](#) 中的。

## AWS RAM 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS RAM。AWS Command Line Interface

**操作** 是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

**场景** 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **accept-resource-share-invitation**

以下代码示例演示了如何使用 `accept-resource-share-invitation`。

## AWS CLI

### 接受资源共享邀请

以下`accept-resource-share-invitation`示例接受指定的资源共享邀请。受邀账户中的委托人可以立即开始使用共享中的资源。

```
aws ram accept-resource-share-invitation \  
  --resource-share-invitation-arn arn:aws:ram:us-west-2:111111111111:resource-  
share-invitation/1e3477be-4a95-46b4-bbe0-c4001EXAMPLE
```

输出：

```
{  
  "resourceShareInvitation": {  
    "resourceShareInvitationArn": "arn:aws:ram:us-west-2:111111111111:resource-  
share-invitation/1e3477be-4a95-46b4-bbe0-c4001EXAMPLE",  
    "resourceShareName": "MyLicenseShare",  
    "resourceShareArn": "arn:aws:ram:us-west-2:111111111111:resource-  
share/27d09b4b-5e12-41d1-a4f2-19dedEXAMPLE",  
    "senderAccountId": "111111111111",  
    "receiverAccountId": "222222222222",  
    "invitationTimestamp": "2021-09-22T15:07:35.620000-07:00",  
    "status": "ACCEPTED"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AcceptResourceShareInvitation](#)中的。

## associate-resource-share-permission

以下代码示例演示了如何使用 `associate-resource-share-permission`。

### AWS CLI

将 RAM 托管权限与资源共享关联

以下`associate-resource-share-permission`示例将相关资源类型的现有托管权限替换为指定的托管权限。对相关资源类型的所有资源的访问权限受新权限的约束。

```
aws ram associate-resource-share-permission \  
  --permission-arn arn:aws:ram::aws:permission/  
AWSRAMPermissionGlueDatabaseReadWrite \  
  --replace \  

```

```
--resource-share-arn arn:aws:ram:us-west-2:123456789012:resource-
share/27d09b4b-5e12-41d1-a4f2-19dedEXAMPLE
```

输出：

```
{
  "returnValue": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateResourceSharePermission](#)中的。

## associate-resource-share

以下代码示例演示了如何使用 associate-resource-share。

### AWS CLI

示例 1：将资源与资源共享关联

以下associate-resource-share示例将许可证配置添加到指定的资源共享。

```
aws ram associate-resource-share \
  --resource-share arn:aws:ram:us-west-2:123456789012:resource-
share/27d09b4b-5e12-41d1-a4f2-19dedEXAMPLE \
  --resource-arns arn:aws:license-manager:us-west-2:123456789012:license-
configuration:lic-36be0485f5ae379cc74cf8e92EXAMPLE
```

输出：

```
{
  "resourceShareAssociations": [
    {
      "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-
share/27d09b4b-5e12-41d1-a4f2-19dedEXAMPLE",
      "associatedEntity": "arn:aws:license-manager:us-
west-2:123456789012:license-configuration:lic-36be0485f5ae379cc74cf8e92EXAMPLE",
      "associationType": "RESOURCE",
      "status": "ASSOCIATING",
      "external": false
    }
  ]
}
```



```
}
```

## 示例 2：将委托人与资源共享关联

以下 `associate-resource-share` 示例向指定组织单位中的所有账户授予对指定资源共享的访问权限。

```
aws ram associate-resource-share \  
  --resource-share-arn arn:aws:ram:us-west-2:123456789012:resource-  
share/27d09b4b-5e12-41d1-a4f2-19dedEXAMPLE \  
  --principals arn:aws:organizations::123456789012:ou/o-63bEXAMPLE/ou-46xi-  
rEXAMPLE
```

输出：

```
{  
  "resourceShareAssociations": [  
    {  
      "status": "ASSOCIATING",  
      "associationType": "PRINCIPAL",  
      "associatedEntity": "arn:aws:organizations::123456789012:ou/  
o-63bEXAMPLE/ou-46xi-rEXAMPLE",  
      "external": false,  
      "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-  
share/27d09b4b-5e12-41d1-a4f2-19dedEXAMPLE"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateResourceShare](#) 中的。

## create-resource-share

以下代码示例演示了如何使用 `create-resource-share`。

### AWS CLI

#### 示例 1：创建资源共享

以下 `create-resource-share` 示例创建了一个具有指定名称的空资源共享。您必须分别向共享添加资源、委托人和权限。

```
aws ram create-resource-share \  
  --name MyNewResourceShare
```

输出：

```
{  
  "resourceShare": {  
    "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-  
share/4476c27d-8feb-4b21-afe9-7de23EXAMPLE",  
    "name": "MyNewResourceShare",  
    "owningAccountId": "123456789012",  
    "allowExternalPrincipals": true,  
    "status": "ACTIVE",  
    "creationTime": 1634586271.302,  
    "lastUpdatedTime": 1634586271.302  
  }  
}
```

示例 2：创建以 AWS 账户为委托人的资源共享

以下 `create-resource-share` 示例创建了一个资源共享并向指定 AWS 账户 (222222222222) 授予访问权限。如果指定的委托人不属于同一个 AWS 组织，则会发送邀请，并且在授予访问权限之前必须接受邀请。

```
aws ram create-resource-share \  
  --name MyNewResourceShare \  
  --principals 222222222222
```

示例 3：创建仅限于您的 AWS 组织的资源共享

以下 `create-resource-share` 示例创建一个资源共享，该共享仅限于您的账户所属的 AWS 组织中的账户，并将指定的 OU 添加为委托人。该 OU 中的所有账户都可以使用资源共享中的资源。

```
aws ram create-resource-share \  
  --name MyNewResourceShare \  
  --no-allow-external-principals \  
  --principals arn:aws:organizations::123456789012:ou/o-63bEXAMPLE/ou-46xi-  
rEXAMPLE
```

输出：

```
{
  "resourceShare": {
    "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-
share/7be8694e-095c-41ca-9ce8-7be4aEXAMPLE",
    "name": "MyNewResourceShare",
    "owningAccountId": "123456789012",
    "allowExternalPrincipals": false,
    "status": "ACTIVE",
    "creationTime": 1634587042.49,
    "lastUpdatedTime": 1634587042.49
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateResourceShare](#)中的。

## delete-resource-share

以下代码示例演示了如何使用 delete-resource-share。

### AWS CLI

#### 删除资源共享

以下delete-resource-share示例删除了指定的资源共享。

```
aws ram delete-resource-share \
  --resource-share-arn arn:aws:ram:us-west-2:123456789012:resource-share/7ab63972-
b505-7e2a-420d-6f5d3EXAMPLE
```

以下输出表示成功：

```
{
  "returnValue": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteResourceShare](#)中的。

## disassociate-resource-share-permission

以下代码示例演示了如何使用 disassociate-resource-share-permission。

## AWS CLI

从资源共享中移除资源类型的 RAM 托管权限

以下`disassociate-resource-share-permission`示例从指定的资源共享中移除 Glue 数据库的 RAM 托管权限。

```
aws ram disassociate-resource-share-permission \  
  --resource-share-arn arn:aws:ram:us-west-2:123456789012:resource-  
share/27d09b4b-5e12-41d1-a4f2-19dedEXAMPLE \  
  --permission-arn arn:aws:ram::aws:permission/  
AWSRAMPermissionGlueDatabaseReadWrite
```

输出：

```
{  
  "returnValue": true  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateResourceSharePermission](#)中的。

## `disassociate-resource-share`

以下代码示例演示了如何使用 `disassociate-resource-share`。

## AWS CLI

从资源共享中移除资源

以下`disassociate-resource-share`示例从指定的资源共享中移除指定资源（在本例中为 VPC 子网）。任何有权访问资源共享的委托人都无法再对该资源执行操作。

```
aws ram disassociate-resource-share \  
  --resource-arns arn:aws:ec2:us-west-2:123456789012:subnet/  
subnet-0250c25a1fEXAMPLE \  
  --resource-share-arn arn:aws:ram:us-west-2:123456789012:resource-share/7ab63972-  
b505-7e2a-420d-6f5d3EXAMPLE
```

输出：

```
{
```

```
"resourceShareAssociations": [  
  "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-  
share/7ab63972-b505-7e2a-420d-6f5d3EXAMPLE",  
  "associatedEntity": "arn:aws:ec2:us-west-2:123456789012:subnet/  
subnet-0250c25a1fEXAMPLE",  
  "associationType": "RESOURCE",  
  "status": "DISASSOCIATING",  
  "external": false  
]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateResourceShare](#)中的。

## enable-sharing-with-aws-organization

以下代码示例演示了如何使用 enable-sharing-with-aws-organization。

### AWS CLI

启用跨 AWS 组织共享资源

以下enable-sharing-with-aws-organization示例启用了跨组织和组织单位的资源共享。

```
aws ram enable-sharing-with-aws-organization
```

以下输出代表成功。

```
{  
  "returnValue": true  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableSharingWithAwsOrganization](#)中的。

## get-permission

以下代码示例演示了如何使用 get-permission。

### AWS CLI

检索 RAM 托管权限的详细信息

以下get-permission示例显示了指定 RAM 托管权限的默认版本的详细信息。

```
aws ram get-permission \  
  --permission-arn arn:aws:ram::aws:permission/  
  AWSRAMPermissionGlueTableReadWriteForDatabase
```

输出：

```
{  
  "permission": {  
    "arn": "arn:aws:ram::aws:permission/  
AWSRAMPermissionGlueTableReadWriteForDatabase",  
    "version": "2",  
    "defaultVersion": true,  
    "name": "AWSRAMPermissionGlueTableReadWriteForDatabase",  
    "resourceType": "glue:Database",  
    "permission": "[\n\"Effect\":\n\"Allow\",  
\"Action\":[\n\"glue:GetTable\n\",  
\"glue:UpdateTable\n\",  
\"glue>DeleteTable\n\",  
\"glue:BatchDeleteTable\n\",  
\"glue:BatchDeleteTableVersion\n\",  
\"glue:GetTableVersion\n\",  
\"glue:GetTableVersions\n\",  
\"glue:GetPartition\n\",  
\"glue:GetPartitions\n\",  
\"glue:BatchGetPartition\n\",  
\"glue:BatchCreatePartition\n\",  
\"glue>CreatePartition\n\",  
\"glue:UpdatePartition\n\",  
\"glue:BatchDeletePartition\n\",  
\"glue>DeletePartition\n\",  
\"glue:GetTables\n\",  
\"glue:SearchTables\n\"]]",  
    "creationTime": 1624912434.431,  
    "lastUpdatedTime": 1624912434.431,  
    "isResourceTypeDefault": false  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPermission](#)中的。

## get-resource-policies

以下代码示例演示了如何使用 get-resource-policies。

AWS CLI

获取资源的策略

以下get-resource-policies示例显示了与资源共享关联的指定资源的基于资源的权限策略。

```
aws ram get-resource-policies \  
  --resource-arns arn:aws:ec2:us-west-2:123456789012:subnet/  
  subnet-0250c25a1fEXAMPLE
```

输出：

```
{
  "policies": [
    {
      "Version": "2008-10-17",
      "Statement": [
        {
          "Sid": "RamStatement1",
          "Effect": "Allow",
          "Principal": {
            "AWS": []
          },
          "Action": [
            "ec2:RunInstances",
            "ec2:CreateNetworkInterface",
            "ec2:DescribeSubnets"
          ],
          "Resource": [
            "arn:aws:ec2:us-west-2:123456789012:subnet/subnet-0250c25a1fEXAMPLE"
          ]
        }
      ]
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetResourcePolicies](#)中的。

## get-resource-share-associations

以下代码示例演示了如何使用 `get-resource-share-associations`。

### AWS CLI

示例 1：列出所有资源类型的所有资源关联

以下`get-resource-share-associations`示例列出了所有资源共享中所有资源类型的资源关联。

```
aws ram get-resource-share-associations \
  --association-type RESOURCE
```

输出：

```
{
  "resourceShareAssociations": [
    {
      "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-share/7ab63972-b505-7e2a-420d-6f5d3EXAMPLE",
      "associatedEntity": "arn:aws:ec2:us-west-2:123456789012:subnet/subnet-0250c25a1fEXAMPLE",
      "resourceShareName": "MySubnetShare",
      "associationType": "RESOURCE",
      "status": "ASSOCIATED",
      "creationTime": 1565303590.973,
      "lastUpdatedTime": 1565303591.695,
      "external": false
    }
  ]
}
```

```

    },
    {
      "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-
share/8167bdfe-4480-4a01-8632-315e0EXAMPLE",
      "associatedEntity": "arn:aws:license-manager:us-
west-2:123456789012:license-configuration:lic-36be0485f5ae379cc74cf8e92EXAMPLE",
      "resourceShareName": "MyLicenseShare",
      "associationType": "RESOURCE",
      "status": "ASSOCIATED",
      "creationTime": 1632342958.457,
      "lastUpdatedTime": 1632342958.907,
      "external": false
    }
  ]
}

```

## 示例 2：列出资源共享的主体关联

以下 `get-resource-share-associations` 示例仅列出指定资源共享的委托人关联。

```

aws ram get-resource-share-associations \
  --resource-share-arns arn:aws:ram:us-west-2:123456789012:resource-
share/7be8694e-095c-41ca-9ce8-7be4aEXAMPLE \
  --association-type PRINCIPAL

```

输出：

```

{
  "resourceShareAssociations": [
    {
      "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-
share/7be8694e-095c-41ca-9ce8-7be4aEXAMPLE",
      "resourceShareName": "MyNewResourceShare",
      "associatedEntity": "arn:aws:organizations::123456789012:ou/
o-63bEXAMPLE/ou-46xi-rEXAMPLE",
      "associationType": "PRINCIPAL",
      "status": "ASSOCIATED",
      "creationTime": 1634587042.49,
      "lastUpdatedTime": 1634587044.291,
      "external": false
    }
  ]
}

```



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetResourceShareAssociations](#)中的。

## get-resource-share-invitations

以下代码示例演示了如何使用 `get-resource-share-invitations`。

### AWS CLI

列出您的资源共享邀请

以下`get-resource-share-invitations`示例列出了您当前的资源共享邀请。

```
aws ram get-resource-share-invitations
```

输出：

```
{
  "resourceShareInvitations": [
    {
      "resourceShareInvitationArn": "arn:aws:ram:us-west-2-1:111111111111:resource-share-invitation/32b639f0-14b8-7e8f-55ea-e6117EXAMPLE",
      "resourceShareName": "project-resource-share",
      "resourceShareArn": "arn:aws:ram:us-west-2:111111111111:resource-share/fcb639f0-1449-4744-35bc-a983fEXAMPLE",
      "senderAccountId": "111111111111",
      "receiverAccountId": "222222222222",
      "invitationTimestamp": 1565312166.258,
      "status": "PENDING"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetResourceShareInvitations](#)中的。

## get-resource-shares

以下代码示例演示了如何使用 `get-resource-shares`。

### AWS CLI

示例 1：列出您拥有并与其他人共享的资源共享

以下`get-resource-shares`示例列出了创建并正在与其他人共享的资源共享。

```
aws ram get-resource-shares \  
  --resource-owner SELF
```

输出：

```
{  
  "resourceShares": [  
    {  
      "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-  
share/3ab63985-99d9-1cd2-7d24-75e93EXAMPLE",  
      "name": "my-resource-share",  
      "owningAccountId": "123456789012",  
      "allowExternalPrincipals": false,  
      "status": "ACTIVE",  
      "tags": [  
        {  
          "key": "project",  
          "value": "lima"  
        }  
      ]  
      "creationTime": 1565295733.282,  
      "lastUpdatedTime": 1565295733.282  
    },  
    {  
      "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-  
share/7ab63972-b505-7e2a-420d-6f5d3EXAMPLE",  
      "name": "my-resource-share",  
      "owningAccountId": "123456789012",  
      "allowExternalPrincipals": true,  
      "status": "ACTIVE",  
      "creationTime": 1565295733.282,  
      "lastUpdatedTime": 1565295733.282  
    }  
  ]  
}
```

示例 2：列出他人拥有并与您共享的资源共享

以下`get-resource-shares`示例列出了其他人创建并与您共享的资源共享。在此示例中，没有。

```
aws ram get-resource-shares \  
  --resource-owner SELF
```

```
--resource-owner OTHER-ACCOUNTS
```

输出：

```
{
  "resourceShares": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetResourceShares](#)中的。

## list-pending-invitation-resources

以下代码示例演示了如何使用 `list-pending-invitation-resources`。

AWS CLI

列出待处理资源共享中可用的资源

以下`list-pending-invitation-resources`示例列出了与指定邀请关联的资源共享中的所有资源。

```
aws ram list-pending-invitation-resources \
  --resource-share-invitation-arn arn:aws:ram:us-west-2:123456789012:resource-
share-invitation/1e3477be-4a95-46b4-bbe0-c4001EXAMPLE
```

输出：

```
{
  "resources": [
    {
      "arn": "arn:aws:ec2:us-west-2:123456789012:subnet/
subnet-04a555b0e6EXAMPLE",
      "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-
share/7be8694e-095c-41ca-9ce8-7be4aEXAMPLE",
      "creationTime": 1634676051.269,
      "lastUpdatedTime": 1634676052.07,
      "status": "AVAILABLE",
      "type": "ec2:Subnet"
    },
    {
      "arn": "arn:aws:license-manager:us-west-2:123456789012:license-
configuration/lic-36be0485f5ae379cc74cf8e92EXAMPLE",
```

```
        "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-
share/7ab63972-b505-7e2a-420d-6f5d3EXAMPLE",
        "creationTime": 1624912434.431,
        "lastUpdatedTime": 1624912434.431,
        "status": "AVAILABLE",
        "type": "license-manager:LicenseConfiguration"
    }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPendingInvitationResources](#)中的。

## list-permissions

以下代码示例演示了如何使用 list-permissions。

### AWS CLI

列出可用的 RAM 托管权限

以下list-permissions示例列出了仅适用于 G AWS lue 数据库资源类型的所有 RAM 托管权限。

```
aws ram list-permissions \
    --resource-type glue:Database
```

输出：

```
{
  "permissions": [
    {
      "arn": "arn:aws:ram::aws:permission/
AWSRAMDefaultPermissionGlueDatabase",
      "version": "1",
      "defaultVersion": true,
      "name": "AWSRAMDefaultPermissionGlueDatabase",
      "resourceType": "glue:Database",
      "creationTime": 1592007820.935,
      "lastUpdatedTime": 1592007820.935,
      "isResourceTypeDefault": true
    },
    {
```

```

        "arn": "arn:aws:ram::aws:permission/
AWSRAMPermissionGlueAllTablesReadWriteForDatabase",
        "version": "2",
        "defaultVersion": true,
        "name": "AWSRAMPermissionGlueAllTablesReadWriteForDatabase",
        "resourceType": "glue:Database",
        "creationTime": 1624912413.323,
        "lastUpdatedTime": 1624912413.323,
        "isResourceTypeDefault": false
    },
    {
        "arn": "arn:aws:ram::aws:permission/
AWSRAMPermissionGlueDatabaseReadWrite",
        "version": "2",
        "defaultVersion": true,
        "name": "AWSRAMPermissionGlueDatabaseReadWrite",
        "resourceType": "glue:Database",
        "creationTime": 1624912417.4,
        "lastUpdatedTime": 1624912417.4,
        "isResourceTypeDefault": false
    },
    {
        "arn": "arn:aws:ram::aws:permission/
AWSRAMPermissionGlueTableReadWriteForDatabase",
        "version": "2",
        "defaultVersion": true,
        "name": "AWSRAMPermissionGlueTableReadWriteForDatabase",
        "resourceType": "glue:Database",
        "creationTime": 1624912434.431,
        "lastUpdatedTime": 1624912434.431,
        "isResourceTypeDefault": false
    }
]
}

```

以下list-permissions示例显示了所有资源类型的可用的 RAM 托管权限。

```
aws ram list-permissions
```

输出：

```
{
  "permissions": [
```

```

    {
      "arn": "arn:aws:ram::aws:permission/
AWSRAMBlankEndEntityCertificateAPICSRPassthroughIssuanceCertificateAuthority",
      "version": "1",
      "defaultVersion": true,
      "name":
"AWSRAMBlankEndEntityCertificateAPICSRPassthroughIssuanceCertificateAuthority",
      "resourceType": "acm-pca:CertificateAuthority",
      "creationTime": 1623264861.085,
      "lastUpdatedTime": 1623264861.085,
      "isResourceTypeDefault": false
    },
    {
      "arn": "arn:aws:ram::aws:permission/AWSRAMDefaultPermissionAppMesh",
      "version": "1",
      "defaultVersion": true,
      "name": "AWSRAMDefaultPermissionAppMesh",
      "resourceType": "appmesh:Mesh",
      "creationTime": 1589307188.584,
      "lastUpdatedTime": 1589307188.584,
      "isResourceTypeDefault": true
    },
    ...TRUNCATED FOR BREVITY...
    {
      "arn": "arn:aws:ram::aws:permission/
AWSRAMSubordinateCACertificatePathLen0IssuanceCertificateAuthority",
      "version": "1",
      "defaultVersion": true,
      "name":
"AWSRAMSubordinateCACertificatePathLen0IssuanceCertificateAuthority",
      "resourceType": "acm-pca:CertificateAuthority",
      "creationTime": 1623264876.75,
      "lastUpdatedTime": 1623264876.75,
      "isResourceTypeDefault": false
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPermissions](#)中的。

## list-principals

以下代码示例演示了如何使用 list-principals。

## AWS CLI

列出有权访问资源的委托人

以下`list-principals`示例显示了可以通过任何资源共享访问指定类型资源的委托人列表。

```
aws ram list-principals \  
  --resource-type ec2:Subnet
```

输出：

```
{  
  "principals": [  
    {  
      "id": "arn:aws:organizations::123456789012:ou/o-gx7EXAMPLE/ou-29c5-  
zEXAMPLE",  
      "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-  
share/7ab63972-b505-7e2a-420d-6f5d3EXAMPLE",  
      "creationTime": 1565298209.737,  
      "lastUpdatedTime": 1565298211.019,  
      "external": false  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPrincipals](#)中的。

## list-resource-share-permissions

以下代码示例演示了如何使用 `list-resource-share-permissions`。

### AWS CLI

列出当前附加到资源共享的所有 RAM 托管权限

以下`list-resource-share-permissions`示例列出了附加到指定资源共享的所有 RAM 托管权限。

```
aws ram list-resource-share-permissions \  
  --resource-share-arn arn:aws:ram:us-west-2:123456789012:resource-  
share/27d09b4b-5e12-41d1-a4f2-19dedEXAMPLE
```

输出：

```
{
  "permissions": [
    {
      "arn": "arn:aws:ram::aws:permission/
AWSRAMDefaultPermissionLicenseConfiguration",
      "version": "1",
      "resourceType": "license-manager:LicenseConfiguration",
      "status": "ASSOCIATED",
      "lastUpdatedTime": 1632342984.234
    },
    {
      "arn": "arn:aws:ram::aws:permission/
AWSRAMPermissionGlueDatabaseReadWrite",
      "version": "2",
      "resourceType": "glue:Database",
      "status": "ASSOCIATED",
      "lastUpdatedTime": 1632512462.297
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResourceSharePermissions](#)中的。

## list-resource-types

以下代码示例演示了如何使用 list-resource-types。

AWS CLI

列出 AWS RAM 支持的资源类型

以下list-resource-types示例列出了 AWS RAM 当前支持的所有资源类型。

```
aws ram list-resource-types
```

输出：

```
{
  "resourceTypes": [
    {
```



```

        "resourceType": "route53resolver:FirewallRuleGroup",
        "serviceName": "route53resolver"
    },
    {
        "resourceType": "ec2:LocalGatewayRouteTable",
        "serviceName": "ec2"
    },
    ...OUTPUT TRUNCATED FOR BREVITY...
    {
        "resourceType": "ec2:Subnet",
        "serviceName": "ec2"
    },
    {
        "resourceType": "ec2:TransitGatewayMulticastDomain",
        "serviceName": "ec2"
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResourceTypes](#)中的。

## list-resources

以下代码示例演示了如何使用 `list-resources`。

### AWS CLI

列出与资源共享关联的资源

以下`list-resources`示例列出了指定资源共享中属于指定资源类型的所有资源。

```

aws ram list-resources \
  --resource-type ec2:Subnet \
  --resource-owner SELF \
  --resource-share-arn arn:aws:ram:us-west-2:123456789012:resource-share/7ab63972-
b505-7e2a-420d-6f5d3EXAMPLE

```

输出：

```

{
  "resources": [
    {

```

```

        "arn": "aarn:aws:ec2:us-west-2:123456789012:subnet/
subnet-0250c25a1f4e15235",
        "type": "ec2:Subnet",
        "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-
share/7ab63972-b505-7e2a-420d-6f5d3EXAMPLE",
        "creationTime": 1565301545.023,
        "lastUpdatedTime": 1565301545.947
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResources](#)中的。

### **promote-resource-share-created-from-policy**

以下代码示例演示了如何使用 `promote-resource-share-created-from-policy`。

#### AWS CLI

在 RAM 中将基于资源策略的资源共享提升到全部功能 AWS

以下 `promote-resource-share-created-from-policy` 示例采用您通过附加基于资源的策略隐式创建的资源共享，并将其转换为 AWS RAM 控制台及其 CLI 和 API 操作的完整功能。

```

aws ram promote-resource-share-created-from-policy \
  --resource-share-arn arn:aws:ram:us-east-1:123456789012:resource-
share/91fa8429-2d06-4032-909a-90909EXAMPLE

```

输出：

```

{
  "returnValue": true
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PromoteResourceShareCreatedFromPolicy](#)中的。

### **reject-resource-share-invitation**

以下代码示例演示了如何使用 `reject-resource-share-invitation`。

## AWS CLI

### 拒绝资源共享邀请

以下`reject-resource-share-invitation`示例拒绝了指定的资源共享邀请。

```
aws ram reject-resource-share-invitation \  
  --resource-share-invitation-arn arn:aws:ram:us-west-2:111111111111:resource-  
share-invitation/32b639f0-14b8-7e8f-55ea-e6117EXAMPLE
```

输出：

```
"resourceShareInvitations": [  
  {  
    "resourceShareInvitationArn": "arn:aws:ram:us-west2-1:111111111111:resource-  
share-invitation/32b639f0-14b8-7e8f-55ea-e6117EXAMPLE",  
    "resourceShareName": "project-resource-share",  
    "resourceShareArn": "arn:aws:ram:us-west-2:111111111111:resource-share/  
fcb639f0-1449-4744-35bc-a983fEXAMPLE",  
    "senderAccountId": "111111111111",  
    "receiverAccountId": "222222222222",  
    "invitationTimestamp": 1565319592.463,  
    "status": "REJECTED"  
  }  
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RejectResourceShareInvitation](#)中的。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

## AWS CLI

### 向资源共享添加标签

以下`tag-resource`示例向指定的资源共享添加标签键`project`和关联值`lima`。

```
aws ram tag-resource \  
  --tags key=project,value=lima \  
  --resource-share-arn arn:aws:ram:us-west-2:123456789012:resource-share/7ab63972-  
b505-7e2a-420d-6f5d3EXAMPLE
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从资源共享中移除标签

以下untag-resource示例从指定的资源共享中删除project标签键和关联值。

```
aws ram untag-resource \  
  --tag-keys project \  
  --resource-share-arn arn:aws:ram:us-west-2:123456789012:resource-share/7ab63972-  
b505-7e2a-420d-6f5d3EXAMPLE
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-resource-share

以下代码示例演示了如何使用 update-resource-share。

### AWS CLI

更新资源共享

以下update-resource-share示例更改了指定的资源共享，以允许不在 AWS 组织中的外部委托人。

```
aws ram update-resource-share \  
  --allow-external-principals \  
  --resource-share-arn arn:aws:ram:us-west-2:123456789012:resource-share/7ab63972-  
b505-7e2a-420d-6f5d3EXAMPLE
```

输出：

```
{  
  "resourceShare": {
```

```
    "resourceShareArn": "arn:aws:ram:us-west-2:123456789012:resource-  
share/7ab63972-b505-7e2a-420d-6f5d3EXAMPLE",  
    "name": "my-resource-share",  
    "owningAccountId": "123456789012",  
    "allowExternalPrincipals": true,  
    "status": "ACTIVE",  
    "creationTime": 1565295733.282,  
    "lastUpdatedTime": 1565303080.023  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateResourceShare](#)中的。

## 使用资源管理器示例 AWS CLI

以下代码示例向您展示了如何使用 with Resource Explorer 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-default-view**

以下代码示例演示了如何使用 associate-default-view。

AWS CLI

将资源浏览器视图设置为其 AWS 区域的默认视图

以下associate-default-view示例将由其 ARN 指定的视图设置为调用该操作的 AWS 区域的默认视图。

```
aws resource-explorer-2 associate-default-view \  
  --view-arn arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-Main-View/  
EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111
```

输出：

```
{  
  "ViewArn": "arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-Main-  
View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111"  
}
```

有关更多信息，请参阅《AWS 资源浏览器用户指南》中的[在 AWS 区域中设置默认视图](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[AssociateDefaultView](#)中的。

## batch-get-view

以下代码示例演示了如何使用 batch-get-view。

### AWS CLI

检索有关多个资源浏览器视图的详细信息

以下 batch-get-view 示例显示了由其 ARN 指定的两个视图的详细信息。使用空格分隔 --view-arn 参数中的多个 ARN。

```
aws resource-explorer-2 batch-get-view \  
  --view-arns arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-EC2-Only-  
View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE22222, \  
              arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-Main-  
View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111
```

输出：

```
{  
  "Views": [  
    {  
      "Filters": {  
        "FilterString": "service:ec2"  
      },  
      "IncludedProperties": [  

```

```

        {
            "Name": "tags"
        }
    ],
    "LastUpdatedAt": "2022-07-13T21:33:45.249000+00:00",
    "Owner": "123456789012",
    "Scope": "arn:aws:iam::123456789012:root",
    "ViewArn": "arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-
EC2-Only-View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE22222"
    },
    {
        "Filters": {
            "FilterString": ""
        },
        "IncludedProperties": [
            {
                "Name": "tags"
            }
        ],
        "LastUpdatedAt": "2022-07-13T20:34:11.314000+00:00",
        "Owner": "123456789012",
        "Scope": "arn:aws:iam::123456789012:root",
        "ViewArn": "arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-
Main-View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111"
    }
]
"Errors": []
}

```

有关视图的更多信息，请参阅 [《资源浏览器用户指南》中的关于AWS 资源浏览器视图](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetView](#)中的。

## create-index

以下代码示例演示了如何使用 create-index。

### AWS CLI

通过创建索引在 AWS 区域中打开资源浏览器

以下create-index示例在调用该操作的 AWS 区域中创建本地索引。AWS 如果您未指定值，AWS CLI 会自动生成一个随机client-token参数值，并将其包含在对的调用中。

```
aws resource-explorer-2 create-index \  
  --region us-east-1
```

输出：

```
{  
  "Arn": "arn:aws:resource-explorer-2:us-east-1:123456789012:index/EXAMPLE8-90ab-  
cdef-fedc-EXAMPLE22222c",  
  "CreatedAt": "2022-11-01T20:00:59.149Z",  
  "State": "CREATING"  
}
```

创建本地索引后，您可以通过运行[update-index-type](#)命令将其转换为账户的聚合索引。

有关更多信息，请参阅 [《资源浏览器用户指南》](#) 中的在 AWS 区域中打开 AWS 资源浏览器以索引您的资源。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateIndex](#) 中的。

## create-view

以下代码示例演示了如何使用 create-view。

### AWS CLI

示例 1：为区域中的索引创建未经过滤的视图 AWS

以下 create-view 示例在指定 AWS 区域中创建了一个视图，该视图不经过任何筛选即可返回该区域中的所有结果。该视图在返回的结果中包含可选的“标签”字段。由于此视图是在包含聚合器索引的区域中创建的，因此它可以包含账户中包含资源浏览器索引的所有区域的结果。

```
aws resource-explorer-2 create-view \  
  --view-name My-Main-View \  
  --included-properties Name=tags \  
  --region us-east-1
```

输出：

```
{  
  "View": {
```



```

    "Filters": {
      "FilterString": ""
    },
    "IncludedProperties": [
      {
        "Name": "tags"
      }
    ],
    "LastUpdatedAt": "2022-07-13T20:34:11.314000+00:00",
    "Owner": "123456789012",
    "Scope": "arn:aws:iam::123456789012:root",
    "ViewArn": "arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-Main-View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111"
  }
}

```

## 示例 2：创建仅返回与 Amazon EC2 关联的资源的视图

以下内容在 AWS 区域中 `create-view` 创建了一个视图 `us-east-1`，该视图仅返回该区域中与 Amazon EC2 服务关联的资源。该视图包括返回结果中的可选 `Tags` 字段。由于此视图是在包含聚合器索引的区域中创建的，因此它可以包含账户中包含资源浏览器索引的所有区域的结果。

```

aws resource-explorer-2 create-view \
  --view-name My-EC2-Only-View \
  --included-properties Name=tags \
  --filters FilterString="service:ec2" \
  --region us-east-1

```

输出：

```

{
  "View": {
    "Filters": {
      "FilterString": "service:ec2"
    },
    "IncludedProperties": [
      {
        "Name": "tags"
      }
    ],
    "LastUpdatedAt": "2022-07-13T21:35:09.059Z",
    "Owner": "123456789012",
  }
}

```

```
    "Scope": "arn:aws:iam::123456789012:root",
    "ViewArn": "arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-EC2-
Only-View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE22222"
  }
}
```

有关更多信息，请参阅《AWS 资源浏览器用户指南》中的“[创建用于搜索的视图](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateView](#)中的。

## delete-index

以下代码示例演示了如何使用 delete-index。

### AWS CLI

通过删除某个 AWS 区域的索引来关闭该区域中的资源浏览器

以下delete-index示例删除您发出请求的 AWS 区域中指定的资源浏览器索引。

```
aws resource-explorer-2 delete-index \
  --arn arn:aws:resource-explorer-2:us-west-2:123456789012:index/EXAMPLE8-90ab-
cdef-fedc-EXAMPLE22222 \
  --region us-west-2
```

输出：

```
{
  "Arn": "arn:aws:resource-explorer-2:us-west-2:123456789012:index/EXAMPLE8-90ab-
cdef-fedc-EXAMPLE22222",
  "State": "DELETING"
}
```

有关删除索引的更多信息，请参阅《[AWS 资源浏览器用户指南](#)》中的在 [AWS 区域中关闭AWS资源浏览器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteIndex](#)中的。

## delete-view

以下代码示例演示了如何使用 delete-view。

## AWS CLI

### 删除资源浏览器视图

以下delete-view示例删除由其 ARN 指定的视图。

```
aws resource-explorer-2 delete-view \  
  --view-arn arn:aws:resource-explorer-2:us-east-1:123456789012:view/EC2-Only-View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111
```

输出：

```
{  
  "ViewArn": "arn:aws:resource-explorer-2:us-east-1:123456789012:view/EC2-Only-View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111"  
}
```

有关更多信息，请参阅《AWS 资源浏览器用户指南》中的[删除视图](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteView](#)中的。

### disassociate-default-view

以下代码示例演示了如何使用 disassociate-default-view。

## AWS CLI

### 移除某个 AWS 区域的默认资源浏览器视图

以下内容disassociate-default-view删除了您调用操作的 AWS 区域的默认资源浏览器视图。执行此操作后，区域中的所有搜索操作都必须明确指定视图，否则操作将失败。

```
aws resource-explorer-2 disassociate-default-view
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 资源浏览器用户指南》[中的在 AWS 区域中设置默认视图](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateDefaultView](#)中的。

### get-default-view

以下代码示例演示了如何使用 get-default-view。

## AWS CLI

检索资源浏览器视图，该视图是其 AWS 区域的默认视图

以下`get-default-view`示例检索视图的 ARN，该视图是您调用操作所在区域 AWS 的默认视图。

```
aws resource-explorer-2 get-default-view
```

输出：

```
{
  "ViewArn": "arn:aws:resource-explorer-2:us-east-1:123456789012:view/default-view/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111"
}
```

有关更多信息，请参阅《AWS 资源浏览器用户指南》中的[在 AWS 区域中设置默认视图](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetDefaultView](#)中的。

## get-index

以下代码示例演示了如何使用 `get-index`。

### AWS CLI

示例 1：检索资源浏览器聚合器索引的详细信息

以下`get-index`示例显示了指定 AWS 区域中资源浏览器索引的详细信息。由于指定的区域包含该账户的聚合器索引，因此输出会列出将数据复制到该区域索引中的区域。

```
aws resource-explorer-2 get-index \
  --region us-east-1
```

输出：

```
{
  "Arn": "arn:aws:resource-explorer-2:us-east-1:123456789012:index/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111",
  "CreatedAt": "2022-07-12T18:59:10.503000+00:00",
```

```
"LastUpdatedAt": "2022-07-13T18:41:58.799000+00:00",
"ReplicatingFrom": [
  "ap-south-1",
  "us-west-2"
],
"State": "ACTIVE",
"Tags": {},
"Type": "AGGREGATOR"
}
```

## 示例 2：检索资源浏览器本地索引的详细信息

以下 `get-index` 示例显示了指定 AWS 区域中资源浏览器索引的详细信息。由于指定的 Region 包含本地索引，因此输出会列出将该区域索引中的数据复制到的区域。

```
aws resource-explorer-2 get-index \
  --region us-west-2
```

输出：

```
{
  "Arn": "arn:aws:resource-explorer-2:us-west-2:123456789012:index/EXAMPLE8-90ab-cdef-fedc-EXAMPLE22222",
  "CreatedAt": "2022-07-12T18:59:10.503000+00:00",
  "LastUpdatedAt": "2022-07-13T18:41:58.799000+00:00",
  "ReplicatingTo": [
    "us-west-2"
  ],
  "State": "ACTIVE",
  "Tags": {},
  "Type": "LOCAL"
}
```

有关索引的更多信息，请参阅 [《资源浏览器用户指南》中的检查哪些 AWS 区域已开启 AWS 资源浏览器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetIndex](#) 中的。

## get-view

以下代码示例演示了如何使用 `get-view`。

## AWS CLI

检索有关资源浏览器视图的详细信息

以下`get-view`示例显示有关由其 ARN 指定的视图的详细信息。

```
aws resource-explorer-2 get-view \  
  --view-arn arn:aws:resource-explorer-2:us-east-1:123456789012:view/EC2-Only-  
View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111
```

输出：

```
{  
  "Tags" : {},  
  "View" : {  
    "Filters" : {  
      "FilterString" : "service:ec2"  
    },  
    "IncludedProperties" : [  
      {  
        "Name" : "tags"  
      }  
    ],  
    "LastUpdatedAt" : "2022-07-13T21:33:45.249Z",  
    "Owner" : "123456789012",  
    "Scope" : "arn:aws:iam::123456789012:root",  
    "ViewArn" : "arn:aws:resource-explorer-2:us-east-1:123456789012:view/EC2-  
Only-View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111"  
  }  
}
```

有关视图的更多信息，请参阅 [《资源浏览器用户指南》中的关于AWS 资源浏览器视图](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetView](#)中的。

## list-indexes

以下代码示例演示了如何使用 `list-indexes`。

## AWS CLI

列出资源浏览器中包含索引的 AWS 区域

以下`list-indexes`示例列出了资源浏览器具有索引的所有区域的索引。响应指定每个索引的类型、其 AWS 区域及其 ARN。

```
aws resource-explorer-2 list-indexes
```

输出：

```
{
  "Indexes": [
    {
      "Arn": "arn:aws:resource-explorer-2:us-west-2:123456789012:index/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111",
      "Region": "us-west-2",
      "Type": "AGGREGATOR"
    },
    {
      "Arn": "arn:aws:resource-explorer-2:us-east-1:123456789012:index/EXAMPLE8-90ab-cdef-fedc-EXAMPLE22222",
      "Region": "us-east-1",
      "Type": "LOCAL"
    },
    {
      "Arn": "arn:aws:resource-explorer-2:us-east-2:123456789012:index/EXAMPLE8-90ab-cdef-fedc-EXAMPLE33333",
      "Region": "us-east-2",
      "Type": "LOCAL"
    },
    {
      "Arn": "arn:aws:resource-explorer-2:us-west-1:123456789012:index/EXAMPLE8-90ab-cdef-fedc-EXAMPLE44444",
      "Region": "us-west-1",
      "Type": "LOCAL"
    }
  ]
}
```

有关索引的更多信息，请参阅 [《资源浏览器用户指南》中的检查哪些 AWS 区域已开启AWS资源浏览器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListIndexes](#)中的。

## list-supported-resource-types

以下代码示例演示了如何使用 `list-supported-resource-types`。

### AWS CLI

列出资源浏览器中包含索引的 AWS 区域

以下 `list-supported-resource-types` 示例列出了 `&arexLong`; 当前支持的所有资源类型。示例响应包含一个 `NextToken` 值，该值表示有更多输出可供通过其他调用进行检索。

```
aws resource-explorer-2 list-supported-resource-types \  
  --max-items 10
```

输出：

```
{  
  "ResourceTypes": [  
    {  
      "ResourceType": "cloudfront:cache-policy",  
      "Service": "cloudfront"  
    },  
    {  
      "ResourceType": "cloudfront:distribution",  
      "Service": "cloudfront"  
    },  
    {  
      "ResourceType": "cloudfront:function",  
      "Service": "cloudfront"  
    },  
    {  
      "ResourceType": "cloudfront:origin-access-identity",  
      "Service": "cloudfront"  
    },  
    {  
      "ResourceType": "cloudfront:origin-request-policy",  
      "Service": "cloudfront"  
    },  
    {  
      "ResourceType": "cloudfront:realtime-log-config",  
      "Service": "cloudfront"  
    },  
    {
```



```

    "ResourceType": "cloudfront:response-headers-policy",
    "Service": "cloudfront"
  },
  {
    "ResourceType": "cloudwatch:alarm",
    "Service": "cloudwatch"
  },
  {
    "ResourceType": "cloudwatch:dashboard",
    "Service": "cloudwatch"
  },
  {
    "ResourceType": "cloudwatch:insight-rule",
    "Service": "cloudwatch"
  }
],
"NextToken": "eyJ0ZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxMH0="
}

```

要获取输出的下一部分，请再次调用该操作，并将上一个调用的NextToken响应值作为的值传递--starting-token。重复此操作NextToken，直到响应中不存在。

```

aws resource-explorer-2 list-supported-resource-types \
  --max-items 10 \
  --starting-token
eyJ0ZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxMH0=

```

输出：

```

{
  "ResourceTypes": [
    {
      "ResourceType": "cloudwatch:metric-stream",
      "Service": "cloudwatch"
    },
    {
      "ResourceType": "dynamodb:table",
      "Service": "dynamodb"
    },
    {
      "ResourceType": "ec2:capacity-reservation",
      "Service": "ec2"
    }
  ],

```

```
{
  "ResourceType": "ec2:capacity-reservation-fleet",
  "Service": "ec2"
},
{
  "ResourceType": "ec2:client-vpn-endpoint",
  "Service": "ec2"
},
{
  "ResourceType": "ec2:customer-gateway",
  "Service": "ec2"
},
{
  "ResourceType": "ec2:dedicated-host",
  "Service": "ec2"
},
{
  "ResourceType": "ec2:dhcp-options",
  "Service": "ec2"
},
{
  "ResourceType": "ec2:egress-only-internet-gateway",
  "Service": "ec2"
},
{
  "ResourceType": "ec2:elastic-gpu",
  "Service": "ec2"
}
],
"NextToken": "eyJ0ZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyMH0="
}
```

有关索引的更多信息，请参阅 [《资源浏览器用户指南》](#) 中的 [检查哪些 AWS 区域已开启AWS资源浏览器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSupportedResourceTypes](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

## AWS CLI

列出附加到资源管理器视图或索引的标签

以下 `list-tags-for-resource` 示例列出了附加到具有指定 ARN 的 view 的标签键和值对。您必须从包含资源的 AWS 区域调用该操作。

```
aws resource-explorer-2 list-tags-for-resource \
  --resource-arn arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-View/
EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111
```

输出：

```
{
  "Tags": {
    "application": "MainCorpApp",
    "department": "1234"
  }
}
```

有关为视图添加标签的更多信息，请参阅《AWS 资源管理器用户指南》中的 [“为视图添加标签以实现访问控制”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## list-views

以下代码示例演示了如何使用 `list-views`。

## AWS CLI

列出某个 AWS 区域中可用的资源浏览器视图

以下 `list-views` 示例列出了您在其中调用操作的区域中可用的所有视图。

```
aws resource-explorer-2 list-views
```

输出：

```
{
  "Views": [
```

```
    "arn:aws:resource-explorer-2:us-east-1:123456789012:view/EC2-Only-View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111",
    "arn:aws:resource-explorer-2:us-east-1:123456789012:view/Default-All-Resources-View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE22222",
    "arn:aws:resource-explorer-2:us-east-1:123456789012:view/Production-Only-View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE33333"
  ]
}
```

有关视图的更多信息，请参阅 [《资源浏览器用户指南》](#) 中的关于 [AWS 资源浏览器视图](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListViews](#) 中的。

## search

以下代码示例演示了如何使用 search。

### AWS CLI

示例 1：使用默认视图进行搜索

以下 search 示例显示了指定中与该服务关联的所有资源。搜索使用该地区的默认视图。示例响应包含一个 NextToken 值，该值表示有更多输出可供通过其他调用进行检索。

```
aws resource-explorer-2 search \
  --query-string "service:iam"
```

输出：

```
{
  "Count": {
    "Complete": true,
    "TotalResources": 55
  },
  "NextToken":
  "AG9V0EF1KLEXAMPLE0hJHVwo5chEXAMPLER5XiEpNrgsEXAMPLE...b0Cm0F0ryHEXAMPLE",
  "Resources": [{
    "Arn": "arn:aws:iam::123456789012:policy/service-role/Some-Policy-For-A-Service-Role",
    "LastReportedAt": "2022-07-21T12:34:42Z",
    "OwningAccountId": "123456789012",
    "Properties": [],
    "Region": "global",
```

```

    "ResourceType": "iam:policy",
    "Service": "iam"
  }, {
    "Arn": "arn:aws:iam::123456789012:policy/service-role/Another-Policy-For-A-Service-Role",
    "LastReportedAt": "2022-07-21T12:34:42Z",
    "OwningAccountId": "123456789012",
    "Properties": [],
    "Region": "global",
    "ResourceType": "iam:policy",
    "Service": "iam"
  }, {
    ... TRUNCATED FOR BREVITY ...
  }],
  "ViewArn": "arn:aws:resource-explorer-2:us-east-1:123456789012:view/my-default-view/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111"
}

```

## 示例 2：使用指定视图进行搜索

以下search示例搜索显示了指定 AWS 区域中通过指定视图可见的所有资源（“\*”）。由于视图附带筛选条件，结果仅包含与 Amazon EC2 关联的资源。

```

aws resource-explorer-2 search \
  -- query-string "*" \
  -- view-arn arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-EC2-view/EXAMPLE8-90ab-cdef-fedc-EXAMPLE22222

```

输出：

```

HTTP/1.1 200 OK
Date: Tue, 01 Nov 2022 20:00:59 GMT
Content-Type: application/json
Content-Length: <PayloadSizeBytes>

{
  "Count": {
    "Complete": true,
    "TotalResources": 67
  },
  "Resources": [{
    "Arn": "arn:aws:ec2:us-east-1:123456789012:network-acl/acl-1a2b3c4d",
    "LastReportedAt": "2022-07-21T18:52:02Z",

```

```
"OwningAccountId": "123456789012",
"Properties": [{
  "Data": [{
    "Key": "Department",
    "Value": "AppDevelopment"
  }, {
    "Key": "Environment",
    "Value": "Production"
  }],
  "LastReportedAt": "2021-11-15T14:48:29Z",
  "Name": "tags"
}],
"Region": "us-east-1",
"ResourceType": "ec2:network-acl",
"Service": "ec2"
}, {
  "Arn": "arn:aws:ec2:us-east-1:123456789012:subnet/subnet-1a2b3c4d",
  "LastReportedAt": "2022-07-21T21:22:23Z",
  "OwningAccountId": "123456789012",
  "Properties": [{
    "Data": [{
      "Key": "Department",
      "Value": "AppDevelopment"
    }, {
      "Key": "Environment",
      "Value": "Production"
    }],
    "LastReportedAt": "2021-07-29T19:02:39Z",
    "Name": "tags"
  }],
  "Region": "us-east-1",
  "ResourceType": "ec2:subnet",
  "Service": "ec2"
}, {
  "Arn": "arn:aws:ec2:us-east-1:123456789012:dhcp-options/dopt-1a2b3c4d",
  "LastReportedAt": "2022-07-21T06:08:53Z",
  "OwningAccountId": "123456789012",
  "Properties": [{
    "Data": [{
      "Key": "Department",
      "Value": "AppDevelopment"
    }, {
      "Key": "Environment",
      "Value": "Production"
    }],
    "LastReportedAt": "2021-07-29T19:02:39Z",
    "Name": "tags"
  }],
  "Region": "us-east-1",
  "ResourceType": "ec2:dhcp-options",
  "Service": "ec2"
}
```

```
    ]],
    "LastReportedAt": "2021-11-15T15:11:05Z",
    "Name": "tags"
  ]],
  "Region": "us-east-1",
  "ResourceType": "ec2:dhcptions",
  "Service": "ec2"
}, {
  ... TRUNCATED FOR BREVITY ...
}],
  "ViewArn": "arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-EC2-
view/EXAMPLE8-90ab-cdef-fedc-EXAMPLE22222"
}
```

有关更多信息，请参阅 [《AWS 资源浏览器用户指南》](#) 中的 [使用AWS 资源浏览器搜索资源](#)。

- 有关 API 的详细信息，请参阅在 AWS CLI 命令参考中 [搜索](#)。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源浏览器视图添加标签

以下 tag-resource 示例将值为“生产”的标签键“环境”添加到具有指定 ARN 的视图中。

```
aws resource-explorer-2 tag-resource \
  --resource-arn arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-View//
EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111 \
  --tags environment=production
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS 资源管理器用户指南》](#) 中的 [“为视图添加标签以实现访问控制”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

## AWS CLI

从资源管理器视图中移除标签

以下 `untag-resource` 示例从具有指定 ARN 的视图中删除所有键名为 “environment” 的标签。

```
aws resource-explorer-2 untag-resource \  
  --resource-arn arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-View//  
EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111 \  
  --tag-keys environment
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 资源管理器用户指南》中的 [“为视图添加标签以实现访问控制”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-index-type

以下代码示例演示了如何使用 `update-index-type`。

### AWS CLI

更改资源浏览器索引的类型

以下 `update-index-type` 示例将指定的索引从类型 `localaggregator` 转换为类型 `aggregator`，以开启在账户中所有 AWS 区域搜索资源的功能。您必须将请求发送到包含要更新的索引的 AWS 区域。

```
aws resource-explorer-2 update-index-type \  
  --arn arn:aws:resource-explorer-2:us-east-1:123456789012:index/EXAMPLE8-90ab-  
cdef-fedc-EXAMPLE11111 \  
  --type aggregator \  
  --region us-east-1
```

输出：

```
{  
  "Arn": "arn:aws:resource-explorer-2:us-east-1:123456789012:index/EXAMPLE8-90ab-  
cdef-fedc-EXAMPLE11111",  
  "LastUpdatedAt": "2022-07-13T18:41:58.799Z",  
  "State": "updating",
```



```
"Type": "aggregator"
}
```

有关更改索引类型的更多信息，请参阅《AWS 资源浏览器用户指南》中的[通过创建聚合器索引来启用跨区域搜索](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateIndexType](#) 中的。

## update-view

以下代码示例演示了如何使用 update-view。

### AWS CLI

示例 1：更新资源浏览器视图的 IncludedProperties 字段

以下 update-view 示例通过添加 `tags` 可选视图来更新指定的视图 `IncludedProperties`。运行此操作后，使用此视图的搜索操作将包含有关结果中显示的资源所附标签的信息。

```
aws resource-explorer-2 update-view \
  --included-properties Name=tags \
  --view-arn arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-View/
EXAMPLE8-90ab-cdef-fedc-EXAMPLE22222
```

输出：

```
{
  "View": {
    "Filters": {
      "FilterString": ""
    },
    "IncludedProperties": [
      {
        "Name": "tags"
      }
    ],
    "LastUpdatedAt": "2022-07-19T17:41:21.710000+00:00",
    "Owner": "123456789012",
    "Scope": "arn:aws:iam::123456789012:root",
    "ViewArn": "arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-EC2-
Only-View/EXAMPLE8-90ab-cdef-fedc-EXAMPLE11111"
  }
}
```

```
}
```

## 示例 2：更新附加到视图的筛选器

以下update-view示例将指定视图更新为使用筛选条件，该筛选条件将结果仅限于与 Amazon EC2 服务关联的资源类型。

```
aws resource-explorer-2 update-view \  
  --filters FilterString="service:ec2" \  
  --view-arn arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-View/  
EXAMPLE8-90ab-cdef-fedc-EXAMPLE22222
```

输出：

```
{  
  "View": {  
    "Filters": {  
      "FilterString": "service:ec2"  
    },  
    "IncludedProperties": [],  
    "LastUpdatedAt": "2022-07-19T17:41:21.710000+00:00",  
    "Owner": "123456789012",  
    "Scope": "arn:aws:iam::123456789012:root",  
    "ViewArn": "arn:aws:resource-explorer-2:us-east-1:123456789012:view/My-View/  
EXAMPLE8-90ab-cdef-fedc-EXAMPLE22222"  
  }  
}
```

有关视图的更多信息，请参阅 [《资源浏览器用户指南》中的关于AWS 资源浏览器视图](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateView](#)中的。

## 使用的 Resource Groups AWS CLI

以下代码示例向您展示了如何使用 with Resource Groups 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### create-group

以下代码示例演示了如何使用 create-group。

#### AWS CLI

##### 示例 1：创建基于标签的资源组

以下create-group示例在当前区域创建了一个基于标签的 Amazon EC2 实例资源组。它基于对带有密钥Name和值的资源的查询WebServers。群组名称是tbq-WebServer。该查询位于传递给命令的单独的 JSON 文件中。

```
aws resource-groups create-group \  
  --name tbq-WebServer \  
  --resource-query file://query.json
```

query.json 的内容：

```
{  
  "Type": "TAG_FILTERS_1_0",  
  "Query": "{\"ResourceTypeFilters\": [\"AWS::EC2::Instance\"], \"TagFilters\":  
  [{\"Key\": \"Name\", \"Values\": [\"WebServers\"]}]}"
```

输出：

```
{  
  "Group": {  
    "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-  
WebServer",  
    "Name": "tbq-WebServer"  
  },  
}
```

```

    "ResourceQuery": {
      "Type": "TAG_FILTERS_1_0",
      "Query": "{\"ResourceTypeFilters\": [\"AWS::EC2::Instance\"], \"TagFilters\": [ { \"Key\": \"Name\", \"Values\": [\"WebServers\"] } ]}"
    }
  }
}

```

## 示例 2：创建 CloudFormation 基于堆栈的资源组

以下 create-group 示例创建了一个名为的 AWS CloudFormation 基于堆栈的资源组。sampleCFNstackgroup 该查询包括指定 CloudFormation 堆栈中所有受 Resource Group AWS s 支持的资源。

```

aws resource-groups create-group \
  --name cbq-CFNstackgroup \
  --resource-query file://query.json

```

query.json 的内容：

```

{
  "Type": "CLOUDFORMATION_STACK_1_0",
  "Query": "{\"ResourceTypeFilters\": [\"AWS::AllSupported\"], \"StackIdentifier\": \"arn:aws:cloudformation:us-west-2:123456789012:stack/MyCFNStack/1415z9z0-z39z-11z8-97z5-500z212zz6fz\"}"
}

```

输出：

```

{
  "Group": {
    "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/cbq-CFNstackgroup",
    "Name": "cbq-CFNstackgroup"
  },
  "ResourceQuery": {
    "Type": "CLOUDFORMATION_STACK_1_0",
    "Query": "{\"ResourceTypeFilters\": [\"AWS::AllSupported\"], \"StackIdentifier\": \"arn:aws:cloudformation:us-east-2:123456789012:stack/MyCFNStack/1415z9z0-z39z-11z8-97z5-500z212zz6fz\"}"
  }
}

```

有关更多信息，请参阅 [Res AWS ource Groups 用户指南中的创建](#) 群组。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateGroup](#) 中的。

## delete-group

以下代码示例演示了如何使用 delete-group。

### AWS CLI

#### 更新资源组的描述

以下 delete-group 示例更新了指定的资源组。

```
aws resource-groups delete-group \  
  --group-name tbq-WebServer
```

输出：

```
{  
  "Group": {  
    "GroupArn": "arn:aws:resource-groups:us-west-2:1234567890:group/tbq-  
WebServer",  
    "Name": "tbq-WebServer"  
  }  
}
```

有关更多信息，请参阅《[Res AWS ource Groups 用户指南](#)》中的 [删除](#) 群组。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteGroup](#) 中的。

## get-group-query

以下代码示例演示了如何使用 get-group-query。

### AWS CLI

#### 将查询附加到资源组

以下 get-group-query 示例显示了附加到指定资源组的查询。

```
aws resource-groups get-group-query \  
  --group-name tbq-WebServer
```

```
--group-name tbq-WebServer
```

输出：

```
{
  "GroupQuery": {
    "GroupName": "tbq-WebServer",
    "ResourceQuery": {
      "Type": "TAG_FILTERS_1_0",
      "Query": "{\"ResourceTypeFilters\": [\"AWS::EC2::Instance\"], \"TagFilters\": [{\"Key\": \"Name\", \"Values\": [\"WebServers\"]}]}"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetGroupQuery](#)中的。

## get-group

以下代码示例演示了如何使用 get-group。

### AWS CLI

获取有关资源组的信息

以下get-group示例显示有关指定资源组的详细信息。要将查询附加到群组，请使用get-group-query。

```
aws resource-groups get-group \
  --group-name tbq-WebServer
```

输出：

```
{
  "Group": {
    "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer",
    "Name": "tbq-WebServer",
    "Description": "A tag-based query resource group of WebServers."
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetGroup](#)中的。

## get-tags

以下代码示例演示了如何使用 get-tags。

### AWS CLI

检索附加到资源组的标签

以下get-tags示例显示了附加到指定资源组（组本身，而不是其成员）的标签键和值对。

```
aws resource-groups get-tags \  
  --arn arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer
```

输出：

```
{  
  "Arn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer",  
  "Tags": {  
    "QueryType": "tags",  
    "QueryResources": "ec2-instances"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTags](#)中的。

## list-group-resources

以下代码示例演示了如何使用 list-group-resources。

### AWS CLI

列出资源组中的所有资源

示例 1：以下list-resource-groups示例列出了属于指定资源组的所有资源。

```
aws resource-groups list-group-resources \  
  --group-name tbq-WebServer
```

输出：

```
{
  "ResourceIdentifiers": [
    {
      "ResourceArn": "arn:aws:ec2:us-west-2:123456789012:instance/i-09f77fa38c12345ab",
      "ResourceType": "AWS::EC2::Instance"
    }
  ]
}
```

示例 2：以下示例列出了组中所有资源的“资源类型”也为“:: EC2 AWS:: Instance”。：

aws 资源组 list-group-resources --group-name tbq-WebServer --filters name=Resource-type, Values=:: EC2:: Instance AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListGroupResources](#)中的。

## list-groups

以下代码示例演示了如何使用 list-groups。

### AWS CLI

列出可用的资源组

以下list-groups示例显示了所有资源组的列表。

```
aws resource-groups list-groups
```

输出：

```
{
  "GroupIdentifiers": [
    {
      "GroupName": "tbq-WebServer",
      "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer3"
    },
    {
      "GroupName": "cbq-CFNStackQuery",
      "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/cbq-CFNStackQuery"
    }
  ]
}
```



```
    }
  ],
  "Groups": [
    {
      "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer",
      "Name": "tbq-WebServer"
    },
    {
      "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/cbq-CFNStackQuery",
      "Name": "cbq-CFNStackQuery"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListGroups](#)中的。

## list-resource-groups

以下代码示例演示了如何使用 `list-resource-groups`。

### AWS CLI

列出资源组中的所有资源

以下`list-resource-groups`示例列出了属于指定资源组的所有资源。

```
aws resource-groups list-group-resources \
  --group-name tbq-WebServer
```

输出：

```
{
  "ResourceIdentifiers": [
    {
      "ResourceArn": "arn:aws:ec2:us-west-2:123456789012:instance/i-09f77fa38c12345ab",
      "ResourceType": "AWS::EC2::Instance"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResourceGroups](#)中的。

## put-group-configuration

以下代码示例演示了如何使用 put-group-configuration。

### AWS CLI

将服务配置附加到资源组

示例 1：以下put-group-configuration示例指定资源组仅包含C5或M5系列中实例的 Amazon EC2 容量预留。

```
aws resource-groups put-group-configuration \  
  --group MyTestGroup \  
  --configuration file://config.json
```

config.json 的内容：

```
[  
  {  
    "Type": "AWS::EC2::HostManagement",  
    "Parameters": [  
      {  
        "Name": "allowed-host-families",  
        "Values": [ "c5", "m5" ]  
      },  
      {  
        "Name": "any-host-based-license-configuration",  
        "Values": [ "true" ]  
      }  
    ]  
  },  
  {  
    "Type": "AWS::ResourceGroups::Generic",  
    "Parameters": [  
      {  
        "Name": "allowed-resource-types",  
        "Values": [ "AWS::EC2::Host" ]  
      },  
      {  
        "Name": "deletion-protection",
```

```

    "Values": [ "UNLESS_EMPTY" ]
  }
]
}
]

```

如果成功，此命令不会产生任何输出。

有关更多信息，请参阅 [Resource Groups API 参考指南中的资源组服务配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutGroupConfiguration](#) 中的。

## search-resources

以下代码示例演示了如何使用 `search-resources`。

### AWS CLI

#### 查找与查询相匹配的资源

以下 `search-resources` 示例检索与指定查询相匹配的所有 AWS 资源的列表。

```
aws resource-groups search-resources \
  --resource-query file://query.json
```

`query.json` 的内容：

```
{
  "Type": "TAG_FILTERS_1_0",
  "Query": "{\"ResourceTypeFilters\": [\"AWS::EC2::Instance\"], \"TagFilters\": [ { \"Key\": \"Patch Group\", \"Values\": [\"Dev\"] } ]}"
}
```

输出：

```
{
  "ResourceIdentifiers": [
    {
      "ResourceArn": "arn:aws:ec2:us-west-2:123456789012:instance/i-01a23bc45d67890ef",
      "ResourceType": "AWS::EC2::Instance"
    }
  ]
}
```

```
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SearchResources](#) 中的。

## tag

以下代码示例演示了如何使用 tag。

### AWS CLI

将标签附加到资源组

以下 tag 示例将指定的标签键和值对附加到指定的资源组（组本身，而不是其成员）。

```
aws resource-groups tag \  
  --tags QueryType=tags,QueryResources=ec2-instances \  
  --arn arn:aws:resource-groups:us-west-2:128716708097:group/tbq-WebServer
```

输出：

```
{  
  "Arn": "arn:aws:resource-groups:us-west-2:128716708097:group/tbq-WebServer",  
  "Tags": {  
    "QueryType": "tags",  
    "QueryResources": "ec2-instances"  
  }  
}
```

有关更多信息，请参阅 Res AWS ource Groups 用户指南中的 [管理标签](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [“标签”](#)。

## untag

以下代码示例演示了如何使用 untag。

### AWS CLI

从资源组中移除标签

以下untags示例从资源组本身（而不是其成员）中移除任何具有指定密钥的标签。

```
aws resource-groups untag \  
  --arn arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer \  
  --keys QueryType
```

输出：

```
{  
  "Arn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer",  
  "Keys": [  
    "QueryType"  
  ]  
}
```

有关更多信息，请参阅 Res AWS ource Groups 用户指南中的[管理标签](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的[取消标记](#)。

## update-group-query

以下代码示例演示了如何使用 update-group-query。

### AWS CLI

示例 1：更新基于标签的资源组的查询

以下update-group-query示例更新了附加到指定基于标签的资源组的查询。

```
aws resource-groups update-group-query \  
  --group-name tbq-WebServer \  
  --resource-query '{"Type":"TAG_FILTERS_1_0", "Query":{"ResourceTypeFilters\  
  ["AWS::EC2::Instance"], "TagFilters":[{"Key":"Name", "Values":["WebServers"]}]}}'
```

输出：

```
{  
  "Group": {  
    "GroupArn": "arn:aws:resource-groups:us-east-2:123456789012:group/tbq-  
    WebServer",
```

```

    "Name": "tbq-WebServer"
  },
  "ResourceQuery": {
    "Type": "TAG_FILTERS_1_0",
    "Query": "{\"ResourceTypeFilters\": [\"AWS::EC2::Instance\"], \"TagFilters\": [ { \"Key\": \"Name\", \"Values\": [ \"WebServers\" ] } ] }"
  }
}

```

有关更多信息，请参阅 [Res AWS ource Groups 用户指南中的更新](#) 群组。

## 示例 2：更新 CloudFormation 基于堆栈的资源组的查询

以下 update-group-query 示例更新了附加到指定 AWS CloudFormation 基于堆栈的资源组的查询。

```

aws resource-groups update-group-query \
  --group-name cbq-CFNstackgroup \
  --resource-query '{"Type": "CLOUDFORMATION_STACK_1_0", "Query": "\
  {"ResourceTypeFilters": ["AWS::AllSupported"], "StackIdentifier": "\
  arn:aws:cloudformation:us-west-2:123456789012:stack/MyCFNStack/1415z9z0-z39z-11z8-97z5-500z212zz6fz"}"}'

```

输出：

```

{
  "Group": {
    "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/cbq-CFNstackgroup",
    "Name": "cbq-CFNstackgroup"
  },
  "ResourceQuery": {
    "Type": "CLOUDFORMATION_STACK_1_0",
    "Query": "{\"ResourceTypeFilters\": [\"AWS::AllSupported\"], \"StackIdentifier\": \"arn:aws:cloudformation:us-west-2:123456789012:stack/MyCFNStack/1415z9z0-z39z-11z8-97z5-500z212zz6fz\"}"
  }
}

```

有关更多信息，请参阅 [Res AWS ource Groups 用户指南中的更新](#) 群组。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateGroupQuery](#) 中的。

## update-group

以下代码示例演示了如何使用 update-group。

### AWS CLI

#### 更新资源组的描述

以下update-group示例更新了指定资源组的描述。

```
aws resource-groups update-group \  
  --group-name tbq-WebServer \  
  --description "Resource group for all web server resources."
```

输出：

```
{  
  "Group": {  
    "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-  
WebServer",  
    "Name": "tbq-WebServer"  
    "Description": "Resource group for all web server resources."  
  }  
}
```

有关更多信息，请参阅 [Res AWS ource Groups 用户指南中的更新群组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateGroup](#)中的。

## Resource Groups 使用标记 API 示例 AWS CLI

以下代码示例向您展示了如何使用 with Resource Groups Tagging API 来执行操作和实现常见场景。

### AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### get-resources

以下代码示例演示了如何使用 `get-resources`。

#### AWS CLI

获取已标记资源的列表

以下 `get-resources` 示例显示账户中使用指定密钥名称和值标记的资源列表。

```
aws resourcegroupstaggingapi get-resources \
  --tag-filters Key=Environment,Values=Production \
  --tags-per-page 100
```

输出：

```
{
  "ResourceTagMappingList": [
    {
      "ResourceARN": " arn:aws:inspector:us-west-2:123456789012:target/0-
nvgVhaxX/template/0-7sbz2Kz0",
      "Tags": [
        {
          "Key": "Environment",
          "Value": "Production"
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅 [Resource Groups 标记 API 参考](#) 中的 `GetResources`。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) `GetResources` 中的。

### get-tag-keys

以下代码示例演示了如何使用 `get-tag-keys`。



## AWS CLI

获取所有标签键的列表

以下`get-tag-keys`示例检索账户中资源使用的所有标签密钥名称的列表。

```
aws resourcegroupstaggingapi get-tag-keys
```

输出：

```
{
  "TagKeys": [
    "Environment",
    "CostCenter",
    "Department"
  ]
}
```

有关更多信息，请参阅 Resour [GetTagKeys](#)ce G rroups 标记 API 参考中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTagKeys](#)中的。

## get-tag-values

以下代码示例演示了如何使用 `get-tag-values`。

## AWS CLI

获取所有标签值的列表

以下`get-tag-values`示例显示了用于中所有资源的指定密钥的所有值

```
aws resourcegroupstaggingapi get-tag-values \
  --key=Environment
```

输出：

```
{
  "TagValues": [
    "Alpha",
    "Gamma",
    "Production"
  ]
}
```

```
]
}
```

有关更多信息，请参阅 [Resource Groups 标记 API 参考](#) 中的。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 GetTagValues](#) 中的。

## tag-resources

以下代码示例演示了如何使用 tag-resources。

### AWS CLI

为资源附加标签

以下 tag-resources 示例使用密钥名称和值对指定资源进行标记。

```
aws resourcegroupstaggingapi tag-resources \
  --resource-arn-list arn:aws:s3:::MyProductionBucket \
  --tags Environment=Production,CostCenter=1234
```

输出：

```
{
  "FailedResourcesMap": {}
}
```

有关更多信息，请参阅 [Resource Groups 标记 API 参考](#) 中的。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 TagResources](#) 中的。

## untag-resources

以下代码示例演示了如何使用 untag-resources。

### AWS CLI

从资源中移除标签

以下 untag-resources 示例从指定资源中删除指定的标签键和任何关联值。

```
aws resourcegroupstaggingapi untag-resources \
```

```
--resource-arn-list arn:aws:s3:::awsexamplebucket \  
--tag-keys Environment CostCenter
```

输出：

```
{  
  "FailedResourcesMap": {}  
}
```

有关更多信息，请参阅 [Resource Groups 标记 API 参考](#) 中的。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 UntagResources](#) 中的。

## AWS RoboMaker 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS RoboMaker。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **batch-describe-simulation-job**

以下代码示例演示了如何使用 `batch-describe-simulation-job`。

AWS CLI

批量描述模拟作业

以下 `batch-describe-simulation-job` 示例检索三个指定模拟作业的详细信息。

命令：

```
aws robomaker batch-describe-simulation-job \
--job arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-66bbb3gpxm8x
arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-p0cpdrrwng2n
arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-g8h6tg1mblgw
```

输出：

```
{
  "jobs": [
    {
      "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-66bbb3gpxm8x",
      "status": "Completed",
      "lastUpdatedAt": 1548959178.0,
      "failureBehavior": "Continue",
      "clientRequestToken": "6020408e-b05c-4310-9f13-4ed71c5221ed",
      "outputLocation": {
        "s3Bucket": "awsrobomakerobjecttracker-1111111111-bundlesbucket-2lk584kiq1oa",
        "s3Prefix": "output"
      },
      "maxJobDurationInSeconds": 3600,
      "simulationTimeMillis": 0,
      "iamRole": "arn:aws:iam::111111111111:role/AWSRoboMakerObjectTracker-154895-SimulationJobRole-14D5ASA7PQE3A",
      "simulationApplications": [
        {
          "application": "arn:aws:robomaker:us-west-2:111111111111:simulation-application/AWSRoboMakerObjectTracker-1548959046124_NPvyfcatq/1548959170096",
          "applicationVersion": "$LATEST",
          "launchConfig": {
            "packageName": "object_tracker_simulation",
            "launchFile": "local_training.launch",
            "environmentVariables": {
              "MARKOV_PRESET_FILE": "object_tracker.py",
              "MODEL_S3_BUCKET": "awsrobomakerobjecttracker-1111111111-bundlesbucket-2lk584kiq1oa",
              "MODEL_S3_PREFIX": "model-store",
              "ROS_AWS_REGION": "us-west-2"
            }
          }
        }
      ]
    }
  ]
}
```

```

    ],
    "tags": {},
    "vpcConfig": {
      "subnets": [
        "subnet-716dd52a",
        "subnet-43c22325",
        "subnet-3f526976"
      ],
      "securityGroups": [
        "sg-3fb40545"
      ],
      "vpcId": "vpc-99895eff",
      "assignPublicIp": true
    }
  },
  {
    "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-
p0cpdrrwng2n",
    "status": "Completed",
    "lastUpdatedAt": 1548168817.0,
    "failureBehavior": "Continue",
    "clientRequestToken": "e4a23e75-f9a7-411d-835f-21881c82c58b",
    "outputLocation": {
      "s3Bucket": "awsrobomakercloudwatch-111111111111-
bundlesbucket-14e5s9jvwtmv7",
      "s3Prefix": "output"
    },
    "maxJobDurationInSeconds": 3600,
    "simulationTimeMillis": 0,
    "iamRole": "arn:aws:iam::111111111111:role/
AWSRoboMakerCloudWatch-154766341-SimulationJobRole-G00BWTQ8YBG6",
    "robotApplications": [
      {
        "application": "arn:aws:robomaker:us-west-2:111111111111:robot-
application/AWSRoboMakerCloudWatch-1547663411642_NZbpqEJ3T/1547663517377",
        "applicationVersion": "$LATEST",
        "launchConfig": {
          "packageName": "cloudwatch_robot",
          "launchFile": "await_commands.launch",
          "environmentVariables": {
            "LAUNCH_ID": "1548168752173",
            "ROS_AWS_REGION": "us-west-2"
          }
        }
      }
    ]
  }
}

```

```
    }
  ],
  "simulationApplications": [
    {
      "application": "arn:aws:robomaker:us-
west-2:111111111111:simulation-application/
AWSRoboMakerCloudWatch-1547663411642_0LI6D1h6/1547663521470",
      "applicationVersion": "$LATEST",
      "launchConfig": {
        "packageName": "cloudwatch_simulation",
        "launchFile": "bookstore_turtlebot_navigation.launch",
        "environmentVariables": {
          "LAUNCH_ID": "1548168752173",
          "ROS_AWS_REGION": "us-west-2",
          "TURTLEBOT3_MODEL": "waffle_pi"
        }
      }
    }
  ],
  "tags": {},
  "vpcConfig": {
    "subnets": [
      "subnet-716dd52a",
      "subnet-43c22325",
      "subnet-3f526976"
    ],
    "securityGroups": [
      "sg-3fb40545"
    ],
    "vpcId": "vpc-99895eff",
    "assignPublicIp": true
  }
},
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-
g8h6tglmblgw",
  "status": "Canceled",
  "lastUpdatedAt": 1546543442.0,
  "failureBehavior": "Fail",
  "clientRequestToken": "d796bbb4-2a2c-1abc-f2a9-0d9e547d853f",
  "outputLocation": {
    "s3Bucket": "sample-bucket",
    "s3Prefix": "SimulationLog_115490482698"
  }
},
```

```

    "maxJobDurationInSeconds": 28800,
    "simulationTimeMillis": 0,
    "iamRole": "arn:aws:iam::111111111111:role/RoboMakerSampleTheFirst",
    "robotApplications": [
      {
        "application": "arn:aws:robomaker:us-west-2:111111111111:robot-
application/RoboMakerHelloWorldRobot/1546541208251",
        "applicationVersion": "$LATEST",
        "launchConfig": {
          "packageName": "hello_world_robot",
          "launchFile": "rotate.launch"
        }
      }
    ],
    "simulationApplications": [
      {
        "application": "arn:aws:robomaker:us-
west-2:111111111111:simulation-application/
RoboMakerHelloWorldSimulation/1546541198985",
        "applicationVersion": "$LATEST",
        "launchConfig": {
          "packageName": "hello_world_simulation",
          "launchFile": "empty_world.launch"
        }
      }
    ],
    "tags": {}
  }
],
  "unprocessedJobs": []
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchDescribeSimulationJob](#)中的。

## cancel-simulation-job

以下代码示例演示了如何使用 `cancel-simulation-job`。

### AWS CLI

#### 取消模拟作业

以下`cancel-simulation-job`示例取消了指定的模拟作业。

```
aws robomaker cancel-simulation-job \  
  --job arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-66bbb3gpxm8x
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelSimulationJob](#)中的。

## create-deployment-job

以下代码示例演示了如何使用 create-deployment-job。

### AWS CLI

#### 创建部署作业

此示例为舰队创建部署任务 MyFleet。它包括一个名为“环境”的环境变量。它还附加了一个名为“区域”的标签。

命令:

```
aws robomaker create-deployment-job --deployment-config  
  concurrentDeploymentPercentage=20,failureThresholdPercentage=25  
  --fleet arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/  
  Trek/1539894765711 --tags Region=West --deployment-application-configs  
  application=arn:aws:robomaker:us-west-2:111111111111:robot-application/  
  RoboMakerVoiceInteractionRobot/1546537110575,applicationVersion=1,launchConfig={environmentV
```

输出:

```
{  
  "arn": "arn:aws:robomaker:us-west-2:111111111111:deployment-job/sim-0974h36s4v0t",  
  "fleet": "arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/  
  MyFleet/1539894765711",  
  "status": "Pending",  
  "deploymentApplicationConfigs": [  
    {  
      "application": "arn:aws:robomaker:us-west-2:111111111111:robot-  
  application/RoboMakerVoiceInteractionRobot/1546537110575",  
      "applicationVersion": "1",  
      "launchConfig": {  
        "packageName": "voice_interaction_robot",  
        "launchFile": "await_commands.launch",  
        "environmentVariables": {
```



```
        "ENVIRONMENT": "Beta"
      }
    }
  ],
  "createdAt": 1550770236.0,
  "deploymentConfig": {
    "concurrentDeploymentPercentage": 20,
    "failureThresholdPercentage": 25
  },
  "tags": {
    "Region": "West"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDeploymentJob](#)中的。

## create-fleet

以下代码示例演示了如何使用 create-fleet。

### AWS CLI

#### 创建舰队

此示例创建了一个舰队。它附加了一个名为 Region 的标签。

命令:

```
aws robomaker create-fleet --name MyFleet --tags Region=East
```

输出:

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/
MyOtherFleet/1550771394395",
  "name": "MyFleet",
  "createdAt": 1550771394.0,
  "tags": {
    "Region": "East"
  }
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFleet](#)中的。

## create-robot-application-version

以下代码示例演示了如何使用 `create-robot-application-version`。

### AWS CLI

#### 创建机器人应用程序版本

此示例创建了一个机器人应用程序版本。

命令:

```
aws robomaker create-robot-application-version --application arn:aws:robomaker:us-west-2:111111111111:robot-application/MyRobotApplication/1551201873931
```

输出:

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:robot-application/MyRobotApplication/1551201873931",
  "name": "MyRobotApplication",
  "version": "1",
  "sources": [
    {
      "s3Bucket": "my-bucket",
      "s3Key": "my-robot-application.tar.gz",
      "etag": "f8cf5526f1c6e7b3a72c3ed3f79c5493-70",
      "architecture": "ARMHF"
    }
  ],
  "robotSoftwareSuite": {
    "name": "ROS",
    "version": "Kinetic"
  },
  "lastUpdatedAt": 1551201873.0,
  "revisionId": "9986bb8d-a695-4ab4-8810-9f4a74d1aa00"
  "tags": {}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRobotApplicationVersion](#)中的。

## create-robot-application

以下代码示例演示了如何使用 create-robot-application。

### AWS CLI

#### 创建机器人应用程序

此示例创建了一个机器人应用程序。

命令:

```
aws robomaker create-robot-application --name MyRobotApplication --sources
s3Bucket=my-bucket,s3Key=my-robot-application.tar.gz,architecture=X86_64 --robot-
software-suite name=ROS,version=Kinetic
```

输出：

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:robot-application/
MyRobotApplication/1551201873931",
  "name": "MyRobotApplication",
  "version": "$LATEST",
  "sources": [
    {
      "s3Bucket": "my-bucket",
      "s3Key": "my-robot-application.tar.gz",
      "architecture": "ARMHF"
    }
  ],
  "robotSoftwareSuite": {
    "name": "ROS",
    "version": "Kinetic"
  },
  "lastUpdatedAt": 1551201873.0,
  "revisionId": "1f3cb539-9239-4841-a656-d3efcffa07e1",
  "tags": {}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRobotApplication](#)中的。

## create-robot

以下代码示例演示了如何使用 create-robot。

### AWS CLI

#### 创建机器人

此示例创建了一个机器人。它使用 ARMHF 架构。它还附加了一个名为 Region 的标签。

#### 命令:

```
aws robomaker create-robot --name MyRobot --architecture ARMHF --greengrass-group-id
0f728a3c-7dbf-4a3e-976d-d16a8360caba --tags Region=East
```

#### 输出:

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:robot/MyRobot/1550772324398",
  "name": "MyRobot",
  "createdAt": 1550772325.0,
  "greengrassGroupId": "0f728a3c-7dbf-4a3e-976d-d16a8360caba",
  "architecture": "ARMHF",
  "tags": {
    "Region": "East"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRobot](#)中的。

## create-simulation-application-version

以下代码示例演示了如何使用 create-simulation-application-version。

### AWS CLI

#### 创建仿真应用程序版本

此示例创建了一个机器人应用程序版本。

#### 命令:

```
aws robomaker create-simulation-application-version --application
arn:aws:robomaker:us-west-2:111111111111:robot-application/
MySimulationApplication/1551203427605
```

输出：

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-application/
MyRobotApplication/1551203427605",
  "name": "MyRobotApplication",
  "version": "1",
  "sources": [
    {
      "s3Bucket": "my-bucket",
      "s3Key": "my-simulation-application.tar.gz",
      "etag": "00d8a94ff113856688c4fce618ae0f45-94",
      "architecture": "X86_64"
    }
  ],
  "simulationSoftwareSuite": {
    "name": "Gazebo",
    "version": "7"
  },
  "robotSoftwareSuite": {
    "name": "ROS",
    "version": "Kinetic"
  },
  "renderingEngine": {
    "name": "OGRE",
    "version": "1.x"
  },
  "lastUpdatedAt": 1551203853.0,
  "revisionId": "ee753e53-519c-4d37-895d-65e79bcd1914",
  "tags": {}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSimulationApplicationVersion](#)中的。

## create-simulation-application

以下代码示例演示了如何使用 create-simulation-application。

## AWS CLI

### 创建仿真应用程序

此示例创建了一个仿真应用程序。

命令:

```
aws robomaker create-simulation-application --name MyRobotApplication --sources
s3Bucket=my-bucket,s3Key=my-simulation-application.tar.gz,architecture=ARMHF
--robot-software-suite name=ROS,version=Kinetic --simulation-software-suite
name=Gazebo,version=7 --rendering-engine name=OGRE,version=1.x
```

输出:

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-application/
MyRobotApplication/1551203301792",
  "name": "MyRobotApplication",
  "version": "$LATEST",
  "sources": [
    {
      "s3Bucket": "my-bucket",
      "s3Key": "my-simulation-application.tar.gz",
      "architecture": "X86_64"
    }
  ],
  "simulationSoftwareSuite": {
    "name": "Gazebo",
    "version": "7"
  },
  "robotSoftwareSuite": {
    "name": "ROS",
    "version": "Kinetic"
  },
  "renderingEngine": {
    "name": "OGRE",
    "version": "1.x"
  },
  "lastUpdatedAt": 1551203301.0,
  "revisionId": "ee753e53-519c-4d37-895d-65e79bcd1914",
  "tags": {}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSimulationApplication](#)中的。

## create-simulation-job

以下代码示例演示了如何使用 create-simulation-job。

### AWS CLI

#### 创建模拟作业

此示例创建了一个模拟作业。它使用机器人应用程序和仿真应用程序。

命令：

```
aws robomaker create-simulation-job --max-job-duration-
in-seconds 3600 --iam-role arn:aws:iam::111111111111:role/
AWSRoboMakerCloudWatch-154766341-SimulationJobRole-G00BWTQ8YBG6 --robot-
applications application=arn:aws:robomaker:us-west-2:111111111111:robot-application/
MyRobotApplication/1551203485821,launchConfig={packageName=hello_world_robot,launchFile=rota
--simulation-applications application=arn:aws:robomaker:us-
west-2:111111111111:simulation-application/
MySimulationApplication/1551203427605,launchConfig={packageName=hello_world_simulation,launc
--tags Region=North
```

输出：

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-w7m68wpr05h8",
  "status": "Pending",
  "lastUpdatedAt": 1551213837.0,
  "failureBehavior": "Fail",
  "clientRequestToken": "b283ccce-e468-43ee-8642-be76a9d69f15",
  "maxJobDurationInSeconds": 3600,
  "simulationTimeMillis": 0,
  "iamRole": "arn:aws:iam::111111111111:role/MySimulationRole",
  "robotApplications": [
    {
      "application": "arn:aws:robomaker:us-west-2:111111111111:robot-
application/MyRobotApplication/1551203485821",
      "applicationVersion": "$LATEST",
      "launchConfig": {
        "packageName": "hello_world_robot",
```

```
        "launchFile": "rotate.launch"
      }
    ],
    "simulationApplications": [
      {
        "application": "arn:aws:robomaker:us-west-2:111111111111:simulation-
application/MySimulationApplication/1551203427605",
        "applicationVersion": "$LATEST",
        "launchConfig": {
          "packageName": "hello_world_simulation",
          "launchFile": "empty_world.launch"
        }
      }
    ],
    "tags": {
      "Region": "North"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSimulationJob](#)中的。

## delete-fleet

以下代码示例演示了如何使用 delete-fleet。

### AWS CLI

#### 删除舰队

此示例删除舰队。

命令:

```
aws robomaker delete-fleet --fleet arn:aws:robomaker:us-
west-2:111111111111:deployment-fleet/MyFleet/1550771394395
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFleet](#)中的。

## delete-robot-application

以下代码示例演示了如何使用 delete-robot-application。



## AWS CLI

### 删除机器人应用程序

此示例删除了机器人应用程序。

命令:

```
aws robomaker delete-robot-application --application arn:aws:robomaker:us-west-2:111111111111:robot-application/MyRobotApplication/1551203485821
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRobotApplication](#)中的。

## **delete-robot**

以下代码示例演示了如何使用 delete-robot。

## AWS CLI

### 删除机器人

此示例删除了一个机器人。

命令:

```
aws robomaker delete-robot --robot arn:aws:robomaker:us-west-2:111111111111:robot/MyRobot/1540829698778
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRobot](#)中的。

## **delete-simulation-application**

以下代码示例演示了如何使用 delete-simulation-application。

## AWS CLI

### 删除仿真应用程序

此示例删除了一个模拟应用程序。

命令:

```
aws robomaker delete-simulation-application --application arn:aws:robomaker:us-west-2:111111111111:simulation-application/MySimulationApplication/1551203427605
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSimulationApplication](#)中的。

## deregister-robot

以下代码示例演示了如何使用 deregister-robot。

### AWS CLI

从舰队中注销机器人的注册

此示例将机器人从舰队中注销。

命令:

```
aws robomaker deregister-robot --fleet arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/MyFleet/1550771358907 --robot arn:aws:robomaker:us-west-2:111111111111:robot/MyRobot/1550772324398
```

输出:

```
{
  "fleet": "arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/MyFleet/1550771358907",
  "robot": "arn:aws:robomaker:us-west-2:111111111111:robot/MyRobot/1550772324398"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterRobot](#)中的。

## describe-deployment-job

以下代码示例演示了如何使用 describe-deployment-job。

### AWS CLI

描述部署作业

以下describe-deployment-job示例检索有关指定部署任务的详细信息。

```
aws robomaker describe-deployment-job \  
  --job arn:aws:robomaker:us-west-2:111111111111:deployment-job/deployment-  
xl8qssl6pbcn
```

输出：

```
{  
  "arn": "arn:aws:robomaker:us-west-2:111111111111:deployment-job/deployment-  
xl8qssl6pbcn",  
  "fleet": "arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/  
Trek/1539894765711",  
  "status": "InProgress",  
  "deploymentConfig": {  
    "concurrentDeploymentPercentage": 20,  
    "failureThresholdPercentage": 25  
  },  
  "deploymentApplicationConfigs": [  
    {  
      "application": "arn:aws:robomaker:us-west-2:111111111111:robot-  
application/RoboMakerHelloWorldRobot/1546541208251",  
      "applicationVersion": "1",  
      "launchConfig": {  
        "packageName": "hello_world_robot",  
        "launchFile": "rotate.launch"  
      }  
    }  
  ],  
  "createdAt": 1551218369.0,  
  "robotDeploymentSummary": [  
    {  
      "arn": "arn:aws:robomaker:us-west-2:111111111111:robot/  
MyRobot/1540834232469",  
      "deploymentStartTime": 1551218376.0,  
      "status": "Deploying",  
      "progressDetail": {}  
    }  
  ],  
  "tags": {}  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDeploymentJob](#)中的。

## describe-fleet

以下代码示例演示了如何使用 describe-fleet。

### AWS CLI

#### 描述一支舰队

以下describe-fleet示例检索指定舰队的详细信息。

```
aws robomaker describe-fleet \  
  --fleet arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/  
  MyFleet/1550771358907
```

#### 输出：

```
{  
  "name": "MyFleet",  
  "arn": "arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/  
  MyFleet/1539894765711",  
  "robots": [  
    {  
      "arn": "arn:aws:robomaker:us-west-2:111111111111:robot/  
  MyRobot/1540834232469",  
      "createdAt": 1540834232.0  
    },  
    {  
      "arn": "arn:aws:robomaker:us-west-2:111111111111:robot/  
  MyOtherRobot/1540829698778",  
      "createdAt": 1540829698.0  
    }  
  ],  
  "createdAt": 1539894765.0,  
  "lastDeploymentStatus": "Succeeded",  
  "lastDeploymentJob": "arn:aws:robomaker:us-west-2:111111111111:deployment-job/  
  deployment-xl8qssl6pbcn",  
  "lastDeploymentTime": 1551218369.0,  
  "tags": {}  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFleet](#)中的。

## describe-robot-application

以下代码示例演示了如何使用 `describe-robot-application`。

### AWS CLI

#### 描述机器人应用程序

此示例描述了一个机器人应用程序。

#### 命令:

```
aws robomaker describe-robot-application --application arn:aws:robomaker:us-west-2:111111111111:robot-application/MyRobotApplication/1551203485821
```

#### 输出 :

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:robot-application/MyRobotApplication/1551203485821",
  "name": "MyRobotApplication",
  "version": "$LATEST",
  "sources": [
    {
      "s3Bucket": "my-bucket",
      "s3Key": "my-robot-application.tar.gz",
      "architecture": "X86_64"
    }
  ],
  "robotSoftwareSuite": {
    "name": "ROS",
    "version": "Kinetic"
  },
  "revisionId": "e72efe0d-f44f-4333-b604-f6fa5c6bb50b",
  "lastUpdatedAt": 1551203485.0,
  "tags": {}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeRobotApplication](#)中的。

## describe-robot

以下代码示例演示了如何使用 `describe-robot`。

## AWS CLI

### 描述机器人

此示例描述了一个机器人。

命令:

```
aws robomaker describe-robot --robot arn:aws:robomaker:us-west-2:111111111111:robot/MyRobot/1550772324398
```

输出:

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:robot/MyRobot/1550772324398",
  "name": "MyRobot",
  "status": "Available",
  "greengrassGroupId": "0f728a3c-7dbf-4a3e-976d-d16a8360caba",
  "createdAt": 1550772325.0,
  "architecture": "ARMHF",
  "tags": {
    "Region": "East"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeRobot](#)中的。

## describe-simulation-application

以下代码示例演示了如何使用 describe-simulation-application。

### AWS CLI

#### 描述仿真应用程序

此示例描述了一个仿真应用程序。

命令:

```
aws robomaker describe-simulation-application --application arn:aws:robomaker:us-west-2:111111111111:simulation-application/MySimulationApplication/1551203427605
```

输出：

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-application/
MySimulationApplication/1551203427605",
  "name": "MySimulationApplication",
  "version": "$LATEST",
  "sources": [
    {
      "s3Bucket": "my-bucket",
      "s3Key": "my-simulation-application.tar.gz",
      "architecture": "X86_64"
    }
  ],
  "simulationSoftwareSuite": {
    "name": "Gazebo",
    "version": "7"
  },
  "robotSoftwareSuite": {
    "name": "ROS",
    "version": "Kinetic"
  },
  "renderingEngine": {
    "name": "OGRE",
    "version": "1.x"
  },
  "revisionId": "783674ab-b7b8-42d9-b01f-9373907987e5",
  "lastUpdatedAt": 1551203427.0,
  "tags": {}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSimulationApplication](#)中的。

## describe-simulation-job

以下代码示例演示了如何使用 describe-simulation-job。

### AWS CLI

#### 描述模拟作业

此示例描述了一个模拟作业。

**命令:**

```
aws robomaker describe-simulation-job --job arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-pql32v7pfjy6
```

**输出:**

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-pql32v7pfjy6",
  "status": "Running",
  "lastUpdatedAt": 1551219349.0,
  "failureBehavior": "Continue",
  "clientRequestToken": "a19ec4b5-e50d-3591-33da-c2e593c60615",
  "outputLocation": {
    "s3Bucket": "my-output-bucket",
    "s3Prefix": "output"
  },
  "maxJobDurationInSeconds": 3600,
  "simulationTimeMillis": 0,
  "iamRole": "arn:aws:iam::111111111111:role/MySimulationRole",
  "robotApplications": [
    {
      "application": "arn:aws:robomaker:us-west-2:111111111111:robot-application/MyRobotApplication/1551206341136",
      "applicationVersion": "$LATEST",
      "launchConfig": {
        "packageName": "hello_world_robot",
        "launchFile": "rotate.launch"
      }
    }
  ],
  "simulationApplications": [
    {
      "application": "arn:aws:robomaker:us-west-2:111111111111:simulation-application/MySimulationApplication/1551206347967",
      "applicationVersion": "$LATEST",
      "launchConfig": {
        "packageName": "hello_world_simulation",
        "launchFile": "empty_world.launch"
      }
    }
  ],
  "tags": {}
}
```



```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSimulationJob](#)中的。

## list-deployment-jobs

以下代码示例演示了如何使用 `list-deployment-jobs`。

### AWS CLI

#### 列出部署任务

以下`list-deployment-jobs`示例检索部署任务列表。

```
aws robomaker list-deployment-jobs
```

输出：

```
{
  "deploymentJobs": [
    {
      "arn": "arn:aws:robomaker:us-west-2:111111111111:deployment-job/sim-6293szzm56rv",
      "fleet": "arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/MyFleet/1539894765711",
      "status": "InProgress",
      "deploymentApplicationConfigs": [
        {
          "application": "arn:aws:robomaker:us-west-2:111111111111:robot-application>HelloWorldRobot/1546537110575",
          "applicationVersion": "1",
          "launchConfig": {
            "packageName": "hello_world_robot",
            "launchFile": "rotate.launch",
            "environmentVariables": {
              "ENVIRONMENT": "Desert"
            }
          }
        }
      ]
    },
    "deploymentConfig": {
      "concurrentDeploymentPercentage": 20,

```

```
        "failureThresholdPercentage": 25
      },
      "createdAt": 1550689373.0
    },
    {
      "arn": "arn:aws:robomaker:us-west-2:111111111111:deployment-job/
deployment-4w4g69p25zdb",
      "fleet": "arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/
MyFleet/1539894765711",
      "status": "Pending",
      "deploymentApplicationConfigs": [
        {
          "application": "arn:aws:robomaker:us-west-2:111111111111:robot-
application/AWSRoboMakerHelloWorld-1544562726923_YGHM_sh5M/1544562822877",
          "applicationVersion": "1",
          "launchConfig": {
            "packageName": "fail",
            "launchFile": "fail"
          }
        }
      ],
      "deploymentConfig": {
        "concurrentDeploymentPercentage": 20,
        "failureThresholdPercentage": 25
      },
      "failureReason": "",
      "failureCode": "",
      "createdAt": 1544719763.0
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDeploymentJobs](#)中的。

## list-fleets

以下代码示例演示了如何使用 list-fleets。

### AWS CLI

#### 列出舰队

此示例列出了舰队。最多返回 20 支舰队。

命令:

```
aws robomaker list-fleets --max-items 20
```

输出:

```
{
  "fleetDetails": [
    {
      "name": "Trek",
      "arn": "arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/MyFleet/1539894765711",
      "createdAt": 1539894765.0,
      "lastDeploymentStatus": "Failed",
      "lastDeploymentJob": "arn:aws:robomaker:us-west-2:111111111111:deployment-job/deployment-4w4g69p25zdb",
      "lastDeploymentTime": 1544719763.0
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFleets](#)中的。

## list-robot-applications

以下代码示例演示了如何使用 list-robot-applications。

AWS CLI

列出机器人应用程序

此示例列出了机器人应用程序。结果仅限于 20 个机器人应用程序。

命令:

```
aws robomaker list-robot-applications --max-results 20
```

输出:

```
{
```

```
"robotApplicationSummaries": [  
  {  
    "name": "MyRobot",  
    "arn": "arn:aws:robomaker:us-west-2:111111111111:robot-application/  
MyRobot/1546537110575",  
    "version": "$LATEST",  
    "lastUpdatedAt": 1546540372.0  
  },  
  {  
    "name": "AnotherRobot",  
    "arn": "arn:aws:robomaker:us-west-2:111111111111:robot-application/  
AnotherRobot/1546541208251",  
    "version": "$LATEST",  
    "lastUpdatedAt": 1546541208.0  
  },  
  {  
    "name": "MySuperRobot",  
    "arn": "arn:aws:robomaker:us-west-2:111111111111:robot-application/  
MySuperRobot/1547663517377",  
    "version": "$LATEST",  
    "lastUpdatedAt": 1547663517.0  
  }  
]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRobotApplications](#)中的。

## list-robots

以下代码示例演示了如何使用 list-robots。

### AWS CLI

#### 列出机器人

此示例列出了机器人。最多返回 20 个机器人。

#### 命令:

```
aws robomaker list-robots --max-results 20
```

#### 输出:

```
{
  "robots": [
    {
      "arn": "arn:aws:robomaker:us-west-2:111111111111:robot/Robot100/1544035373264",
      "name": "Robot100",
      "status": "Available",
      "createdAt": 1544035373.0,
      "architecture": "X86_64"
    },
    {
      "arn": "arn:aws:robomaker:us-west-2:111111111111:robot/Robot101/1542146976587",
      "name": "Robot101",
      "status": "Available",
      "createdAt": 1542146976.0,
      "architecture": "X86_64"
    },
    {
      "arn": "arn:aws:robomaker:us-west-2:111111111111:robot/Robot102/1540834232469",
      "fleetArn": "arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/Trek/1539894765711",
      "status": "Available",
      "createdAt": 1540834232.0,
      "architecture": "X86_64",
      "lastDeploymentJob": "arn:aws:robomaker:us-west-2:111111111111:deployment-job/deployment-jb007b75gl5f",
      "lastDeploymentTime": 1550689533.0
    },
    {
      "arn": "arn:aws:robomaker:us-west-2:111111111111:robot/MyRobot/1540829698778",
      "name": "MyRobot",
      "status": "Registered",
      "createdAt": 1540829698.0,
      "architecture": "X86_64"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRobots](#)中的。

## list-simulation-applications

以下代码示例演示了如何使用 `list-simulation-applications`。

### AWS CLI

列出仿真应用程序

此示例列出了仿真应用程序。最多将返回 20 个仿真应用程序。

命令:

```
aws robomaker list-simulation-applications --max-results 20
```

输出:

```
{
  "simulationApplicationSummaries": [
    {
      "name": "AWSRoboMakerObjectTracker-1548959046124_NPvyfcatq",
      "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-application/AWSRoboMakerObjectTracker-1548959046124_NPvyfcatq/1548959170096",
      "version": "$LATEST",
      "lastUpdatedAt": 1548959170.0
    },
    {
      "name": "RoboMakerHelloWorldSimulation",
      "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-application/RoboMakerHelloWorldSimulation/1546541198985",
      "version": "$LATEST",
      "lastUpdatedAt": 1546541198.0
    },
    {
      "name": "RoboMakerObjectTrackerSimulation",
      "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-application/RoboMakerObjectTrackerSimulation/1545846795615",
      "version": "$LATEST",
      "lastUpdatedAt": 1545847405.0
    },
    {
      "name": "RoboMakerVoiceInteractionSimulation",
      "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-application/RoboMakerVoiceInteractionSimulation/1546537100507",
      "version": "$LATEST",

```

```
    "lastUpdatedAt": 1546540352.0
  },
  {
    "name": "AWSRoboMakerCloudWatch-1547663411642_0LI6D1h6",
    "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-application/
AWSRoboMakerCloudWatch-1547663411642_0LI6D1h6/1547663521470",
    "version": "$LATEST",
    "lastUpdatedAt": 1547663521.0
  },
  {
    "name": "AWSRoboMakerDeepRacer-1545848257672_1YZCaieQ-",
    "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-application/
AWSRoboMakerDeepRacer-1545848257672_1YZCaieQ-/1545848370525",
    "version": "$LATEST",
    "lastUpdatedAt": 1545848370.0
  }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSimulationApplications](#)中的。

## list-simulation-jobs

以下代码示例演示了如何使用 list-simulation-jobs。

### AWS CLI

列出模拟作业

此示例列出了模拟作业。

命令:

```
aws robomaker list-simulation-jobs
```

输出:

```
{
  "simulationJobSummaries": [
    {
      "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-job/
sim-66bbb3gpxm8x",
      "lastUpdatedAt": 1548959178.0,

```

```
    "status": "Completed",
    "simulationApplicationNames": [
      "AWSRoboMakerObjectTracker-1548959046124_NPvyfcatq"
    ],
    "robotApplicationNames": [
      null
    ]
  },
  {
    "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-
b27c4rkrtzcw",
    "lastUpdatedAt": 1543514088.0,
    "status": "Canceled",
    "simulationApplicationNames": [
      "AWSRoboMakerPersonDetection-1543513948280_T8rHW2_lu"
    ],
    "robotApplicationNames": [
      "AWSRoboMakerPersonDetection-1543513948280_EYaMT0mYb"
    ]
  },
  {
    "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-job/
sim-51vxjbzy4q8t",
    "lastUpdatedAt": 1543508858.0,
    "status": "Canceled",
    "simulationApplicationNames": [
      "AWSRoboMakerCloudWatch-1543504747391_1FF9ZQyx6"
    ],
    "robotApplicationNames": [
      "AWSRoboMakerCloudWatch-1543504747391_axbYa3S3K"
    ]
  },
  {
    "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-
kgf1fqxflqbx",
    "lastUpdatedAt": 1543504862.0,
    "status": "Completed",
    "simulationApplicationNames": [
      "AWSRoboMakerCloudWatch-1543504747391_1FF9ZQyx6"
    ],
    "robotApplicationNames": [
      "AWSRoboMakerCloudWatch-1543504747391_axbYa3S3K"
    ]
  },
}
```



```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-
vw81vh061nqt",
  "lastUpdatedAt": 1543441430.0,
  "status": "Completed",
  "simulationApplicationNames": [
    "AWSRoboMakerHelloWorld-1543437372341__yb_Jg961"
  ],
  "robotApplicationNames": [
    "AWSRoboMakerHelloWorld-1543437372341_lNbmKHvs9"
  ]
},
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-
txy5ypxmhz84",
  "lastUpdatedAt": 1543437488.0,
  "status": "Completed",
  "simulationApplicationNames": [
    "AWSRoboMakerHelloWorld-1543437372341__yb_Jg961"
  ],
  "robotApplicationNames": [
    "AWSRoboMakerHelloWorld-1543437372341_lNbmKHvs9"
  ]
}
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSimulationJobs](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出资源的标签

此示例列出了 AWS RoboMaker 资源的标签。

命令:

```
aws robomaker list-tags-for-resource --resource-arn "arn:aws:robomaker:us-
west-2:111111111111:robot/Robby_the_Robot/1544035373264"
```

输出：

```
{
  "tags": {
    "Region": "North",
    "Stage": "Initial"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## register-robot

以下代码示例演示了如何使用 register-robot。

### AWS CLI

#### 注册机器人

此示例将机器人注册到舰队。

命令：

```
aws robomaker register-robot --fleet arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/MyFleet/1550771358907 --robot arn:aws:robomaker:us-west-2:111111111111:robot/MyRobot/1550772324398
```

输出：

```
{
  "fleet": "arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/MyFleet/1550771358907",
  "robot": "arn:aws:robomaker:us-west-2:111111111111:robot/MyRobot/1550772324398"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterRobot](#)中的。

## restart-simulation-job

以下代码示例演示了如何使用 restart-simulation-job。

## AWS CLI

### 重启模拟

此示例重新启动模拟。

命令:

```
aws robomaker restart-simulation-job --job arn:aws:robomaker:us-west-2:111111111111:simulation-job/sim-t6rdgt70mftr
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RestartSimulationJob](#)中的。

## sync-deployment-job

以下代码示例演示了如何使用 sync-deployment-job。

## AWS CLI

### 同步部署作业

此示例同步部署作业。

命令:

```
aws robomaker sync-deployment-job --fleet arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/Trek/1539894765711
```

输出:

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:deployment-job/
deployment-09ccxs3tlfms",
  "fleet": "arn:aws:robomaker:us-west-2:111111111111:deployment-fleet/
MyFleet/1539894765711",
  "status": "Pending",
  "deploymentConfig": {
    "concurrentDeploymentPercentage": 20,
    "failureThresholdPercentage": 25
  },
  "deploymentApplicationConfigs": [
```

```
{
  "application": "arn:aws:robomaker:us-west-2:111111111111:robot-
application/MyRobotApplication/1546541208251",
  "applicationVersion": "1",
  "launchConfig": {
    "packageName": "hello_world_simulation",
    "launchFile": "empty_world.launch"
  }
},
"createdAt": 1551286954.0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SyncDeploymentJob](#)中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

为资源添加标签

此示例为资源添加了标签。它附加了两个标签：“区域”和“舞台”。

命令:

```
aws robomaker tag-resource --resource-arn "arn:aws:robomaker:us-
west-2:111111111111:robot/MyRobot/1544035373264" --tags Region=North,Stage=Initial
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

取消对资源的标记

此示例从资源中删除标签。它会移除“区域”标签。

**命令:**

```
aws robomaker untag-resource --resource-arn "arn:aws:robomaker:us-west-2:111111111111:robot/MyRobot/1544035373264" --tag-keys Region
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

**update-robot-application**

以下代码示例演示了如何使用 update-robot-application。

**AWS CLI****更新机器人应用程序**

此示例更新机器人应用程序。

**命令:**

```
aws robomaker update-robot-application --application arn:aws:robomaker:us-west-2:111111111111:robot-application/MyRobotApplication/1551203485821 --sources s3Bucket=my-bucket,s3Key=my-robot-application.tar.gz,architecture=X86_64 --robot-software-suite name=ROS,version=Kinetic
```

**输出:**

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:robot-application/MyRobotApplication/1551203485821",
  "name": "MyRobotApplication",
  "version": "$LATEST",
  "sources": [
    {
      "s3Bucket": "my-bucket",
      "s3Key": "my-robot-application.tar.gz",
      "architecture": "X86_64"
    }
  ],
  "robotSoftwareSuite": {
    "name": "ROS",
    "version": "Kinetic"
  }
}
```

```
  },
  "lastUpdatedAt": 1551287993.0,
  "revisionId": "20b5e331-24fd-4504-8b8c-531afe5f4c94"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRobotApplication](#)中的。

## update-simulation-application

以下代码示例演示了如何使用 update-simulation-application。

### AWS CLI

#### 更新仿真应用程序

此示例更新了仿真应用程序。

命令:

```
aws robomaker update-simulation-application --application
arn:aws:robomaker:us-west-2:111111111111:simulation-application/
MySimulationApplication/1551203427605 --sources s3Bucket=my-bucket,s3Key=my-
simulation-application.tar.gz,architecture=X86_64 --robot-software-suite
name=ROS,version=Kinetic --simulation-software-suite name=Gazebo,version=7 --
rendering-engine name=OGRE,version=1.x
```

输出:

```
{
  "arn": "arn:aws:robomaker:us-west-2:111111111111:simulation-application/
MySimulationApplication/1551203427605",
  "name": "MySimulationApplication",
  "version": "$LATEST",
  "sources": [
    {
      "s3Bucket": "my-bucket",
      "s3Key": "my-simulation-application.tar.gz",
      "architecture": "X86_64"
    }
  ],
  "simulationSoftwareSuite": {
    "name": "Gazebo",
```

```
    "version": "7"
  },
  "robotSoftwareSuite": {
    "name": "ROS",
    "version": "Kinetic"
  },
  "renderingEngine": {
    "name": "OGRE",
    "version": "1.x"
  },
  "lastUpdatedAt": 1551289361.0,
  "revisionId": "4a22cb5d-93c5-4cef-9311-52bdd119b79e"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateSimulationApplication](#)中的。

## 使用 Route 53 示例 AWS CLI

以下代码示例向您展示了如何在 Route 53 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **change-resource-record-sets**

以下代码示例演示了如何使用 change-resource-record-sets。

### AWS CLI

#### 创建、更新或删除资源记录集

以下 `change-resource-record-sets` 命令使用文件中的 `hosted-zone-idZ1R8UBAEXAMPLE` 和 JSON 格式的配置创建资源记录集：`C:\awscli\route53\change-resource-record-sets.json`

```
aws route53 change-resource-record-sets --hosted-zone-id Z1R8UBAEXAMPLE --change-batch file://C:\awscli\route53\change-resource-record-sets.json
```

有关更多信息，请参阅《亚马逊 Route 53 API 参考》 `ChangeResourceRecordSets` 中的 POST。

JSON 文件中的配置取决于您要创建的资源记录集的类型：

`BasicWeightedAliasWeighted AliasLatencyLatency AliasFailoverFailover` 别名

基本语法：

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "TTL": time to live in seconds,
        "ResourceRecords": [
          {
            "Value": "applicable value for the record type"
          },
          {...}
        ]
      }
    },
    {...}
  ]
}
```

加权语法：

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
```



```

"ResourceRecordSet": {
  "Name": "DNS domain name",
  "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
  "SetIdentifier": "unique description for this resource record set",
  "Weight": value between 0 and 255,
  "TTL": time to live in seconds,
  "ResourceRecords": [
    {
      "Value": "applicable value for the record type"
    },
    {...}
  ],
  "HealthCheckId": "optional ID of an Amazon Route 53 health check"
}
},
{...}
]
}

```

#### 别名语法：

```

{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "AliasTarget": {
          "HostedZoneId": "hosted zone ID for your CloudFront distribution, Amazon
          S3 bucket, Elastic Load Balancing load balancer, or Amazon Route 53 hosted zone",
          "DNSName": "DNS domain name for your CloudFront distribution, Amazon S3
          bucket, Elastic Load Balancing load balancer, or another resource record set in
          this hosted zone",
          "EvaluateTargetHealth": true|false
        },
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}

```

## 加权别名语法：

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Weight": value between 0 and 255,
        "AliasTarget": {
          "HostedZoneId": "hosted zone ID for your CloudFront distribution, Amazon
          S3 bucket, Elastic Load Balancing load balancer, or Amazon Route 53 hosted zone",
          "DNSName": "DNS domain name for your CloudFront distribution, Amazon S3
          bucket, Elastic Load Balancing load balancer, or another resource record set in
          this hosted zone",
          "EvaluateTargetHealth": true|false
        },
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}
```

## 延迟语法：

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Region": "Amazon EC2 region name",
        "TTL": time to live in seconds,
        "ResourceRecords": [
          {
            "Value": "applicable value for the record type"
          }
        ]
      }
    }
  ]
}
```

```

    },
    {...}
  ],
  "HealthCheckId": "optional ID of an Amazon Route 53 health check"
}
},
{...}
]
}

```

### 延迟别名语法：

```

{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Region": "Amazon EC2 region name",
        "AliasTarget": {
          "HostedZoneId": "hosted zone ID for your CloudFront distribution, Amazon
          S3 bucket, Elastic Load Balancing load balancer, or Amazon Route 53 hosted zone",
          "DNSName": "DNS domain name for your CloudFront distribution, Amazon S3
          bucket, Elastic Load Balancing load balancer, or another resource record set in
          this hosted zone",
          "EvaluateTargetHealth": true|false
        },
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}

```

### 故障转移语法：

```

{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {

```

```

    "Action": "CREATE"|"DELETE"|"UPSERT",
    "ResourceRecordSet": {
      "Name": "DNS domain name",
      "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
      "SetIdentifier": "unique description for this resource record set",
      "Failover": "PRIMARY" | "SECONDARY",
      "TTL": time to live in seconds,
      "ResourceRecords": [
        {
          "Value": "applicable value for the record type"
        },
        {...}
      ],
      "HealthCheckId": "ID of an Amazon Route 53 health check"
    }
  },
  {...}
]
}

```

#### 故障转移别名语法：

```

{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Failover": "PRIMARY" | "SECONDARY",
        "AliasTarget": {
          "HostedZoneId": "hosted zone ID for your CloudFront distribution, Amazon
          S3 bucket, Elastic Load Balancing load balancer, or Amazon Route 53 hosted zone",
          "DNSName": "DNS domain name for your CloudFront distribution, Amazon S3
          bucket, Elastic Load Balancing load balancer, or another resource record set in
          this hosted zone",
          "EvaluateTargetHealth": true|false
        },
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}

```

```
{...}  
]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ChangeResourceRecordSets](#)中的。

## change-tags-for-resource

以下代码示例演示了如何使用 `change-tags-for-resource`。

### AWS CLI

以下命令将名为 `owner` 的标签添加到由 ID 指定的运行状况检查资源：

```
aws route53 change-tags-for-resource --resource-type healthcheck --resource-id  
6233434j-18c1-34433-ba8e-3443434 --add-tags Key=owner,Value=myboss
```

以下命令 `owner` 从 ID 指定的托管区域资源中移除名为的标签：

```
aws route53 change-tags-for-resource --resource-type hostedzone --resource-id  
Z1523434445 --remove-tag-keys owner
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ChangeTagsForResource](#)中的。

## create-health-check

以下代码示例演示了如何使用 `create-health-check`。

### AWS CLI

创建运行状况检查

以下 `create-health-check` 命令使用文件中的调用者引用 `2014-04-01-18:47` 和 JSON 格式的配置创建运行状况检查： `C:\awscli\route53\create-health-check.json`

```
aws route53 create-health-check --caller-reference 2014-04-01-18:47 --health-check-  
config file://C:\awscli\route53\create-health-check.json
```

JSON 语法：

```
{
```

```

  "IPAddress": "IP address of the endpoint to check",
  "Port": port on the endpoint to check--required when Type is "TCP",
  "Type": "HTTP"|"HTTPS"|"HTTP_STR_MATCH"|"HTTPS_STR_MATCH"|"TCP",
  "ResourcePath": "path of the file that you want Amazon Route 53 to request--all
Types except TCP",
  "FullyQualifiedDomainName": "domain name of the endpoint to check--all Types
except TCP",
  "SearchString": "if Type is HTTP_STR_MATCH or HTTPS_STR_MATCH, the string to
search for in the response body from the specified resource",
  "RequestInterval": 10 | 30,
  "FailureThreshold": integer between 1 and 10
}

```

要将运行状况检查添加到 Route 53 资源记录集，请使用 `change-resource-record-sets` 命令。

有关更多信息，请参阅《亚马逊 Route 53 开发者指南》中的 Amazon Route 53 健康检查和 DNS 故障转移。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateHealthCheck](#) 中的。

## create-hosted-zone

以下代码示例演示了如何使用 `create-hosted-zone`。

### AWS CLI

#### 创建托管区域

以下 `create-hosted-zone` 命令添加使用呼叫者引 `example.com` 命名的托管区域 `2014-04-01-18:47`。可选注释包含一个空格，因此必须用引号将其括起来：

```
aws route53 create-hosted-zone --name example.com --caller-reference
2014-04-01-18:47 --hosted-zone-config Comment="command-line version"
```

有关更多信息，请参阅《Amazon Route 53 开发者指南》中的使用托管区域。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateHostedZone](#) 中的。

## delete-health-check

以下代码示例演示了如何使用 `delete-health-check`。

## AWS CLI

### 删除运行状况检查

以下delete-health-check命令会删除带有health-check-id为的运行状况检查e75b48d9-547a-4c3d-88a5-ae4002397608：

```
aws route53 delete-health-check --health-check-id e75b48d9-547a-4c3d-88a5-ae4002397608
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteHealthCheck](#)中的。

## delete-hosted-zone

以下代码示例演示了如何使用 delete-hosted-zone。

### AWS CLI

#### 删除托管区域

以下delete-hosted-zone命令删除带有以下值的id托管区域Z36KTIQEXAMPLE：

```
aws route53 delete-hosted-zone --id Z36KTIQEXAMPLE
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteHostedZone](#)中的。

## get-change

以下代码示例演示了如何使用 get-change。

### AWS CLI

#### 获取资源记录集更改的状态

以下get-change命令获取请求的状态和其他信息，该change-resource-record-sets请求的状态和其它信息Id为/change/CWPIK4URU2I5S：

```
aws route53 get-change --id /change/CWPIK4URU2I5S
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetChange](#)中的。

## get-health-check

以下代码示例演示了如何使用 `get-health-check`。

### AWS CLI

获取有关运行状况检查的信息

以下`get-health-check`命令获取有关具有以下值的运行状况检查的信息  
02ec8401-9879-4259-91fa-04e66d094674 : health-check-id

```
aws route53 get-health-check --health-check-id 02ec8401-9879-4259-91fa-04e66d094674
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetHealthCheck](#)中的。

## get-hosted-zone

以下代码示例演示了如何使用 `get-hosted-zone`。

### AWS CLI

获取有关托管区域的信息

以下`get-hosted-zone`命令使用以下命令获取有关托管区域的信息  
Z1R8UBAEXAMPLE : id

```
aws route53 get-hosted-zone --id Z1R8UBAEXAMPLE
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetHostedZone](#)中的。

## list-health-checks

以下代码示例演示了如何使用 `list-health-checks`。

### AWS CLI

列出与当前 AWS 账户关联的运行状况检查

以下`list-health-checks`命令列出了与当前 AWS 账户关联的前 100 个运行状况检查的详细信息。  
:



```
aws route53 list-health-checks
```

如果您有超过 100 个运行状况检查，或者您想将它们按小于 100 的群组列出，请包含 `--max-items` 参数。例如，要一次列出一个运行状况检查，请使用以下命令：

```
aws route53 list-health-checks --max-items 1
```

要查看下一次运行状况检查，请 `NextToken` 从对上一个命令的响应中取值并将其包含在 `--starting-token` 参数中，例如：

```
aws route53 list-health-checks --max-items 1 --starting-token Z3M3LMPEXAMPLE
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListHealthChecks](#) 中的。

## list-hosted-zones-by-name

以下代码示例演示了如何使用 `list-hosted-zones-by-name`。

### AWS CLI

以下命令按域名排序最多列出 100 个托管区域：

```
aws route53 list-hosted-zones-by-name
```

输出：

```
{
  "HostedZones": [
    {
      "ResourceRecordSetCount": 2,
      "CallerReference": "test20150527-2",
      "Config": {
        "Comment": "test2",
        "PrivateZone": false
      },
      "Id": "/hostedzone/Z119WBBTVP5WFX",
      "Name": "2.example.com."
    },
    {
      "ResourceRecordSetCount": 2,
```

```
    "CallerReference": "test20150527-1",
    "Config": {
      "Comment": "test",
      "PrivateZone": false
    },
    "Id": "/hostedzone/Z3P5QSUBK4P0TI",
    "Name": "www.example.com."
  }
],
"IsTruncated": false,
"MaxItems": "100"
}
```

以下命令列出按名称排序的托管区域，开头www.example.com为：

```
aws route53 list-hosted-zones-by-name --dns-name www.example.com
```

输出：

```
{
  "HostedZones": [
    {
      "ResourceRecordSetCount": 2,
      "CallerReference": "mwunderl20150527-1",
      "Config": {
        "Comment": "test",
        "PrivateZone": false
      },
      "Id": "/hostedzone/Z3P5QSUBK4P0TI",
      "Name": "www.example.com."
    }
  ],
  "DNSName": "www.example.com",
  "IsTruncated": false,
  "MaxItems": "100"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListHostedZonesByName](#)中的。

## list-hosted-zones

以下代码示例演示了如何使用 list-hosted-zones。

## AWS CLI

列出与当前 AWS 账户关联的托管区域

以下 `list-hosted-zones` 命令列出了与当前 AWS 账户关联的前 100 个托管区域的摘要信息。：

```
aws route53 list-hosted-zones
```

如果您有超过 100 个托管区域，或者想要将它们按小于 100 的组列出，请包含 `--max-items` 参数。例如，要一次列出一个托管区域，请使用以下命令：

```
aws route53 list-hosted-zones --max-items 1
```

要查看有关下一个托管区域的信息，请从上一个命令的响应中获取 `NextToken` 的值，并将其包含在 `--starting-token` 参数中，例如：

```
aws route53 list-hosted-zones --max-items 1 --starting-token Z3M3LMPEXAMPLE
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListHostedZones](#) 中的。

## list-query-logging-configs

以下代码示例演示了如何使用 `list-query-logging-configs`。

### AWS CLI

列出查询日志配置

以下 `list-query-logging-configs` 示例列出了有关您 AWS 账户中托管区域前 100 个查询日志配置的信息 `Z10X3WQEXAMPLE`。

```
aws route53 list-query-logging-configs \  
  --hosted-zone-id Z10X3WQEXAMPLE
```

输出：

```
{  
  "QueryLoggingConfigs": [  
    {  
      "Id": "964ff34e-ae03-4f06-80a2-9683cexample",
```

```
        "HostedZoneId": "Z10X3WQEXAMPLE",
        "CloudWatchLogsLogGroupArn": "arn:aws:logs:us-east-1:111122223333:log-
group:/aws/route53/example.com:*"
    }
]
}
```

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[记录 DNS 查询](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListQueryLoggingConfigs](#)中的。

## list-resource-record-sets

以下代码示例演示了如何使用 `list-resource-record-sets`。

### AWS CLI

列出托管区域中的资源记录集

以下 `list-resource-record-sets` 命令列出了有关指定托管区域中前 100 个资源记录集的摘要信息。：

```
aws route53 list-resource-record-sets --hosted-zone-id Z2LD58HEXAMPLE
```

如果托管区域包含的资源记录集超过 100 个，或者您想将它们按小于 100 的组列出，请包含 `--max-items` 参数。例如，要逐一列出一个资源记录集，请使用以下命令：

```
aws route53 list-resource-record-sets --hosted-zone-id Z2LD58HEXAMPLE --max-items 1
```

要查看有关托管区域中下一个资源记录集的信息，请 `NextToken` 从对上一个命令的响应中获取的值，并将其包含在 `--starting-token` 参数中，例如：

```
aws route53 list-resource-record-sets --hosted-zone-id Z2LD58HEXAMPLE --max-items 1
--starting-token Z3M3LMPEXAMPLE
```

要查看具有特定名称的所有资源记录集，请使用 `--query` 参数将其过滤掉。例如：

```
aws route53 list-resource-record-sets --hosted-zone-id Z2LD58HEXAMPLE --query
"ResourceRecordSets[?Name == 'example.domain.']"
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListResourceRecordSets](#)中的。

## 使用 Route 53 的域名注册示例 AWS CLI

以下代码示例向您展示了如何在 Route 53 域注册中 AWS Command Line Interface 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **check-domain-availability**

以下代码示例演示了如何使用 `check-domain-availability`。

#### AWS CLI

确定是否可以在 Route 53 上注册域名

以下 `check-domain-availability` 命令返回有关域名 `example.com` 是否可以使用 Route 53 注册的信息。

此命令仅在 `us-east-1` 区域中运行。如果您的默认区域设置为 `us-east-1`，则可以省略该 `region` 参数。

```
aws route53domains check-domain-availability \  
  --region us-east-1 \  
  --domain-name example.com
```

输出：

```
{  
  "Availability": "UNAVAILABLE"  
}
```

Route 53 支持大量顶级域名 (TLD)，例如 .com 和 .jp，但我们并不支持所有可用的顶级域名。如果您检查域名的可用性，而 Route 53 不支持该顶级域名，`check-domain-availability` 则会返回以下消息。

```
An error occurred (UnsupportedTLD) when calling the CheckDomainAvailability
operation: <top-level domain> tld is not supported.
```

有关在 Route 53 注册域名时可以使用的顶级域名列表，请参阅《亚马逊 Route 53 开发者指南》中的“您可以在亚马逊 Route [53 上注册的域名](#)”。有关使用亚马逊 Route 53 注册域名的更多信息，请参阅《亚马逊 Route 53 开发者指南》中的[注册新域名](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CheckDomainAvailability](#) 中的。

## check-domain-transferability

以下代码示例演示了如何使用 `check-domain-transferability`。

### AWS CLI

确定域名是否可以转移到 Route 53

以下 `check-domain-transferability` 命令返回有关您是否可以将域名转移 `example.com` 到 Route 53 的信息。

此命令仅在 `us-east-1` 区域中运行。如果您的默认区域设置为 `us-east-1`，则可以省略该 `region` 参数。

```
aws route53domains check-domain-transferability \
  --region us-east-1 \
  --domain-name example.com
```

输出：

```
{
  "Transferability": {
    "Transferable": "UNTRANSFERABLE"
  }
}
```

有关更多信息，请参阅《亚马逊 [Route 53 开发者指南](#)》中的[将域名注册转移到亚马逊 Route 53](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CheckDomainTransferability](#) 中的。

## delete-tags-for-domain

以下代码示例演示了如何使用 delete-tags-for-domain。

### AWS CLI

#### 删除域名的标签

以下delete-tags-for-domain命令从指定域中删除三个标签。请注意，您只能指定标签键，而不是标签值。

此命令仅在us-east-1区域中运行。如果您的默认区域设置为us-east-1，则可以省略该region参数。

```
aws route53domains delete-tags-for-domain \  
  --region us-east-1 \  
  --domain-name example.com \  
  --tags-to-delete accounting-key hr-key engineering-key
```

此命令不生成任何输出。

要确认标签已删除，可以运行[list-tags-for-domain](#)。有关更多信息，请参阅《[亚马逊 Route 53 开发者指南](#)》中的[为亚马逊 Route 53 资源添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTagsForDomain](#)中的。

## disable-domain-auto-renew

以下代码示例演示了如何使用 disable-domain-auto-renew。

### AWS CLI

#### 禁用域名的自动续订

以下disable-domain-auto-renew命令将 Route 53 配置为example.com在域名注册到期之前不自动续订该域。

此命令仅在us-east-1区域中运行。如果您的默认区域设置为us-east-1，则可以省略该region参数。

```
aws route53domains disable-domain-auto-renew \  
  --region us-east-1 \  
  --domain-name example.com
```

```
--domain-name example.com
```

此命令不生成任何输出。

要确认设置已更改，可以运行[get-domain-detail](#)。如果禁用了自动续订，则的值AutoRenew为False。有关自动续订的更多信息，请参阅《亚马逊 Route 53 开发者指南》中的“续订域名注册 < <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/domain-renew.html>”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableDomainAutoRenew](#)中的。

## disable-domain-transfer-lock

以下代码示例演示了如何使用 disable-domain-transfer-lock。

### AWS CLI

#### 禁用域名的转移锁定

以下disable-domain-transfer-lock命令可移除域名的转移锁定，example.com以便可以将域名转移到其他注册商。此命令会更改clientTransferProhibited状态。

此命令仅在us-east-1区域中运行。如果您的默认区域设置为us-east-1，则可以省略该region参数。

```
aws route53domains disable-domain-transfer-lock \  
  --region us-east-1 \  
  --domain-name example.com
```

输出：

```
{  
  "OperationId": "3f28e0ac-126a-4113-9048-cc930example"  
}
```

要确认转移锁已更改，可以运行[get-domain-detail](#)。禁用转移锁时，的值StatusList不包括clientTransferProhibited。

有关转移流程的更多信息，请参阅《亚马逊 [Route 53 开发者指南](#)》中的[将域名从 Amazon Route 53 转移到其他注册商](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableDomainTransferLock](#)中的。



## enable-domain-auto-renew

以下代码示例演示了如何使用 `enable-domain-auto-renew`。

### AWS CLI

启用域名的自动续订

以下 `enable-domain-auto-renew` 命令将 Route 53 配置为 `example.com` 在域名注册到期之前自动续订该域。

此命令仅在 `us-east-1` 区域中运行。如果您的默认区域设置为 `us-east-1`，则可以省略该 `region` 参数。

```
aws route53domains enable-domain-auto-renew \  
  --region us-east-1 \  
  --domain-name example.com
```

此命令不生成任何输出。要确认设置已更改，可以运行 [get-domain-detail](#)。如果启用了自动续订，则的值 `AutoRenew` 为 `True`。

有关自动续订的更多信息，请参阅《亚马逊 Route 53 开发者指南》中的“续订域名注册 < <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/domain-renew.html>”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [EnableDomainAutoRenew](#) 中的。

## enable-domain-transfer-lock

以下代码示例演示了如何使用 `enable-domain-transfer-lock`。

### AWS CLI

在域名上启用转移锁定

以下 `enable-domain-transfer-lock` 命令会锁定指定的域，使其无法转移到其他注册商。此命令会更改 `clientTransferProhibited` 状态。

此命令仅在 `us-east-1` 区域中运行。如果您的默认区域设置为 `us-east-1`，则可以省略该 `region` 参数。

```
aws route53domains enable-domain-transfer-lock \  
  --region us-east-1 \  
  --domain-name example.com
```

```
--domain-name example.com
```

输出：

```
{
  "OperationId": "3f28e0ac-126a-4113-9048-cc930example"
}
```

要确认转移锁已更改，可以运行[get-domain-detail](#)。启用转移锁定后，的值StatusList包括clientTransferProhibited。

有关转移流程的更多信息，请参阅《亚马逊 [Route 53 开发者指南](#)》中的[将域名从 Amazon Route 53 转移到其他注册商](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableDomainTransferLock](#)中的。

## get-contact-reachability-status

以下代码示例演示了如何使用 get-contact-reachability-status。

### AWS CLI

确定注册人联系人是否已回复确认电子邮件

以下get-contact-reachability-status命令返回有关指定域名的注册人联系人是否已回复确认电子邮件的信息。

此命令仅在us-east-1区域中运行。如果您的默认区域设置为us-east-1，则可以省略该region参数。

```
aws route53domains get-contact-reachability-status \
  --region us-east-1 \
  --domain-name example.com
```

输出：

```
{
  "domainName": "example.com",
  "status": "DONE"
}
```

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[重新发送授权和确认电子邮件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetContactReachabilityStatus](#)中的。

## get-domain-detail

以下代码示例演示了如何使用 `get-domain-detail`。

### AWS CLI

获取有关指定域的详细信息

以下`get-domain-detail`命令显示有关指定域的详细信息。

此命令仅在`us-east-1`区域中运行。如果您的默认区域设置为`us-east-1`，则可以省略该`region`参数。

```
aws route53domains get-domain-detail \
  --region us-east-1 \
  --domain-name example.com
```

输出：

```
{
  "DomainName": "example.com",
  "Nameservers": [
    {
      "Name": "ns-2048.awsdns-64.com",
      "GlueIps": []
    },
    {
      "Name": "ns-2049.awsdns-65.net",
      "GlueIps": []
    },
    {
      "Name": "ns-2050.awsdns-66.org",
      "GlueIps": []
    },
    {
      "Name": "ns-2051.awsdns-67.co.uk",
      "GlueIps": []
    }
  ],
  "AutoRenew": true,
  "AdminContact": {
```

```
    "FirstName": "Saanvi",
    "LastName": "Sarkar",
    "ContactType": "COMPANY",
    "OrganizationName": "Example",
    "AddressLine1": "123 Main Street",
    "City": "Anytown",
    "State": "WA",
    "CountryCode": "US",
    "ZipCode": "98101",
    "PhoneNumber": "+1.8005551212",
    "Email": "ssarkar@example.com",
    "ExtraParams": []
  },
  "RegistrantContact": {
    "FirstName": "Alejandro",
    "LastName": "Rosalez",
    "ContactType": "COMPANY",
    "OrganizationName": "Example",
    "AddressLine1": "123 Main Street",
    "City": "Anytown",
    "State": "WA",
    "CountryCode": "US",
    "ZipCode": "98101",
    "PhoneNumber": "+1.8005551212",
    "Email": "arosalez@example.com",
    "ExtraParams": []
  },
  "TechContact": {
    "FirstName": "Wang",
    "LastName": "Xiulan",
    "ContactType": "COMPANY",
    "OrganizationName": "Example",
    "AddressLine1": "123 Main Street",
    "City": "Anytown",
    "State": "WA",
    "CountryCode": "US",
    "ZipCode": "98101",
    "PhoneNumber": "+1.8005551212",
    "Email": "wxiulan@example.com",
    "ExtraParams": []
  },
  "AdminPrivacy": true,
  "RegistrantPrivacy": true,
  "TechPrivacy": true,
```

```
"RegistrarName": "Amazon Registrar, Inc.",
"WhoIsServer": "whois.registrar.amazon.com",
"RegistrarUrl": "http://registrar.amazon.com",
"AbuseContactEmail": "abuse@registrar.amazon.com",
"AbuseContactPhone": "+1.2062661000",
"CreationDate": 1444934889.601,
"ExpirationDate": 1602787689.0,
"StatusList": [
  "clientTransferProhibited"
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDomainDetail](#)中的。

## get-domain-suggestions

以下代码示例演示了如何使用 `get-domain-suggestions`。

### AWS CLI

#### 获取建议的域名列表

以下`get-domain-suggestions`命令根据域名显示建议的域名列表`example.com`。该响应仅包含可用的域名。此命令仅在`us-east-1`区域中运行。如果您的默认区域设置为`us-east-1`，则可以省略该`region`参数。

```
aws route53domains get-domain-suggestions \
  --region us-east-1 \
  --domain-name example.com \
  --suggestion-count 10 \
  --only-available
```

输出：

```
{
  "SuggestionsList": [
    {
      "DomainName": "egzaampal.com",
      "Availability": "AVAILABLE"
    },
    {
```

```
    "DomainName": "examplelaw.com",
    "Availability": "AVAILABLE"
  },
  {
    "DomainName": "examplehouse.net",
    "Availability": "AVAILABLE"
  },
  {
    "DomainName": "homeexample.net",
    "Availability": "AVAILABLE"
  },
  {
    "DomainName": "examplelist.com",
    "Availability": "AVAILABLE"
  },
  {
    "DomainName": "examplenews.net",
    "Availability": "AVAILABLE"
  },
  {
    "DomainName": "officeexample.com",
    "Availability": "AVAILABLE"
  },
  {
    "DomainName": "exampleworld.com",
    "Availability": "AVAILABLE"
  },
  {
    "DomainName": "exampleart.com",
    "Availability": "AVAILABLE"
  }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDomainSuggestions](#)中的。

## get-operation-detail

以下代码示例演示了如何使用 `get-operation-detail`。

### AWS CLI

获取操作的当前状态

某些域名注册操作异步运行，并在完成之前返回响应。这些操作会返回一个操作 ID，您可以使用它来获取当前状态。以下 `get-operation-detail` 命令返回指定操作的状态。

此命令仅在 `us-east-1` 区域中运行。如果您的默认区域设置为 `us-east-1`，则可以省略该 `region` 参数。

```
aws route53domains get-operation-detail \  
  --region us-east-1 \  
  --operation-id edbd8d63-7fe7-4343-9bc5-54033example
```

输出：

```
{  
  "OperationId": "edbd8d63-7fe7-4343-9bc5-54033example",  
  "Status": "SUCCESSFUL",  
  "DomainName": "example.com",  
  "Type": "DOMAIN_LOCK",  
  "SubmittedDate": 1573749367.864  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetOperationDetail](#) 中的。

## list-domains

以下代码示例演示了如何使用 `list-domains`。

### AWS CLI

列出使用当前 AWS 账户注册的域名

以下 `list-domains` 命令列出了有关在当前 AWS 账户中注册的域名的摘要信息。

此命令仅在 `us-east-1` 区域中运行。如果您的默认区域设置为 `us-east-1`，则可以省略该 `region` 参数。

```
aws route53domains list-domains  
  --region us-east-1
```

输出：

```
{
  "Domains": [
    {
      "DomainName": "example.com",
      "AutoRenew": true,
      "TransferLock": true,
      "Expiry": 1602712345.0
    },
    {
      "DomainName": "example.net",
      "AutoRenew": true,
      "TransferLock": true,
      "Expiry": 1602723456.0
    },
    {
      "DomainName": "example.org",
      "AutoRenew": true,
      "TransferLock": true,
      "Expiry": 1602734567.0
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDomains](#)中的。

## list-operations

以下代码示例演示了如何使用 list-operations。

### AWS CLI

列出返回操作 ID 的操作的状态

一些域名注册操作异步运行，并在完成之前返回响应。这些操作会返回一个操作 ID，您可以使用它来获取当前状态。以下list-operations命令列出了有关当前域名注册操作的摘要信息，包括状态。

此命令仅在us-east-1区域中运行。如果您的默认区域设置为us-east-1，则可以省略该region参数。

```
aws route53domains list-operations
```



```
--region us-east-1
```

输出：

```
{
  "Operations": [
    {
      "OperationId": "aab9822f-1da0-4bf3-8a15-fd4e0example",
      "Status": "SUCCESSFUL",
      "Type": "DOMAIN_LOCK",
      "SubmittedDate": 1455321739.986
    },
    {
      "OperationId": "c24379ed-76be-42f8-bdad-9379bexample",
      "Status": "SUCCESSFUL",
      "Type": "UPDATE_NAMESERVER",
      "SubmittedDate": 1468960475.109
    },
    {
      "OperationId": "f47e1297-ef9e-4c2b-ae1e-a5fcbexample",
      "Status": "SUCCESSFUL",
      "Type": "RENEW_DOMAIN",
      "SubmittedDate": 1473561835.943
    },
    {
      "OperationId": "75584f23-b15f-459e-aed7-dc6f5example",
      "Status": "SUCCESSFUL",
      "Type": "UPDATE_DOMAIN_CONTACT",
      "SubmittedDate": 1547501003.41
    }
  ]
}
```

输出包括所有返回操作 ID 的操作，以及您在使用当前 AWS 账户注册的所有域名上执行的操作。如果您只想获取在指定日期之后提交的操作，则可以包含 `submitted-since` 参数并以 Unix 格式和协调世界时 (UTC) 指定日期。以下命令获取世界标准时间 2020 年 1 月 1 日上午 12:00 之后提交的所有操作的状态。

```
aws route53domains list-operations \
  --submitted-since 1577836800
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListOperations](#) 中的。

## list-tags-for-domain

以下代码示例演示了如何使用 `list-tags-for-domain`。

### AWS CLI

列出域名的标签

以下 `list-tags-for-domain` 命令列出了当前与指定域关联的标签。

此命令仅在 `us-east-1` 区域中运行。如果您的默认区域设置为 `us-east-1`，则可以省略该 `region` 参数。

```
aws route53domains list-tags-for-domain \  
  --region us-east-1 \  
  --domain-name example.com
```

输出：

```
{  
  "TagList": [  
    {  
      "Key": "key1",  
      "Value": "value1"  
    },  
    {  
      "Key": "key2",  
      "Value": "value2"  
    }  
  ]  
}
```

有关更多信息，请参阅《[亚马逊 Route 53 开发者指南](#)》中的[为亚马逊 Route 53 资源添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForDomain](#) 中的。

## register-domain

以下代码示例演示了如何使用 `register-domain`。

### AWS CLI

注册域名

以下`register-domain`命令注册一个域，从 JSON 格式的文件中检索所有参数值。

此命令仅在`us-east-1`区域中运行。如果您的默认区域设置为`us-east-1`，则可以省略该`region`参数。

```
aws route53domains register-domain \  
  --region us-east-1 \  
  --cli-input-json file://register-domain.json
```

`register-domain.json` 的内容：

```
{  
  "DomainName": "example.com",  
  "DurationInYears": 1,  
  "AutoRenew": true,  
  "AdminContact": {  
    "FirstName": "Martha",  
    "LastName": "Rivera",  
    "ContactType": "PERSON",  
    "OrganizationName": "Example",  
    "AddressLine1": "1 Main Street",  
    "City": "Anytown",  
    "State": "WA",  
    "CountryCode": "US",  
    "ZipCode": "98101",  
    "PhoneNumber": "+1.8005551212",  
    "Email": "mrivera@example.com"  
  },  
  "RegistrantContact": {  
    "FirstName": "Li",  
    "LastName": "Juan",  
    "ContactType": "PERSON",  
    "OrganizationName": "Example",  
    "AddressLine1": "1 Main Street",  
    "City": "Anytown",  
    "State": "WA",  
    "CountryCode": "US",  
    "ZipCode": "98101",  
    "PhoneNumber": "+1.8005551212",  
    "Email": "ljuan@example.com"  
  },  
  "TechContact": {  
    "FirstName": "Mateo",
```

```
    "LastName": "Jackson",
    "ContactType": "PERSON",
    "OrganizationName": "Example",
    "AddressLine1": "1 Main Street",
    "City": "Anytown",
    "State": "WA",
    "CountryCode": "US",
    "ZipCode": "98101",
    "PhoneNumber": "+1.8005551212",
    "Email": "mjackson@example.com"
  },
  "PrivacyProtectAdminContact": true,
  "PrivacyProtectRegistrantContact": true,
  "PrivacyProtectTechContact": true
}
```

输出：

```
{
  "OperationId": "b114c44a-9330-47d1-a6e8-a0b11example"
}
```

要确认操作成功，可以运行 `get-operation-detail`。有关更多信息，请参阅[get-operation-detail](#)。

有关更多信息，请参阅 Amazon Route 53 开发者指南[中的注册新域名](#)。

有关哪些顶级域名 (TLD) 需要值 `ExtraParams` 以及有效值的信息，请参阅[ExtraParam](#) 《Amazon Route 53 API 参考》。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RegisterDomain](#) 中的。

## renew-domain

以下代码示例演示了如何使用 `renew-domain`。

### AWS CLI

#### 续订域名

以下 `renew-domain` 命令将指定域名续订五年。要获取的值 `current-expiry-year`，请使用 `get-domain-detail` 命令，然后 `ExpirationDate` 从 Unix 格式转换的值。

此命令仅在us-east-1区域中运行。如果您的默认区域设置为us-east-1，则可以省略该region参数。

```
aws route53domains renew-domain \  
  --region us-east-1 \  
  --domain-name example.com \  
  --duration-in-years 5 \  
  --current-expiry-year 2020
```

输出：

```
{  
  "OperationId": "3f28e0ac-126a-4113-9048-cc930example"  
}
```

要确认操作成功，可以运行get-operation-detail。有关更多信息，请参阅[get-operation-detail](#)。

每个顶级域名 (TLD) (例如.com 或.org) 的注册管理机构控制您可以续订域名的最大年数。要获得域名的最长续订期限，请参阅《亚马逊 Route 53 开发者指南》中“[您可以在亚马逊 Route 53 注册的域名](#)”中的 TLD 的“注册和续订期限”部分。

有关更多信息，请参阅 Amazon Route 53 开发者指南[中的续订域名注册](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RenewDomain](#)中的。

## resend-contact-reachability-email

以下代码示例演示了如何使用 resend-contact-reachability-email。

### AWS CLI

要将确认电子邮件重新发送到注册人的当前电子邮件地址，请联系人

以下resend-contact-reachability-email命令将确认电子邮件重新发送到 example.com 域名注册人联系人的当前电子邮件地址。

此命令仅在us-east-1区域中运行。如果您的默认区域设置为us-east-1，则可以省略该region参数。

```
aws route53domains resend-contact-reachability-email \  
  --domain-name example.com \  
  --current-expiry-year 2020
```

```
--region us-east-1 \  
--domain-name example.com
```

输出：

```
{  
  "domainName": "example.com",  
  "emailAddress": "moliveira@example.com",  
  "isAlreadyVerified": true  
}
```

如果的值 `isAlreadyVerified` 为 `true`，如本例所示，则注册人联系人已经确认可以访问指定的电子邮件地址。

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[重新发送授权和确认电子邮件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ResendContactReachabilityEmail](#)中的。

## retrieve-domain-auth-code

以下代码示例演示了如何使用 `retrieve-domain-auth-code`。

### AWS CLI

获取域名的授权码，以便您可以将域名转移到其他注册商

以下 `retrieve-domain-auth-code` 命令获取 `example.com` 域名的当前授权码。当您想将域名转移给其他域名注册商时，您可以将此值提供给该注册商。

此命令仅在 `us-east-1` 区域中运行。如果您的默认区域设置为 `us-east-1`，则可以省略该 `region` 参数。

```
aws route53domains retrieve-domain-auth-code \  
--region us-east-1 \  
--domain-name example.com
```

输出：

```
{  
  "AuthCode": ")o!v3dJeXampLe"  
}
```

有关更多信息，请参阅 [《亚马逊 Route 53 开发者指南》](#) 中的 [将域名从 Amazon Route 53 转移到其他注册商](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RetrieveDomainAuthCode](#) 中的。

## transfer-domain

以下代码示例演示了如何使用 transfer-domain。

### AWS CLI

将域名转移到亚马逊 Route 53

以下 transfer-domain 命令使用由 JSON 格式文件提供的参数将域传输到 Route 53。C:\temp\transfer-domain.json

此命令仅在 us-east-1 区域中运行。如果您的默认区域设置为 us-east-1，则可以省略该 region 参数。

```
aws route53domains transfer-domain \  
  --region us-east-1 \  
  --cli-input-json file://C:\temp\transfer-domain.json
```

transfer-domain.json 的内容：

```
{  
  "DomainName": "example.com",  
  "DurationInYears": 1,  
  "Nameservers": [  
    {  
      "Name": "ns-2048.awsdns-64.com"  
    },  
    {  
      "Name": "ns-2049.awsdns-65.net"  
    },  
    {  
      "Name": "ns-2050.awsdns-66.org"  
    },  
    {  
      "Name": "ns-2051.awsdns-67.co.uk"  
    }  
  ],  
  "AuthCode": ")o!v3dJeXampLe",
```

```
"AutoRenew": true,
"AdminContact": {
  "FirstName": "Martha",
  "LastName": "Rivera",
  "ContactType": "PERSON",
  "OrganizationName": "Example",
  "AddressLine1": "1 Main Street",
  "City": "Anytown",
  "State": "WA",
  "CountryCode": "US",
  "ZipCode": "98101",
  "PhoneNumber": "+1.8005551212",
  "Email": "mrivera@example.com"
},
"RegistrantContact": {
  "FirstName": "Li",
  "LastName": "Juan",
  "ContactType": "PERSON",
  "OrganizationName": "Example",
  "AddressLine1": "1 Main Street",
  "City": "Anytown",
  "State": "WA",
  "CountryCode": "US",
  "ZipCode": "98101",
  "PhoneNumber": "+1.8005551212",
  "Email": "ljuan@example.com"
},
"TechContact": {
  "FirstName": "Mateo",
  "LastName": "Jackson",
  "ContactType": "PERSON",
  "OrganizationName": "Example",
  "AddressLine1": "1 Main Street",
  "City": "Anytown",
  "State": "WA",
  "CountryCode": "US",
  "ZipCode": "98101",
  "PhoneNumber": "+1.8005551212",
  "Email": "mjackson@example.com"
},
"PrivacyProtectAdminContact": true,
"PrivacyProtectRegistrantContact": true,
"PrivacyProtectTechContact": true
```



```
}
```

输出：

```
{
  "OperationId": "b114c44a-9330-47d1-a6e8-a0b11example"
}
```

要确认操作成功，可以运行`get-operation-detail`。有关更多信息，请参阅[get-operation-detail](#)。

有关更多信息，请参阅《亚马逊 [Route 53 开发者指南](#)》中的[将域名注册转移到亚马逊 Route 53](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TransferDomain](#)中的。

## update-domain-contact-privacy

以下代码示例演示了如何使用 `update-domain-contact-privacy`。

### AWS CLI

#### 更新域名联系人的隐私设置

以下`update-domain-contact-privacy`命令关闭 `example.com` 域名管理员联系人的隐私保护。此命令仅在`us-east-1`区域中运行。

如果您的默认区域设置为`us-east-1`，则可以省略该`region`参数。

```
aws route53domains update-domain-contact-privacy \
  --region us-east-1 \
  --domain-name example.com \
  --no-admin-privacy
```

输出：

```
{
  "OperationId": "b3a219e9-d801-4244-b533-b7256example"
}
```

要确认操作成功，可以运行`get-operation-detail`。有关更多信息，请参阅[get-operation-detail](#)。

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[启用或禁用域名联系信息的隐私保护](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateDomainContactPrivacy](#)中的。

## update-domain-contact

以下代码示例演示了如何使用 update-domain-contact。

### AWS CLI

#### 更新域名的联系信息

以下 update-domain-contact 命令更新域的联系信息，从 JSON 格式的文件中获取参数。C:\temp\update-domain-contact.json

此命令仅在 us-east-1 区域中运行。如果您的默认区域设置为 us-east-1，则可以省略该 region 参数。

```
aws route53domains update-domain-contact \
  --region us-east-1 \
  --cli-input-json file://C:\temp\update-domain-contact.json
```

update-domain-contact.json 的内容：

```
{
  "AdminContact": {
    "AddressLine1": "101 Main Street",
    "AddressLine2": "Suite 1a",
    "City": "Seattle",
    "ContactType": "COMPANY",
    "CountryCode": "US",
    "Email": "w.xiulan@example.com",
    "FirstName": "Wang",
    "LastName": "Xiulan",
    "OrganizationName": "Example",
    "PhoneNumber": "+1.8005551212",
    "State": "WA",
    "ZipCode": "98101"
  },
  "DomainName": "example.com",
  "RegistrantContact": {
    "AddressLine1": "101 Main Street",
```

```
    "AddressLine2": "Suite 1a",
    "City": "Seattle",
    "ContactType": "COMPANY",
    "CountryCode": "US",
    "Email": "w.xiulan@example.com",
    "FirstName": "Wang",
    "LastName": "Xiulan",
    "OrganizationName": "Example",
    "PhoneNumber": "+1.8005551212",
    "State": "WA",
    "ZipCode": "98101"
  },
  "TechContact": {
    "AddressLine1": "101 Main Street",
    "AddressLine2": "Suite 1a",
    "City": "Seattle",
    "ContactType": "COMPANY",
    "CountryCode": "US",
    "Email": "w.xiulan@example.com",
    "FirstName": "Wang",
    "LastName": "Xiulan",
    "OrganizationName": "Example",
    "PhoneNumber": "+1.8005551212",
    "State": "WA",
    "ZipCode": "98101"
  }
}
```

输出：

```
{
  "OperationId": "b3a219e9-d801-4244-b533-b7256example"
}
```

要确认操作成功，可以运行[get-domain-detail](#)。有关更多信息，请参阅 Amazon Route 53 开发者指南中的[更新域名的联系信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDomainContact](#)中的。

## update-domain-nameservers

以下代码示例演示了如何使用 update-domain-nameservers。

## AWS CLI

### 更新域名的名称服务器

以下update-domain-nameservers命令更新域名的名称服务器。

此命令仅在us-east-1区域中运行。如果您的默认区域设置为us-east-1，则可以省略该region参数。

```
aws route53domains update-domain-nameservers \  
  --region us-east-1 \  
  --domain-name example.com \  
  --nameservers Name=ns-1.awsdns-01.org Name=ns-2.awsdns-02.co.uk  
  Name=ns-3.awsdns-03.net Name=ns-4.awsdns-04.com
```

输出：

```
{  
  "OperationId": "f1691ec4-0e7a-489e-82e0-b19d3example"  
}
```

要确认操作成功，可以运行[get-domain-detail](#)。

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[添加或更改域名的名称服务器和 Glue 记录](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDomainNameservers](#)中的。

## update-tags-for-domain

以下代码示例演示了如何使用 update-tags-for-domain。

### AWS CLI

#### 为域名添加或更新标签

以下update-tags-for-domain命令添加或更新 example.com 域的两个密钥和相应值。要更新密钥的值，只需添加密钥和新值即可。一次只能在一个域中添加或更新标签。

此命令仅在us-east-1区域中运行。如果您的默认区域设置为us-east-1，则可以省略该region参数。

```
aws route53domains update-tags-for-domain \  
  --region us-east-1 \  
  --domain-name example.com \  
  --tags Key=tag-key Value=tag-value
```

```
--region us-east-1 \  
--domain-name example.com \  
--tags-to-update "Key=key1,Value=value1" "Key=key2,Value=value2"
```

此命令不生成任何输出。要确认标签已添加或更新，可以运行[list-tags-for-domain](#)。

有关更多信息，请参阅《[亚马逊 Route 53 开发者指南](#)》中的[为亚马逊 Route 53 资源添加标签](#)。

• 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateTagsForDomain](#)中的。

## view-billing

以下代码示例演示了如何使用 view-billing。

### AWS CLI

获取当前 AWS 账户域名注册费用的账单信息

以下view-billing命令返回当前账户自2018年1月1日（Unix时间为1514764800美元）至2019年12月31日午夜（Unix时间为1577836800）期间的所有与域名相关的账单记录。

此命令仅在us-east-1区域中运行。如果您的默认区域设置为us-east-1，则可以省略该region参数。

```
aws route53domains view-billing \  
--region us-east-1 \  
--start-time 1514764800 \  
--end-time 1577836800
```

输出：

```
{  
  "BillingRecords": [  
    {  
      "DomainName": "example.com",  
      "Operation": "RENEW_DOMAIN",  
      "InvoiceId": "149962827",  
      "BillDate": 1536618063.181,  
      "Price": 12.0  
    },  
    {  
      "DomainName": "example.com",  
      "Operation": "RENEW_DOMAIN",
```

```
        "InvoiceId": "290913289",
        "BillDate": 1568162630.884,
        "Price": 12.0
    }
]
}
```

有关更多信息，请参阅[ViewBilling](#) 《亚马逊 Route 53 API 参考》。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ViewBilling](#)中的。

## 使用 Route 53 的解析器示例 AWS CLI

以下代码示例向您展示了如何使用与 Route 53 Resolver AWS Command Line Interface 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **associate-firewall-rule-group**

以下代码示例演示了如何使用 `associate-firewall-rule-group`。

#### AWS CLI

将防火墙规则组与 VPC 关联

以下`associate-firewall-rule-group`示例将 DNS 防火墙规则组与 Amazon VPC 关联起来。

```
aws route53resolver associate-firewall-rule-group \
    --name test-association \
```

```
--firewall-rule-group-id rslvr-frg-47f93271fexample \  
--vpc-id vpc-31e92222 \  
--priority 101
```

输出：

```
{  
  "FirewallRuleGroupAssociation": {  
    "Id": "rslvr-frgassoc-57e8873d7example",  
    "Arn": "arn:aws:route53resolver:us-west-2:123456789012:firewall-rule-group-  
association/rslvr-frgassoc-57e8873d7example",  
    "FirewallRuleGroupId": "rslvr-frg-47f93271fexample",  
    "VpcId": "vpc-31e92222",  
    "Name": "test-association",  
    "Priority": 101,  
    "MutationProtection": "DISABLED",  
    "Status": "UPDATING",  
    "StatusMessage": "Creating Firewall Rule Group Association",  
    "CreatorRequestId": "2ca1a304-32b3-4f5f-bc4c-EXAMPLE11111",  
    "CreationTime": "2021-05-25T21:47:48.755768Z",  
    "ModificationTime": "2021-05-25T21:47:48.755768Z"  
  }  
}
```

有关更多信息，请参阅《亚马逊 Route 53 开发者指南》中的[管理您的 VPC 和 Route 53 解析器 DNS 防火墙规则组之间的关联](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateFirewallRuleGroup](#)中的。

## associate-resolver-endpoint-ip-address

以下代码示例演示了如何使用 associate-resolver-endpoint-ip-address。

### AWS CLI

将另一个 IP 地址与解析器端点相关联

以下associate-resolver-endpoint-ip-address示例将另一个 IP 地址与入站 Resolver 端点相关联。如果您仅指定子网 ID 而在--ip-address参数中省略了 IP 地址，则 Resolver 会从指定子网中的可用 IP 地址中为您选择一个 IP 地址。

```
aws route53resolver associate-resolver-endpoint-ip-address \  

```

```
--resolver-endpoint-id rslvr-in-497098ad5example \  
--ip-address="SubnetId=subnet-12d8exam,Ip=192.0.2.118"
```

输出：

```
{  
  "ResolverEndpoint": {  
    "Id": "rslvr-in-497098ad5example",  
    "CreatorRequestId": "AWSConsole.25.0123456789",  
    "Arn": "arn:aws:route53resolver:us-west-2:111122223333:resolver-endpoint/  
rslvr-in-497098ad5example",  
    "Name": "my-inbound-endpoint",  
    "SecurityGroupIds": [  
      "sg-05cd7b25d6example"  
    ],  
    "Direction": "INBOUND",  
    "IpAddressCount": 3,  
    "HostVPCId": "vpc-304bexam",  
    "Status": "UPDATING",  
    "StatusMessage": "Updating the Resolver Endpoint",  
    "CreationTime": "2020-01-02T23:25:45.538Z",  
    "ModificationTime": "2020-01-02T23:25:45.538Z"  
  }  
}
```

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[您在创建或编辑入站终端节点时指定的值](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[AssociateResolverEndpointIpAddress](#)中的。

## associate-resolver-rule

以下代码示例演示了如何使用 `associate-resolver-rule`。

### AWS CLI

#### 将解析器规则与 VPC 关联

以下 `associate-resolver-rule` 示例将解析器规则与 Amazon VPC 关联起来。运行命令后，Resolver 开始根据规则中的设置（例如转发的查询的域名）将 DNS 查询转发到您的网络。

```
aws route53resolver associate-resolver-rule \  

```



```
--name my-resolver-rule-association \  
--resolver-rule-id rslvr-rr-42b60677c0example \  
--vpc-id vpc-304bexam
```

输出：

```
{  
  "ResolverRuleAssociation": {  
    "Id": "rslvr-rrassoc-d61cbb2c8bexample",  
    "ResolverRuleId": "rslvr-rr-42b60677c0example",  
    "Name": "my-resolver-rule-association",  
    "VPCId": "vpc-304bexam",  
    "Status": "CREATING",  
    "StatusMessage": "[Trace id: 1-5dc5a8fa-ec2cc480d2ef07617example] Creating  
the association."  
  }  
}
```

有关更多信息，请参阅 Amazon Route 53 开发者指南中的将[出站 DNS 查询转发到您的网络](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateResolverRule](#) 中的。

## create-firewall-domain-list

以下代码示例演示了如何使用 create-firewall-domain-list。

### AWS CLI

创建 Route 53 解析器 DNS 防火墙域列表

以下 create-firewall-domain-list 示例在您的 AWS 账户中创建名为 test 的 Route 53 Resolver DNS 防火墙域列表。

```
aws route53resolver create-firewall-domain-list \  
  --creator-request-id my-request-id \  
  --name test
```

输出：

```
{  
  "FirewallDomainList": {  
    "Id": "rslvr-fdl-d61cbb2cbexample",
```

```
    "Arn": "arn:aws:route53resolver:us-west-2:123456789012:firewall-domain-list/
rslvr-fdl-d61cbb2cbexample",
    "Name": "test",
    "DomainCount": 0,
    "Status": "COMPLETE",
    "StatusMessage": "Created Firewall Domain List",
    "CreatorRequestId": "my-request-id",
    "CreationTime": "2021-05-25T15:55:51.115365Z",
    "ModificationTime": "2021-05-25T15:55:51.115365Z"
  }
}
```

有关更多信息，请参阅《Amazon Route 53 开发者指南》中的[管理自己的域名列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFirewallDomainList](#)中的。

## create-firewall-rule-group

以下代码示例演示了如何使用 create-firewall-rule-group。

### AWS CLI

#### 创建防火墙规则组

以下create-firewall-rule-group示例创建 DNS 防火墙规则组。

```
aws route53resolver create-firewall-rule-group \
  --creator-request-id my-request-id \
  --name test
```

输出：

```
{
  "FirewallRuleGroup": {
    "Id": "rslvr-frg-47f93271fexample",
    "Arn": "arn:aws:route53resolver:us-west-2:123456789012:firewall-rule-group/
rslvr-frg-47f93271fexample",
    "Name": "test",
    "RuleCount": 0,
    "Status": "COMPLETE",
    "StatusMessage": "Created Firewall Rule Group",
    "OwnerId": "123456789012",
    "CreatorRequestId": "my-request-id",
```

```

    "ShareStatus": "NOT_SHARED",
    "CreationTime": "2021-05-25T18:59:26.490017Z",
    "ModificationTime": "2021-05-25T18:59:26.490017Z"
  }
}

```

有关更多信息，请参阅 [Amazon Route 53 开发者指南中的管理 DNS 防火墙中的规则组和规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateFirewallRuleGroup](#) 中的。

## create-firewall-rule

以下代码示例演示了如何使用 `create-firewall-rule`。

### AWS CLI

#### 创建防火墙规则

以下 `create-firewall-rule` 示例在 DNS 防火墙规则中为 DNS 防火墙域列表中列出的域创建防火墙规则。

```

aws route53resolver create-firewall-rule \
  --name allow-rule \
  --firewall-rule-group-id rslvr-frg-47f93271fexample \
  --firewall-domain-list-id rslvr-fdl-9e956e9ffexample \
  --priority 101 \
  --action ALLOW

```

输出：

```

{
  "FirewallRule": {
    "FirewallRuleGroupId": "rslvr-frg-47f93271fexample",
    "FirewallDomainListId": "rslvr-fdl-9e956e9ffexample",
    "Name": "allow-rule",
    "Priority": 101,
    "Action": "ALLOW",
    "CreatorRequestId": "d81e3fb7-020b-415e-939f-EXAMPLE11111",
    "CreationTime": "2021-05-25T21:44:00.346093Z",
    "ModificationTime": "2021-05-25T21:44:00.346093Z"
  }
}

```

有关更多信息，请参阅 [Amazon Route 53 开发者指南中的管理 DNS 防火墙中的规则组和规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateFirewallRule](#) 中的。

## create-resolver-endpoint

以下代码示例演示了如何使用 create-resolver-endpoint。

### AWS CLI

#### 创建入站解析器端点

以下 create-resolver-endpoint 示例创建了一个入站解析器端点。您可以使用相同的命令来创建入站和出站终端节点。

```
aws route53resolver -- create-resolver-endpoint name -- creator-request-id 2020-01-01-18:47 --
"sg-f62bexam" -- security-group-ids "sg-f62be my-inbound-endpoint xam" -- direction INBOUND
-- ip-addresses =subnet-ba47exam , Ip=192.0.2.255 =subnet-12d8exam , Ip=192.0.2.254
SubnetId SubnetId
```

输出：

```
{
  "ResolverEndpoint": {
    "Id": "rslvr-in-f9ab8a03f1example",
    "CreatorRequestId": "2020-01-01-18:47",
    "Arn": "arn:aws:route53resolver:us-west-2:111122223333:resolver-endpoint/
rslvr-in-f9ab8a03f1example",
    "Name": "my-inbound-endpoint",
    "SecurityGroupIds": [
      "sg-f62bexam"
    ],
    "Direction": "INBOUND",
    "IpAddressCount": 2,
    "HostVPCId": "vpc-304examp",
    "Status": "CREATING",
    "StatusMessage": "[Trace id: 1-5dc1ff84-f3477826e4a190025example] Creating
the Resolver Endpoint",
    "CreationTime": "2020-01-01T23:02:29.583Z",
    "ModificationTime": "2020-01-01T23:02:29.583Z"
  }
}
```

## 创建出站解析器终端节点

以下 `create-resolver-endpoint` 示例使用 JSON 格式文档中的值创建出站解析器端点。 `create-outbound-resolver-endpoint.json`

```
aws route53resolver create-resolver-endpoint \  
  --cli-input-json file:///c:\temp\create-outbound-resolver-endpoint.json
```

`create-outbound-resolver-endpoint.json` 的内容：

```
{  
  "CreatorRequestId": "2020-01-01-18:47",  
  "Direction": "OUTBOUND",  
  "IpAddresses": [  
    {  
      "Ip": "192.0.2.255",  
      "SubnetId": "subnet-ba47exam"  
    },  
    {  
      "Ip": "192.0.2.254",  
      "SubnetId": "subnet-12d8exam"  
    }  
  ],  
  "Name": "my-outbound-endpoint",  
  "SecurityGroupIds": [ "sg-05cd7b25d6example" ],  
  "Tags": [  
    {  
      "Key": "my-key-name",  
      "Value": "my-key-value"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon Route 53 Developer Guide 中 [解析 VPC 与您的网络之间的 DNS 查询](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateResolverEndpoint](#) 中的。

## `create-resolver-rule`

以下代码示例演示了如何使用 `create-resolver-rule`。

## AWS CLI

### 创建解析器规则

以下 `create-resolver-rule` 示例创建了 Resolver 转发规则。该规则使用出站终端节点 `rslvr-out-d5e5920e37example` 将 DNS 查询转发到 IP 地址 `10.24.8.75` 和 `10.24.8.156`。

```
aws route53resolver create-resolver-rule \
  --creator-request-id 2020-01-02-18:47 \
  --domain-name example.com \
  --name my-rule \
  --resolver-endpoint-id rslvr-out-d5e5920e37example \
  --rule-type FORWARD \
  --target-ips "Ip=10.24.8.75" "Ip=10.24.8.156"
```

输出：

```
{
  "ResolverRule": {
    "Status": "COMPLETE",
    "RuleType": "FORWARD",
    "ResolverEndpointId": "rslvr-out-d5e5920e37example",
    "Name": "my-rule",
    "DomainName": "example.com.",
    "CreationTime": "2022-05-10T21:35:30.923187Z",
    "TargetIps": [
      {
        "Ip": "10.24.8.75",
        "Port": 53
      },
      {
        "Ip": "10.24.8.156",
        "Port": 53
      }
    ],
    "CreatorRequestId": "2022-05-10-16:33",
    "ModificationTime": "2022-05-10T21:35:30.923187Z",
    "ShareStatus": "NOT_SHARED",
    "Arn": "arn:aws:route53resolver:us-east-1:111117012054:resolver-rule/rslvr-rr-b1e0b905e93611111",
    "OwnerId": "111111111111",
    "Id": "rslvr-rr-rslvr-rr-b1e0b905e93611111",
  }
}
```

```
    "StatusMessage": "[Trace id: 1-22222222-3e56afcc71a3724664f22e24]
    Successfully created Resolver Rule."
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateResolverRule](#)中的。

## delete-firewall-domain-list

以下代码示例演示了如何使用 delete-firewall-domain-list。

### AWS CLI

#### 删除 Route 53 解析器 DNS 防火墙域列表

以下delete-firewall-domain-list示例删除了您 AWS 账户中名为 test 的 Route 53 Resolver DNS 防火墙域列表。

```
aws route53resolver delete-firewall-domain-list \
  --firewall-domain-list-id rslvr-fdl-9e956e9ffexample
```

输出：

```
{
  "FirewallDomainList": {
    "Id": "rslvr-fdl-9e956e9ffexample",
    "Arn": "arn:aws:route53resolver:us-west-2:123456789012:firewall-domain-list/
rslvr-fdl-9e956e9ffexample",
    "Name": "test",
    "DomainCount": 6,
    "Status": "DELETING",
    "StatusMessage": "Deleting the Firewall Domain List",
    "CreatorRequestId": "my-request-id",
    "CreationTime": "2021-05-25T15:55:51.115365Z",
    "ModificationTime": "2021-05-25T18:58:05.588024Z"
  }
}
```

有关更多信息，请参阅《Amazon Route 53 开发者指南》中的[管理自己的域名列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFirewallDomainList](#)中的。

## delete-firewall-rule-group

以下代码示例演示了如何使用 delete-firewall-rule-group。

### AWS CLI

删除防火墙规则组

以下delete-firewall-rule-group示例删除防火墙规则组。

```
aws route53resolver delete-firewall-rule-group \  
  --firewall-rule-group-id rslvr-frg-47f93271fexample
```

输出：

```
{  
  "FirewallRuleGroup": {  
    "Id": "rslvr-frg-47f93271fexample",  
    "Arn": "arn:aws:route53resolver:us-west-2:123456789012:firewall-rule-group/  
rslvr-frg-47f93271fexample",  
    "Name": "test",  
    "RuleCount": 0,  
    "Status": "UPDATING",  
    "StatusMessage": "Updating Firewall Rule Group",  
    "OwnerId": "123456789012",  
    "CreatorRequestId": "my-request-id",  
    "ShareStatus": "NOT_SHARED",  
    "CreationTime": "2021-05-25T18:59:26.490017Z",  
    "ModificationTime": "2021-05-25T21:51:53.028688Z"  
  }  
}
```

有关更多信息，请参阅 [Amazon Route 53 开发者指南中的管理 DNS 防火墙中的规则组和规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFirewallRuleGroup](#)中的。

## delete-firewall-rule

以下代码示例演示了如何使用 delete-firewall-rule。

### AWS CLI

删除防火墙规则



以下delete-firewall-rule示例删除了指定的防火墙规则。

```
aws route53resolver delete-firewall-rule \  
  --firewall-rule-group-id rslvr-frg-47f93271fexample \  
  --firewall-domain-list-id rslvr-fdl-9e956e9ffexample
```

输出：

```
{  
  "FirewallRule": {  
    "FirewallRuleGroupId": "rslvr-frg-47f93271fexample",  
    "FirewallDomainListId": "rslvr-fdl-9e956e9ffexample",  
    "Name": "allow-rule",  
    "Priority": 102,  
    "Action": "ALLOW",  
    "CreatorRequestId": "d81e3fb7-020b-415e-939f-EXAMPLE11111",  
    "CreationTime": "2021-05-25T21:44:00.346093Z",  
    "ModificationTime": "2021-05-25T21:45:59.611600Z"  
  }  
}
```

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[管理 DNS 防火墙中的规则组和规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFirewallRule](#)中的。

## delete-resolver-endpoint

以下代码示例演示了如何使用 delete-resolver-endpoint。

### AWS CLI

#### 删除解析器端点

以下delete-resolver-endpoint示例删除了指定的端点。

**重要信息**如果您删除入站终端节点，则来自您的网络的 DNS 查询将不再转发到您在终端节点中指定的 VPC 中的 Resolver。如果您删除了出站端点，对于指定了所删除端点的规则，Resolver 会停止将 DNS 查询从您的 VPC 转发到您的网络。

```
aws route53resolver delete-resolver-endpoint \  
  --resolver-endpoint-id rslvr-in-497098ad59example
```

输出：

```
{
  "ResolverEndpoint": {
    "Id": "rslvr-in-497098ad59example",
    "CreatorRequestId": "AWSConsole.25.157290example",
    "Arn": "arn:aws:route53resolver:us-west-2:111122223333:resolver-endpoint/rslvr-in-497098ad59example",
    "Name": "my-inbound-endpoint",
    "SecurityGroupIds": [
      "sg-05cd7b25d6example"
    ],
    "Direction": "INBOUND",
    "IpAddressCount": 5,
    "HostVPCId": "vpc-304bexam",
    "Status": "DELETING",
    "StatusMessage": "[Trace id: 1-5dc5b658-811b5be0922bbc382example] Deleting ResolverEndpoint.",
    "CreationTime": "2020-01-01T23:25:45.538Z",
    "ModificationTime": "2020-01-02T23:25:45.538Z"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteResolverEndpoint](#)中的。

## delete-resolver-rule

以下代码示例演示了如何使用 delete-resolver-rule。

AWS CLI

删除解析器规则

以下delete-resolver-rule示例删除了指定的规则。

注意如果规则与任何 VPC 关联，则必须先取消该规则与 VPC 的关联，然后才能将其删除。

```
aws route53resolver delete-resolver-rule \
  --resolver-rule-id rslvr-rr-5b3809426example
```

输出：

```
{
  "ResolverRule": {
    "Id": "rslvr-rr-5b3809426bexample",
    "CreatorRequestId": "2020-01-03-18:47",
    "Arn": "arn:aws:route53resolver:us-west-2:111122223333:resolver-rule/rslvr-rr-5b3809426bexample",
    "DomainName": "zenith.example.com.",
    "Status": "DELETING",
    "StatusMessage": "[Trace id: 1-5dc5e05b-602e67b052cb74f05example] Deleting Resolver Rule.",
    "RuleType": "FORWARD",
    "Name": "my-resolver-rule",
    "TargetIps": [
      {
        "Ip": "192.0.2.50",
        "Port": 53
      }
    ],
    "ResolverEndpointId": "rslvr-out-d5e5920e3example",
    "OwnerId": "111122223333",
    "ShareStatus": "NOT_SHARED"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteResolverRule](#)中的。

## disassociate-firewall-rule-group

以下代码示例演示了如何使用 `disassociate-firewall-rule-group`。

### AWS CLI

取消防火墙规则组与 VPC 的关联

以下`disassociate-firewall-rule-group`示例取消了 DNS 防火墙规则组与 Amazon VPC 的关联。

```
aws route53resolver disassociate-firewall-rule-group \
  --firewall-rule-group-association-id rslvr-frgassoc-57e8873d7example
```

输出：

```
{
  "FirewallRuleGroupAssociation": {
    "Id": "rslvr-frgassoc-57e8873d7example",
    "Arn": "arn:aws:route53resolver:us-west-2:123456789012:firewall-rule-group-association/rslvr-frgassoc-57e8873d7example",
    "FirewallRuleGroupId": "rslvr-frg-47f93271fexample",
    "VpcId": "vpc-31e92222",
    "Name": "test-association",
    "Priority": 103,
    "MutationProtection": "DISABLED",
    "Status": "DELETING",
    "StatusMessage": "Deleting the Firewall Rule Group Association",
    "CreatorRequestId": "2ca1a304-32b3-4f5f-bc4c-EXAMPLE11111",
    "CreationTime": "2021-05-25T21:47:48.755768Z",
    "ModificationTime": "2021-05-25T21:51:02.377887Z"
  }
}
```

有关更多信息，请参阅《亚马逊 Route 53 开发者指南》中的[管理您的 VPC 和 Route 53 解析器 DNS 防火墙规则组之间的关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateFirewallRuleGroup](#) 中的。

## disassociate-resolver-endpoint-ip-address

以下代码示例演示了如何使用 `disassociate-resolver-endpoint-ip-address`。

### AWS CLI

解除 IP 地址与解析器端点的关联

以下 `disassociate-resolver-endpoint-ip-address` 示例从指定的 Resolver 入站或出站终端节点中删除 IP 地址。

注意一个端点必须至少有两个 IP 地址。如果一个端点当前只有两个 IP 地址，并且您想将一个地址替换为另一个地址，则必须先使用 [associate-resolver-endpoint-ip-address](#) 来关联新的 IP 地址。然后，您可以取消其中一个原始 IP 地址与终端节点的关联。

```
aws route53resolver disassociate-resolver-endpoint-ip-address \
  --resolver-endpoint-id rslvr-in-f9ab8a03f1example \
  --ip-address="SubnetId=subnet-12d8a459,Ip=172.31.40.121"
```

输出：

```
{
  "ResolverEndpoint": {
    "Id": "rslvr-in-f9ab8a03f1example",
    "CreatorRequestId": "2020-01-01-18:47",
    "Arn": "arn:aws:route53resolver:us-west-2:111122223333:resolver-endpoint/rslvr-in-f9ab8a03f1example",
    "Name": "my-inbound-endpoint",
    "SecurityGroupIds": [
      "sg-f62bexam"
    ],
    "Direction": "INBOUND",
    "IpAddressCount": 3,
    "HostVPCId": "vpc-304bexam",
    "Status": "UPDATING",
    "StatusMessage": "Updating the Resolver Endpoint",
    "CreationTime": "2020-01-01T23:02:29.583Z",
    "ModificationTime": "2020-01-05T23:02:29.583Z"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateResolverEndpointIpAddress](#)中的。

## disassociate-resolver-rule

以下代码示例演示了如何使用 disassociate-resolver-rule。

### AWS CLI

取消解析程序规则与 Amazon VPC 的关联

以下disassociate-resolver-rule示例删除了指定的 Resolver 规则与指定 VPC 之间的关联。在以下情况下，您可以取消规则与 VPC 的关联：

对于源自此 VPC 的 DNS 查询，您希望 Resolver 停止向您的网络转发规则中指定的域名的查询。您想删除转发规则。如果规则当前与一个或多个 VPC 关联，您必须先从所有 VPC 上解除规则的关联，然后才能删除规则。

```
aws route53resolver disassociate-resolver-rule \
  --resolver-rule-id rslvr-rr-4955cb98ceexample \
```

```
--vpc-id vpc-304bexam
```

输出：

```
{
  "ResolverRuleAssociation": {
    "Id": "rslvr-rrassoc-322f4e8b9cexample",
    "ResolverRuleId": "rslvr-rr-4955cb98ceexample",
    "Name": "my-resolver-rule-association",
    "VPCId": "vpc-304bexam",
    "Status": "DELETING",
    "StatusMessage": "[Trace id: 1-5dc5ffa2-a26c38004c1f94006example] Deleting
Association"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateResolverRule](#)中的。

## get-firewall-config

以下代码示例演示了如何使用 `get-firewall-config`。

### AWS CLI

获取 VPC 的防火墙配置

以下`get-firewall-config`示例检索指定 VPC 的 DNS 防火墙行为。

```
aws route53resolver get-firewall-config \
  --resource-id vpc-31e92222
```

输出：

```
{
  "FirewallConfig": {
    "Id": "rslvr-fc-86016850cexample",
    "ResourceId": "vpc-31e92222",
    "OwnerId": "123456789012",
    "FirewallFailOpen": "DISABLED"
  }
}
```

有关更多信息，请参阅《亚马逊 Route 53 开发者指南》中的 [DNS 防火墙 VPC 配置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFirewallConfig](#)中的。

## get-firewall-domain-list

以下代码示例演示了如何使用 get-firewall-domain-list。

### AWS CLI

获取 Route 53 解析器 DNS 防火墙域名列表

以下get-firewall-domain-list示例使用您指定 ID 检索域列表。

```
aws route53resolver get-firewall-domain-list \  
  --firewall-domain-list-id rslvr-fdl-42b60677cexample
```

输出：

```
{  
  "FirewallDomainList": {  
    "Id": "rslvr-fdl-9e956e9ffexample",  
    "Arn": "arn:aws:route53resolver:us-west-2:123457689012:firewall-domain-list/  
rslvr-fdl-42b60677cexample",  
    "Name": "test",  
    "DomainCount": 0,  
    "Status": "COMPLETE",  
    "StatusMessage": "Created Firewall Domain List",  
    "CreatorRequestId": "my-request-id",  
    "CreationTime": "2021-05-25T15:55:51.115365Z",  
    "ModificationTime": "2021-05-25T15:55:51.115365Z"  
  }  
}
```

有关更多信息，请参阅《Amazon Route 53 开发者指南》中的[管理自己的域名列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFirewallDomainList](#)中的。

## get-firewall-rule-group-association

以下代码示例演示了如何使用 get-firewall-rule-group-association。

## AWS CLI

### 获取防火墙规则组关联

以下`get-firewall-rule-group-association`示例检索防火墙规则组关联。

```
aws route53resolver get-firewall-rule-group-association \  
  --firewall-rule-group-association-id rslvr-frgassoc-57e8873d7example
```

输出：

```
{  
  "FirewallRuleGroupAssociation": {  
    "Id": "rslvr-frgassoc-57e8873d7example",  
    "Arn": "arn:aws:route53resolver:us-west-2:123456789012:firewall-rule-group-  
association/rslvr-frgassoc-57e8873d7example",  
    "FirewallRuleGroupId": "rslvr-frg-47f93271fexample",  
    "VpcId": "vpc-31e92222",  
    "Name": "test-association",  
    "Priority": 101,  
    "MutationProtection": "DISABLED",  
    "Status": "COMPLETE",  
    "StatusMessage": "Finished rule group association update",  
    "CreatorRequestId": "2ca1a304-32b3-4f5f-bc4c-EXAMPLE11111",  
    "CreationTime": "2021-05-25T21:47:48.755768Z",  
    "ModificationTime": "2021-05-25T21:47:48.755768Z"  
  }  
}
```

有关更多信息，请参阅《亚马逊 Route 53 开发者指南》中的[管理您的 VPC 和 Route 53 解析器 DNS 防火墙规则组之间的关联](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFirewallRuleGroupAssociation](#)中的。

## get-firewall-rule-group-policy

以下代码示例演示了如何使用 `get-firewall-rule-group-policy`。

## AWS CLI

### 获取 AWS IAM 政策



以下`get-firewall-rule-group-policy`示例获取了用于共享指定规则组的 Identity and Access Management (AWS IAM) 策略。

```
aws route53resolver get-firewall-rule-group-policy \
  --arn arn:aws:route53resolver:us-west-2:AWS_ACCOUNT_ID:firewall-rule-group/
  rslvr-frg-47f93271fexample
```

输出：

```
{
  "FirewallRuleGroupPolicy": "{\"Version\":\"2012-10-17\",
  \"Statement\": [{\"Sid\":\"test\", \"Effect\":\"Allow\", \"Principal\": {\"AWS\": \"arn:aws:iam::AWS_ACCOUNT_ID:root\"}, \"Action\": [\"route53resolver:GetFirewallRuleGroup\", \"route53resolver:ListFirewallRuleGroups\"], \"Resource\": \"arn:aws:route53resolver:us-east-1:AWS_ACCOUNT_ID:firewall-rule-group/rslvr-frg-47f93271fexample\"}]}"
}
```

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[管理 DNS 防火墙中的规则组和规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetFirewallRuleGroupPolicy](#)中的。

## get-firewall-rule-group

以下代码示例演示了如何使用 `get-firewall-rule-group`。

### AWS CLI

获取防火墙规则组

以下`get-firewall-rule-group`示例使用您提供的 ID 检索有关 DNS 防火墙规则组的信息。

```
aws route53resolver get-firewall-rule-group \
  --firewall-rule-group-id rslvr-frg-47f93271fexample
```

输出：

```
{
  "FirewallRuleGroup": {
    "Id": "rslvr-frg-47f93271fexample",
    "Arn": "arn:aws:route53resolver:us-west-2:123456789012:firewall-rule-group/
    rslvr-frg-47f93271fexample",
  }
}
```

```

    "Name": "test",
    "RuleCount": 0,
    "Status": "COMPLETE",
    "StatusMessage": "Created Firewall Rule Group",
    "OwnerId": "123456789012",
    "CreatorRequestId": "my-request-id",
    "ShareStatus": "NOT_SHARED",
    "CreationTime": "2021-05-25T18:59:26.490017Z",
    "ModificationTime": "2021-05-25T18:59:26.490017Z"
  }
}

```

有关更多信息，请参阅 [Amazon Route 53 开发者指南中的管理 DNS 防火墙中的规则组和规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetFirewallRuleGroup](#) 中的。

## get-resolver-endpoint

以下代码示例演示了如何使用 `get-resolver-endpoint`。

### AWS CLI

获取有关解析器端点的信息

以下 `get-resolver-endpoint` 示例显示出站指定终端节点的详细信息。您可以通过指定适用的终端节点 ID 来同时 `get-resolver-endpoint` 用于入站和出站终端节点。

```

aws route53resolver get-resolver-endpoint \
  --resolver-endpoint-id rslvr-out-d5e5920e37example

```

输出：

```

{
  "ResolverEndpoint": {
    "Id": "rslvr-out-d5e5920e37example",
    "CreatorRequestId": "2020-01-01-18:47",
    "Arn": "arn:aws:route53resolver:us-west-2:111122223333:resolver-endpoint/rslvr-out-d5e5920e37example",
    "Name": "my-outbound-endpoint",
    "SecurityGroupIds": [
      "sg-05cd7b25d6example"
    ],
    "Direction": "OUTBOUND",
  }
}

```

```
    "IpAddressCount": 2,
    "HostVPCId": "vpc-304bexam",
    "Status": "OPERATIONAL",
    "StatusMessage": "This Resolver Endpoint is operational.",
    "CreationTime": "2020-01-01T23:50:50.979Z",
    "ModificationTime": "2020-01-02T23:50:50.979Z"
  }
}
```

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[您在创建或编辑入站终端节点时指定的值](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetResolverEndpoint](#)中的。

## get-resolver-rule-association

以下代码示例演示了如何使用 get-resolver-rule-association。

### AWS CLI

获取有关解析器规则和 VPC 之间关联的信息

以下 get-resolver-rule-association 示例显示了有关指定解析器规则与 VPC 之间关联的详细信息。您可以使用关联解析器规则和 VP [associate-resolver-ruleC](#)。

```
aws route53resolver get-resolver-rule-association \
  --resolver-rule-association-id rslvr-rrassoc-d61cbb2c8bexample
```

输出：

```
{
  "ResolverRuleAssociation": {
    "Id": "rslvr-rrassoc-d61cbb2c8bexample",
    "ResolverRuleId": "rslvr-rr-42b60677c0example",
    "Name": "my-resolver-rule-association",
    "VPCId": "vpc-304bexam",
    "Status": "COMPLETE",
    "StatusMessage": ""
  }
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetResolverRuleAssociation](#)中的。

## get-resolver-rule

以下代码示例演示了如何使用 `get-resolver-rule`。

### AWS CLI

获取有关解析器规则的信息

以下 `get-resolver-rule` 示例显示有关指定解析器规则的详细信息，例如该规则转发 DNS 查询的域名以及与该规则关联的出站解析器终端节点的 ID。

```
aws route53resolver get-resolver-rule \  
  --resolver-rule-id rslvr-rr-42b60677c0example
```

输出：

```
{  
  "ResolverRule": {  
    "Id": "rslvr-rr-42b60677c0example",  
    "CreatorRequestId": "2020-01-01-18:47",  
    "Arn": "arn:aws:route53resolver:us-west-2:111122223333:resolver-rule/rslvr-  
rr-42b60677c0example",  
    "DomainName": "example.com.",  
    "Status": "COMPLETE",  
    "StatusMessage": "[Trace id: 1-5dc4b177-ff1d9d001a0f80005example]  
Successfully created Resolver Rule.",  
    "RuleType": "FORWARD",  
    "Name": "my-rule",  
    "TargetIps": [  
      {  
        "Ip": "192.0.2.45",  
        "Port": 53  
      }  
    ],  
    "ResolverEndpointId": "rslvr-out-d5e5920e37example",  
    "OwnerId": "111122223333",  
    "ShareStatus": "NOT_SHARED"  
  }  
}
```

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[您在创建或编辑规则时指定的值](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetResolverRule](#)中的。

## import-firewall-domains

以下代码示例演示了如何使用 `import-firewall-domains`。

### AWS CLI

将域名导入域名列表

以下 `import-firewall-domains` 示例将一组域从文件导入到您指定的 DNS 防火墙域列表中。

```
aws route53resolver import-firewall-domains \  
  --firewall-domain-list-id rslvr-fdl-d61cbb2cbexample \  
  --operation REPLACE \  
  --domain-file-url s3://PATH/TO/YOUR/FILE
```

输出：

```
{  
  "Id": "rslvr-fdl-d61cbb2cbexample",  
  "Name": "test",  
  "Status": "IMPORTING",  
  "StatusMessage": "Importing domains from provided file."  
}
```

有关更多信息，请参阅《Amazon Route 53 开发者指南》中的[管理自己的域名列表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ImportFirewallDomains](#) 中的。

## list-firewall-configs

以下代码示例演示了如何使用 `list-firewall-configs`。

### AWS CLI

列出防火墙配置

以下 `list-firewall-configs` 示例列出了您的 DNS 防火墙配置。

```
aws route53resolver list-firewall-configs
```

输出：

```
{
```

```
"FirewallConfigs": [  
  {  
    "Id": "rslvr-fc-86016850cexample",  
    "ResourceId": "vpc-31e92222",  
    "OwnerId": "123456789012",  
    "FirewallFailOpen": "DISABLED"  
  }  
]  
}
```

有关更多信息，请参阅《亚马逊 Route 53 开发者指南》中的 [DNS 防火墙 VPC 配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListFirewallConfigs](#) 中的。

## list-firewall-domain-lists

以下代码示例演示了如何使用 list-firewall-domain-lists。

### AWS CLI

列出所有 Route 53 解析器 DNS 防火墙域列表

以下 list-firewall-domain-lists 示例列出了所有域名列表。

```
aws route53resolver list-firewall-domain-lists
```

输出：

```
{  
  "FirewallDomainLists": [  
    {  
      "Id": "rslvr-fdl-2c46f2ecfexample",  
      "Arn": "arn:aws:route53resolver:us-west-2:123456789012:firewall-domain-list/rslvr-fdl-2c46f2ecfexample",  
      "Name": "AWSManagedDomainsMalwareDomainList",  
      "CreatorRequestId": "AWSManagedDomainsMalwareDomainList",  
      "ManagedOwnerName": "Route 53 Resolver DNS Firewall"  
    },  
    {  
      "Id": "rslvr-fdl-aa970e9e1example",  
      "Arn": "arn:aws:route53resolver:us-west-2:123456789012:firewall-domain-list/rslvr-fdl-aa970e9e1example",  
    }  
  ]  
}
```

```

        "Name": "AWSManagedDomainsBotnetCommandandControl",
        "CreatorRequestId": "AWSManagedDomainsBotnetCommandandControl",
        "ManagedOwnerName": "Route 53 Resolver DNS Firewall"
    },
    {
        "Id": "rslvr-fdl-42b60677cexample",
        "Arn": "arn:aws:route53resolver:us-west-2:123456789111:firewall-domain-
list/rslvr-fdl-42b60677cexample",
        "Name": "test",
        "CreatorRequestId": "my-request-id"
    }
]
}

```

有关更多信息，请参阅《亚马逊 Route 53 开发者指南》中的 Route 53 [Resolver DNS 防火墙域列表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListFirewallDomainLists](#) 中的。

## list-firewall-domains

以下代码示例演示了如何使用 list-firewall-domains。

### AWS CLI

在域名列表中列出域名

以下 list-firewall-domains 示例列出了您指定的 DNS 防火墙域列表中的域。

```
aws route53resolver list-firewall-domains \
  --firewall-domain-list-id rslvr-fdl-d61cbb2cbexample
```

输出：

```

{
  "Domains": [
    "test1.com.",
    "test2.com.",
    "test3.com."
  ]
}

```

有关更多信息，请参阅《Amazon Route 53 开发者指南》中的[管理自己的域名列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFirewallDomains](#)中的。

## list-firewall-rule-group-associations

以下代码示例演示了如何使用 list-firewall-rule-group-associations。

### AWS CLI

列出 DNS 防火墙规则组关联

以下list-firewall-rule-group-associations示例列出了您与 Amazon VPC 的 DNS 防火墙规则组关联。

```
aws route53resolver list-firewall-rule-group-associations
```

输出：

```
{
  "FirewallRuleGroupAssociations": [
    {
      "Id": "rslvr-frgassoc-57e8873d7example",
      "Arn": "arn:aws:route53resolver:us-west-2:123456789012:firewall-rule-group-association/rslvr-frgassoc-57e8873d7example",
      "FirewallRuleGroupId": "rslvr-frg-47f93271fexample",
      "VpcId": "vpc-31e92222",
      "Name": "test-association",
      "Priority": 101,
      "MutationProtection": "DISABLED",
      "Status": "UPDATING",
      "StatusMessage": "Creating Firewall Rule Group Association",
      "CreatorRequestId": "2ca1a304-32b3-4f5f-bc4c-EXAMPLE11111",
      "CreationTime": "2021-05-25T21:47:48.755768Z",
      "ModificationTime": "2021-05-25T21:47:48.755768Z"
    }
  ]
}
```

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[管理您的 VPC 和 Route 53 Resolver DNS 防火墙规则组之间的关联](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFirewallRuleGroupAssociations](#)中的。



## list-firewall-rule-groups

以下代码示例演示了如何使用 `list-firewall-rule-groups`。

### AWS CLI

获取您的防火墙规则组列表

以下 `list-firewall-rule-groups` 示例列出了您的 DNS 防火墙规则组。

```
aws route53resolver list-firewall-rule-groups
```

输出：

```
{
  "FirewallRuleGroups": [
    {
      "Id": "rslvr-frg-47f93271fexample",
      "Arn": "arn:aws:route53resolver:us-west-2:123456789012:firewall-rule-group/rslvr-frg-47f93271fexample",
      "Name": "test",
      "OwnerId": "123456789012",
      "CreatorRequestId": "my-request-id",
      "ShareStatus": "NOT_SHARED"
    }
  ]
}
```

有关更多信息，请参阅 [Amazon Route 53 开发者指南中的管理 DNS 防火墙中的规则组和规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListFirewallRuleGroups](#) 中的。

## list-firewall-rules

以下代码示例演示了如何使用 `list-firewall-rules`。

### AWS CLI

列出防火墙规则

以下 `list-firewall-rules` 示例列出了防火墙规则组中的所有 DNS 防火墙规则。

```
aws route53resolver list-firewall-rules \
```

```
--firewall-rule-group-id rslvr-frg-47f93271fexample
```

输出：

```
{
  "FirewallRules": [
    {
      "FirewallRuleGroupId": "rslvr-frg-47f93271fexample",
      "FirewallDomainListId": "rslvr-fdl-9e956e9ffexample",
      "Name": "allow-rule",
      "Priority": 101,
      "Action": "ALLOW",
      "CreatorRequestId": "d81e3fb7-020b-415e-939f-EXAMPLE11111",
      "CreationTime": "2021-05-25T21:44:00.346093Z",
      "ModificationTime": "2021-05-25T21:44:00.346093Z"
    }
  ]
}
```

有关更多信息，请参阅 [Amazon Route 53 开发者指南中的管理 DNS 防火墙中的规则组和规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListFirewallRules](#) 中的。

## list-resolver-endpoint-ip-addresses

以下代码示例演示了如何使用 `list-resolver-endpoint-ip-addresses`。

### AWS CLI

列出指定入站或出站终端节点的 IP 地址

以下 `list-resolver-endpoint-ip-addresses` 示例列出了与入站终端节点关联的 IP 地址的相关信息 `rslvr-in-f9ab8a03f1example`。您也可以通过指定适用的终端节点 ID 来 `list-resolver-endpoint-ip-addresses` 用于出站终端节点。

```
aws route53resolver list-resolver-endpoint-ip-addresses \
  --resolver-endpoint-id rslvr-in-f9ab8a03f1example
```

输出：

```
{
  "MaxResults": 10,
```

```
"IpAddresses": [  
  {  
    "IpId": "rni-1de60cdbfeexample",  
    "SubnetId": "subnet-ba47exam",  
    "Ip": "192.0.2.44",  
    "Status": "ATTACHED",  
    "StatusMessage": "This IP address is operational.",  
    "CreationTime": "2020-01-03T23:02:29.587Z",  
    "ModificationTime": "2020-01-03T23:03:05.555Z"  
  },  
  {  
    "IpId": "rni-aac7085e38example",  
    "SubnetId": "subnet-12d8exam",  
    "Ip": "192.0.2.45",  
    "Status": "ATTACHED",  
    "StatusMessage": "This IP address is operational.",  
    "CreationTime": "2020-01-03T23:02:29.593Z",  
    "ModificationTime": "2020-01-03T23:02:55.060Z"  
  }  
]  
}
```

有关输出中值的更多信息，请参阅 Amazon Route 53 开发者指南中[创建或编辑入站终端节点时指定的值以及创建或编辑出站终端节点时指定的值](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListResolverEndpointIpAddresses](#)中的。

## list-resolver-endpoints

以下代码示例演示了如何使用 `list-resolver-endpoints`。

### AWS CLI

列出某个 AWS 区域中的解析器终端节点

以下 `list-resolver-endpoints` 示例列出了当前账户中存在的入站和出站 Resolver 终端节点。

```
aws route53resolver list-resolver-endpoints
```

输出：

```
{  
  "MaxResults": 10,  
}
```

```
"ResolverEndpoints": [
  {
    "Id": "rslvr-in-497098ad59example",
    "CreatorRequestId": "2020-01-01-18:47",
    "Arn": "arn:aws:route53resolver:us-west-2:111122223333:resolver-
endpoint/rslvr-in-497098ad59example",
    "Name": "my-inbound-endpoint",
    "SecurityGroupIds": [
      "sg-05cd7b25d6example"
    ],
    "Direction": "INBOUND",
    "IpAddressCount": 2,
    "HostVPCId": "vpc-304bexam",
    "Status": "OPERATIONAL",
    "StatusMessage": "This Resolver Endpoint is operational.",
    "CreationTime": "2020-01-01T23:25:45.538Z",
    "ModificationTime": "2020-01-01T23:25:45.538Z"
  },
  {
    "Id": "rslvr-out-d5e5920e37example",
    "CreatorRequestId": "2020-01-01-18:48",
    "Arn": "arn:aws:route53resolver:us-west-2:111122223333:resolver-
endpoint/rslvr-out-d5e5920e37example",
    "Name": "my-outbound-endpoint",
    "SecurityGroupIds": [
      "sg-05cd7b25d6example"
    ],
    "Direction": "OUTBOUND",
    "IpAddressCount": 2,
    "HostVPCId": "vpc-304bexam",
    "Status": "OPERATIONAL",
    "StatusMessage": "This Resolver Endpoint is operational.",
    "CreationTime": "2020-01-01T23:50:50.979Z",
    "ModificationTime": "2020-01-01T23:50:50.979Z"
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResolverEndpoints](#)中的。

## list-resolver-rule-associations

以下代码示例演示了如何使用 list-resolver-rule-associations。

## AWS CLI

### 列出解析器规则和 VPC 之间的关联

以下 `list-resolver-rule-associations` 示例列出了当前 AWS 账户中解析器规则与 VPC 之间的关联。

```
aws route53resolver list-resolver-rule-associations
```

输出：

```
{
  "MaxResults": 30,
  "ResolverRuleAssociations": [
    {
      "Id": "rslvr-autodefined-assoc-vpc-304bexam-internet-resolver",
      "ResolverRuleId": "rslvr-autodefined-rr-internet-resolver",
      "Name": "System Rule Association",
      "VPCId": "vpc-304bexam",
      "Status": "COMPLETE",
      "StatusMessage": ""
    },
    {
      "Id": "rslvr-rrassoc-d61cbb2c8bexample",
      "ResolverRuleId": "rslvr-rr-42b60677c0example",
      "Name": "my-resolver-rule-association",
      "VPCId": "vpc-304bexam",
      "Status": "COMPLETE",
      "StatusMessage": ""
    }
  ]
}
```

有关更多信息，请参阅《亚马逊 Route 53 开发者指南》中的 [Route 53 解析器如何将 DNS 查询从您的 VPC 转发到您的网络](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListResolverRuleAssociations](#) 中的。

### **list-resolver-rules**

以下代码示例演示了如何使用 `list-resolver-rules`。

## AWS CLI

### 列出解析器规则

以下`list-resolver-rules`示例列出了当前 AWS 账户中的所有 Resolver 规则。

```
aws route53resolver list-resolver-rules
```

输出：

```
{
  "MaxResults": 30,
  "ResolverRules": [
    {
      "Id": "rslvr-autodefined-rr-internet-resolver",
      "CreatorRequestId": "",
      "Arn": "arn:aws:route53resolver:us-west-2::autodefined-rule/rslvr-
autodefined-rr-internet-resolver",
      "DomainName": ".",
      "Status": "COMPLETE",
      "RuleType": "RECURSIVE",
      "Name": "Internet Resolver",
      "OwnerId": "Route 53 Resolver",
      "ShareStatus": "NOT_SHARED"
    },
    {
      "Id": "rslvr-rr-42b60677c0example",
      "CreatorRequestId": "2020-01-01-18:47",
      "Arn": "arn:aws:route53resolver:us-west-2:111122223333:resolver-rule/
rslvr-rr-42b60677c0bc4e299",
      "DomainName": "example.com.",
      "Status": "COMPLETE",
      "StatusMessage": "[Trace id: 1-5dc4b177-ff1d9d001a0f80005example]
Successfully created Resolver Rule.",
      "RuleType": "FORWARD",
      "Name": "my-rule",
      "TargetIps": [
        {
          "Ip": "192.0.2.45",
          "Port": 53
        }
      ],
      "ResolverEndpointId": "rslvr-out-d5e5920e37example",
    }
  ]
}
```

```
        "OwnerId": "111122223333",
        "ShareStatus": "NOT_SHARED"
    }
]
}
```

有关更多信息，请参阅《亚马逊 Route 53 开发者指南》中的 [Route 53 解析器如何将 DNS 查询从您的 VPC 转发到您的网络](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListResolverRules](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

列出解析器资源的标签

以下 `list-tags-for-resource` 示例列出了分配给指定解析器规则的标签。

```
aws route53resolver list-tags-for-resource \
  --resource-arn "arn:aws:route53resolver:us-west-2:111122223333:resolver-rule/
  rslvr-rr-42b60677c0example"
```

输出：

```
{
  "Tags": [
    {
      "Key": "my-key-1",
      "Value": "my-value-1"
    },
    {
      "Key": "my-key-2",
      "Value": "my-value-2"
    }
  ]
}
```

有关使用标签进行成本分配的信息，请参阅《Billing and Cost Management 用户指南》中的 [使用成本分配标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## put-firewall-rule-group-policy

以下代码示例演示了如何使用 put-firewall-rule-group-policy。

### AWS CLI

附加 AWS IAM 策略以共享防火墙规则组策略

以下put-firewall-rule-group-policy示例附加了用于共享规则组的 I AWS identity and Access Management (AWS IAM) 策略。

```
aws route53resolver put-firewall-rule-group-policy \
  --firewall-rule-group-policy "{\"Version\":\"2012-10-17\",
  \"Statement\": [{\"Sid\":\"test\", \"Effect\":\"Allow\", \"Principal
  \": {\"AWS\": \"arn:aws:iam::AWS_ACCOUNT_ID:root\"}, \"Action\":
  [\"route53resolver:GetFirewallRuleGroup\", \"route53resolver:ListFirewallRuleGroups
  \"], \"Resource\": \"arn:aws:route53resolver:us-east-1:AWS_ACCOUNT_ID:firewall-rule-
  group/rslvr-frg-47f93271fexample\"}]}"
```

输出：

```
{
  "ReturnValue": true
}
```

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[管理 DNS 防火墙中的规则组和规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutFirewallRuleGroupPolicy](#)中的。

## put-resolver-rule-policy

以下代码示例演示了如何使用 put-resolver-rule-policy。

### AWS CLI

与其他账户共享 Resolver 规则 AWS

以下put-resolver-rule-policy示例指定了您要与其他 AWS 账户共享的 Resolver 规则、要与之共享规则的账户，以及您希望该账户能够对规则执行的与规则相关的操作。



注意您必须使用创建规则的另一账户的凭据运行此命令。

```
aws route53resolver put-resolver-rule-policy \
  --region us-east-1 \
  --arn "arn:aws:route53resolver:us-east-1:111122223333:resolver-rule/rslvr-
rr-42b60677c0example" \
  --resolver-rule-policy "{\"Version\": \"2012-10-17\", \
  \"Statement\": [ { \
  \"Effect\" : \"Allow\", \
  \"Principal\" : {\"AWS\" : \"444455556666\" }, \
  \"Action\" : [ \
  \"route53resolver:GetResolverRule\", \
  \"route53resolver:AssociateResolverRule\", \
  \"route53resolver:DisassociateResolverRule\", \
  \"route53resolver:ListResolverRules\", \
  \"route53resolver:ListResolverRuleAssociations\" ], \
  \"Resource\" : [ \"arn:aws:route53resolver:us-east-1:111122223333:resolver-
rule/rslvr-rr-42b60677c0example\" ] } ] }"
```

输出：

```
{
  "ReturnValue": true
}
```

运行后`put-resolver-rule-policy`，您可以运行以下两个 Resource Access Manager (RAM) 命令。您必须使用要与之共享规则的账户：

`get-resource-share-invitations`返回值`resourceShareInvitationArn`。您需要此值才能接受使用共享规则的邀请。`accept-resource-share-invitation`接受使用共享规则的邀请。

有关更多信息，请参阅 文档：

[get-resource-share-invitationsaccept-resource-share-invitations](#) Amazon Route 53 开发者指南中的 [与其他 AWS 账户共享转发规则并使用共享规则](#)

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutResolverRulePolicy](#) 中的。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

## AWS CLI

将标签与解析器资源相关联

以下tag-resource示例将两个标签键/值对与指定的 Resolver 规则关联起来。

```
aws route53resolver tag-resource \  
  --resource-arn "arn:aws:route53resolver:us-west-2:111122223333:resolver-rule/  
rslvr-rr-42b60677c0example" \  
  --tags "Key=my-key-1,Value=my-value-1" "Key=my-key-2,Value=my-value-2"
```

此命令不生成任何输出。

有关使用标签进行成本分配的信息，请参阅《B AWS Billing and Cost Management 用户指南》中的使用成本[分配标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

## AWS CLI

从解析器资源中移除标签

以下untag-resource示例从指定的 Resolver 规则中删除两个标签。

```
aws route53resolver untag-resource \  
  --resource-arn "arn:aws:route53resolver:us-west-2:111122223333:resolver-rule/  
rslvr-rr-42b60677c0example" \  
  --tag-keys my-key-1 my-key-2
```

此命令不生成任何输出。要确认标签已被删除，可以使用[list-tags-for-resource](#)。

有关使用标签进行成本分配的信息，请参阅《B AWS Billing and Cost Management 用户指南》中的使用成本[分配标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-firewall-config

以下代码示例演示了如何使用 update-firewall-config。

## AWS CLI

### 更新防火墙配置

以下update-firewall-config示例更新 DNS 防火墙配置。

```
aws route53resolver update-firewall-config \  
  --resource-id vpc-31e92222 \  
  --firewall-fail-open DISABLED
```

输出：

```
{  
  "FirewallConfig": {  
    "Id": "rslvr-fc-86016850cexample",  
    "ResourceId": "vpc-31e92222",  
    "OwnerId": "123456789012",  
    "FirewallFailOpen": "DISABLED"  
  }  
}
```

有关更多信息，请参阅《亚马逊 Route 53 开发者指南》中的 [DNS 防火墙 VPC 配置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateFirewallConfig](#)中的。

### update-firewall-domains

以下代码示例演示了如何使用 update-firewall-domains。

## AWS CLI

### 更新域名列表

以下update-firewall-domains示例使用您提供的 ID 将域名添加到域名列表中。

```
aws route53resolver update-firewall-domains \  
  --firewall-domain-list-id rslvr-fdl-42b60677cexampleb \  
  --operation ADD \  
  --domains test1.com test2.com test3.com
```

输出：

```
{
  "Id": "rslvr-fdl-42b60677cexample",
  "Name": "test",
  "Status": "UPDATING",
  "StatusMessage": "Updating the Firewall Domain List"
}
```

有关更多信息，请参阅《Amazon Route 53 开发者指南》中的[管理自己的域名列表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateFirewallDomains](#)中的。

## update-firewall-rule-group-association

以下代码示例演示了如何使用 update-firewall-rule-group-association。

### AWS CLI

#### 更新防火墙规则组关联

以下update-firewall-rule-group-association示例更新了防火墙规则组关联。

```
aws route53resolver update-firewall-rule-group-association \
  --firewall-rule-group-association-id rslvr-frgassoc-57e8873d7example \
  --priority 103
```

#### 输出：

```
{
  "FirewallRuleGroupAssociation": {
    "Id": "rslvr-frgassoc-57e8873d7example",
    "Arn": "arn:aws:route53resolver:us-west-2:123456789012:firewall-rule-group-association/rslvr-frgassoc-57e8873d7example",
    "FirewallRuleGroupId": "rslvr-frg-47f93271fexample",
    "VpcId": "vpc-31e92222",
    "Name": "test-association",
    "Priority": 103,
    "MutationProtection": "DISABLED",
    "Status": "UPDATING",
    "StatusMessage": "Updating the Firewall Rule Group Association Attributes",
    "CreatorRequestId": "2ca1a304-32b3-4f5f-bc4c-EXAMPLE11111",
    "CreationTime": "2021-05-25T21:47:48.755768Z",
  }
}
```

```
    "ModificationTime": "2021-05-25T21:50:09.272569Z"
  }
}
```

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[管理您的 VPC 和 Route 53 Resolver DNS 防火墙规则组之间的关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateFirewallRuleGroupAssociation](#) 中的。

## update-firewall-rule

以下代码示例演示了如何使用 update-firewall-rule。

### AWS CLI

#### 更新防火墙规则

以下 update-firewall-rule 示例使用您指定的参数更新防火墙规则。

```
aws route53resolver update-firewall-rule \
  --firewall-rule-group-id rslvr-frg-47f93271fexample \
  --firewall-domain-list-id rslvr-fdl-9e956e9ffexample \
  --priority 102
```

输出：

```
{
  "FirewallRule": {
    "FirewallRuleGroupId": "rslvr-frg-47f93271fexample",
    "FirewallDomainListId": "rslvr-fdl-9e956e9ffexample",
    "Name": "allow-rule",
    "Priority": 102,
    "Action": "ALLOW",
    "CreatorRequestId": "d81e3fb7-020b-415e-939f-EXAMPLE11111",
    "CreationTime": "2021-05-25T21:44:00.346093Z",
    "ModificationTime": "2021-05-25T21:45:59.611600Z"
  }
}
```

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[管理 DNS 防火墙中的规则组和规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateFirewallRule](#) 中的。

## update-resolver-endpoint

以下代码示例演示了如何使用 `update-resolver-endpoint`。

### AWS CLI

#### 更新解析器端点的名称

以下 `update-resolver-endpoint` 示例更新了解析器端点的名称。不支持更新其他值。

```
aws route53resolver update-resolver-endpoint \  
  --resolver-endpoint-id rslvr-in-b5d45e32bdc445f09 \  
  --name my-renamed-inbound-endpoint
```

#### 输出：

```
{  
  "ResolverEndpoint": {  
    "Id": "rslvr-in-b5d45e32bdexample",  
    "CreatorRequestId": "2020-01-02-18:48",  
    "Arn": "arn:aws:route53resolver:us-west-2:111122223333:resolver-endpoint/  
rslvr-in-b5d45e32bdexample",  
    "Name": "my-renamed-inbound-endpoint",  
    "SecurityGroupIds": [  
      "sg-f62bexam"  
    ],  
    "Direction": "INBOUND",  
    "IpAddressCount": 2,  
    "HostVPCId": "vpc-304bexam",  
    "Status": "OPERATIONAL",  
    "StatusMessage": "This Resolver Endpoint is operational.",  
    "CreationTime": "2020-01-01T18:33:59.265Z",  
    "ModificationTime": "2020-01-08T18:33:59.265Z"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateResolverEndpoint](#)中的。

## update-resolver-rule

以下代码示例演示了如何使用 `update-resolver-rule`。

## AWS CLI

### 示例 1：更新设置解析器端点

以下update-resolver-rule示例更新了规则的名称、将 DNS 查询转发到的本地网络上的 IP 地址以及用于将查询转发到网络的出站 Resolver 端点的 ID。

注意的现有值TargetIps会被覆盖，因此您必须指定更新后希望规则拥有的所有 IP 地址。

```
aws route53resolver update-resolver-rule \  
  --resolver-rule-id rslvr-rr-1247fa64f3example \  
  --config Name="my-2nd-rule",TargetIps=[{Ip=192.0.2.45,Port=53},  
{Ip=192.0.2.46,Port=53}],ResolverEndpointId=rslvr-out-7b89ed0d25example
```

输出：

```
{  
  "ResolverRule": {  
    "Id": "rslvr-rr-1247fa64f3example",  
    "CreatorRequestId": "2020-01-02-18:47",  
    "Arn": "arn:aws:route53resolver:us-west-2:111122223333:resolver-rule/rslvr-  
rr-1247fa64f3example",  
    "DomainName": "www.example.com.",  
    "Status": "COMPLETE",  
    "StatusMessage": "[Trace id: 1-5dcc90b9-8a8ee860aba1ebd89example]  
Successfully updated Resolver Rule.",  
    "RuleType": "FORWARD",  
    "Name": "my-2nd-rule",  
    "TargetIps": [  
      {  
        "Ip": "192.0.2.45",  
        "Port": 53  
      },  
      {  
        "Ip": "192.0.2.46",  
        "Port": 53  
      }  
    ],  
    "ResolverEndpointId": "rslvr-out-7b89ed0d25example",  
    "OwnerId": "111122223333",  
    "ShareStatus": "NOT_SHARED"  
  }  
}
```

## 示例 2：使用文件进行“配置”设置更新解析器端点

或者，您可以将config设置包含在JSON文件中，然后在调用时指定该文件update-resolver-rule。

```
aws route53resolver update-resolver-rule \  
  --resolver-rule-id rslvr-rr-1247fa64f3example \  
  --config file://c:\temp\update-resolver-rule.json
```

update-resolver-rule.json 的内容。

```
{  
  "Name": "my-2nd-rule",  
  "TargetIps": [  
    {  
      "Ip": "192.0.2.45",  
      "Port": 53  
    },  
    {  
      "Ip": "192.0.2.46",  
      "Port": 53  
    }  
  ],  
  "ResolverEndpointId": "rslvr-out-7b89ed0d25example"  
}
```

有关更多信息，请参阅 Amazon Route 53 开发者指南中的[您在创建或编辑规则时指定的值](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateResolverRule](#)中的。

## 使用 Amazon S3 的示例 AWS CLI

以下代码示例向您展示了如何在 Amazon S3 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。



## 主题

- [操作](#)

## 操作

### **abort-multipart-upload**

以下代码示例演示了如何使用 `abort-multipart-upload`。

#### AWS CLI

中止指定的分段上传

以下 `abort-multipart-upload` 命令中止存储桶 `my-bucket` 中键 `multipart/01` 的分段上传：

```
aws s3api abort-multipart-upload \  
  --bucket my-bucket \  
  --key multipart/01 \  
  --upload-id  
dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3URCS
```

此命令所需的上传 ID 由 `create-multipart-upload` 输出，也可以使用 `list-multipart-uploads` 进行检索。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AbortMultipartUpload](#) 中的。

### **complete-multipart-upload**

以下代码示例演示了如何使用 `complete-multipart-upload`。

#### AWS CLI

以下命令完成存储桶 `my-bucket` 中密钥 `multipart/01` 的分段上传：

```
aws s3api complete-multipart-upload --multipart-upload file://  
mpustruct --bucket my-bucket --key 'multipart/01' --upload-id  
dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3URCS
```

此命令所需的上传 ID 由 `create-multipart-upload` 输出，也可以使用 `list-multipart-uploads` 进行检索。

上述命令中的分段上传选项采用 JSON 结构，用于描述分段上传中应重新组合成完整文件的各个部分。在此示例中，file:// 前缀用于从名为 mpustruct 的本地文件夹中的文件加载 JSON 结构。

mpustruct :

```
{
  "Parts": [
    {
      "ETag": "e868e0f4719e394144ef36531ee6824c",
      "PartNumber": 1
    },
    {
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",
      "PartNumber": 2
    },
    {
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",
      "PartNumber": 3
    }
  ]
}
```

每次使用 upload-part 命令上传分段时，都会输出每个分段的 ETag 值，也可以通过调用 list-parts 来检索，或者通过对每个分段进行 MD5 校验和执行计算。

输出：

```
{
  "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",
  "Bucket": "my-bucket",
  "Location": "https://my-bucket.s3.amazonaws.com/multipart%2F01",
  "Key": "multipart/01"
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CompleteMultipartUpload](#) 中的。

## copy-object

以下代码示例演示了如何使用 copy-object。

## AWS CLI

以下命令将对象从 bucket-1 复制到 bucket-2 :

```
aws s3api copy-object --copy-source bucket-1/test.txt --key test.txt --bucket
bucket-2
```

输出 :

```
{
  "CopyObjectResult": {
    "LastModified": "2015-11-10T01:07:25.000Z",
    "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""
  },
  "VersionId": "YdnYvTCVDqRRFA.NFJjy36p0hxifM1kA"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CopyObject](#)中的。

## cp

以下代码示例演示了如何使用 cp。

### AWS CLI

示例 1：将本地文件复制到 S3

以下cp命令将单个文件复制到指定的存储桶和密钥：

```
aws s3 cp test.txt s3://mybucket/test2.txt
```

输出：

```
upload: test.txt to s3://mybucket/test2.txt
```

示例 2：将带有过期日期的本地文件复制到 S3

以下cp命令将单个文件复制到指定的存储桶和密钥，该文件将在指定的 ISO 8601 时间戳过期：

```
aws s3 cp test.txt s3://mybucket/test2.txt \
--expires 2014-10-01T20:30:00Z
```

输出：

```
upload: test.txt to s3://mybucket/test2.txt
```

示例 3：将文件从 S3 复制到 S3

以下cp命令将单个 s3 对象复制到指定的存储桶和密钥：

```
aws s3 cp s3://mybucket/test.txt s3://mybucket/test2.txt
```

输出：

```
copy: s3://mybucket/test.txt to s3://mybucket/test2.txt
```

示例 4：将 S3 对象复制到本地文件

以下cp命令将单个对象复制到本地的指定文件中：

```
aws s3 cp s3://mybucket/test.txt test2.txt
```

输出：

```
download: s3://mybucket/test.txt to test2.txt
```

示例 5：将 S3 对象从一个存储桶复制到另一个存储桶

以下cp命令将单个对象复制到指定的存储桶，同时保留其原始名称：

```
aws s3 cp s3://mybucket/test.txt s3://mybucket2/
```

输出：

```
copy: s3://mybucket/test.txt to s3://mybucket2/test.txt
```

示例 6：以递归方式将 S3 对象复制到本地目录

当与参数一起传递时--recursive，以下cp命令会递归地将指定前缀和存储桶下的所有对象复制到指定目录。在此示例中mybucket，存储桶包含对象test1.txt和test2.txt：

```
aws s3 cp s3://mybucket . \
```

```
--recursive
```

输出：

```
download: s3://mybucket/test1.txt to test1.txt
download: s3://mybucket/test2.txt to test2.txt
```

示例 7：以递归方式将本地文件复制到 S3

当与参数一起传递时`--recursive`，以下`cp`命令会递归地将指定目录下的所有文件复制到指定的存储桶和前缀，同时使用`--exclude`参数排除某些文件。在此示例中，`myDir`该目录包含文件`test1.txt`和`test2.jpg`：

```
aws s3 cp myDir s3://mybucket/ \
  --recursive \
  --exclude "*.jpg"
```

输出：

```
upload: myDir/test1.txt to s3://mybucket/test1.txt
```

示例 8：以递归方式将 S3 对象复制到另一个存储桶

当与参数一起传递时`--recursive`，以下`cp`命令以递归方式将指定存储桶下的所有对象复制到另一个存储桶，同时使用`--exclude`参数排除某些对象。在此示例中`mybucket`，存储桶包含对象`test1.txt`和`another/test1.txt`：

```
aws s3 cp s3://mybucket/ s3://mybucket2/ \
  --recursive \
  --exclude "another/*"
```

输出：

```
copy: s3://mybucket/test1.txt to s3://mybucket2/test1.txt
```

您可以组合`--exclude`和`--include`选项，仅复制与模式匹配的对象，不包括所有其他对象：

```
aws s3 cp s3://mybucket/logs/ s3://mybucket2/logs/ \
  --recursive \
  --exclude "*" \
```

```
--include "*.log"
```

输出：

```
copy: s3://mybucket/logs/test/test.log to s3://mybucket2/logs/test/test.log
copy: s3://mybucket/logs/test3.log to s3://mybucket2/logs/test3.log
```

示例 9：在复制 S3 对象时设置访问控制列表 (ACL)

以下cp命令将单个对象复制到指定的存储桶和密钥，同时将 ACL 设置为public-read-write：

```
aws s3 cp s3://mybucket/test.txt s3://mybucket/test2.txt \
--acl public-read-write
```

输出：

```
copy: s3://mybucket/test.txt to s3://mybucket/test2.txt
```

请注意，如果您使用的是该--acl选项，请确保所有关联的 IAM 策略都包含以下"s3:PutObjectAcl"操作：

```
aws iam get-user-policy \
--user-name myuser \
--policy-name mypolicy
```

输出：

```
{
  "UserName": "myuser",
  "PolicyName": "mypolicy",
  "PolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": [
          "s3:PutObject",
          "s3:PutObjectAcl"
        ],
        "Resource": [
          "arn:aws:s3:::mybucket/*"
        ]
      }
    ]
  }
}
```

```
        "Effect": "Allow",
        "Sid": "Stmt1234567891234"
    }
  ]
}
```

#### 示例 10：为 S3 对象授予权限

以下cp命令说明了如何使用该--grants选项向由 URI 标识的所有用户授予读取权限，并向通过其 Canonical ID 标识的特定用户授予完全控制权：

```
aws s3 cp file.txt s3://mybucket/ --grants read=uri=http://acs.amazonaws.com/groups/global/AllUsers
full=id=79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

输出：

```
upload: file.txt to s3://mybucket/file.txt
```

#### 示例 11：将本地文件流上传到 S3

PowerShell 可能会更改管道输入的编码或向管道输入添加 CRLF。

以下cp命令将本地文件流从标准输入上传到指定的存储桶和密钥：

```
aws s3 cp - s3://mybucket/stream.txt
```

#### 示例 12：将大于 50GB 的本地文件流上传到 S3

以下cp命令将 51GB 的本地文件流从标准输入上传到指定的存储桶和密钥。必须提供该--expected-size选项，否则当上传达到默认分段限制为 10,000 时，上传可能会失败：

```
aws s3 cp - s3://mybucket/stream.txt --expected-size 54760833024
```

#### 示例 13：将 S3 对象下载为本地文件流

PowerShell 可能会更改管道输出或重定向输出的编码或添加 CRLF。

以下cp命令将 S3 对象作为流下载到本地标准输出。作为直播下载目前与以下--recursive参数不兼容：

```
aws s3 cp s3://mybucket/stream.txt -
```

#### 示例 14 : 上传到 S3 接入点

以下cp命令将单个文件 (mydoc.txt) 上传到密钥 (myaccesspoint) 处的接入点 (mykey) :

```
aws s3 cp mydoc.txt s3://arn:aws:s3:us-west-2:123456789012:accesspoint/
myaccesspoint/mykey
```

输出 :

```
upload: mydoc.txt to s3://arn:aws:s3:us-west-2:123456789012:accesspoint/
myaccesspoint/mykey
```

#### 示例 15 : 从 S3 接入点下载

以下cp命令将单个对象 (mykey) 从接入点 (myaccesspoint) 下载到本地文件 (mydoc.txt) :

```
aws s3 cp s3://arn:aws:s3:us-west-2:123456789012:accesspoint/myaccesspoint/mykey
mydoc.txt
```

输出 :

```
download: s3://arn:aws:s3:us-west-2:123456789012:accesspoint/myaccesspoint/mykey to
mydoc.txt
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [Cp](#)。

## create-bucket

以下代码示例演示了如何使用 create-bucket。

### AWS CLI

#### 示例 1 : 创建存储桶

以下 create-bucket 示例创建一个名为 my-bucket 的存储桶 :

```
aws s3api create-bucket \
  --bucket my-bucket \
```



```
--region us-east-1
```

输出：

```
{
  "Location": "/my-bucket"
}
```

有关更多信息，请参阅《Amazon S3 用户指南》中的[创建存储桶](#)。

示例 2：创建带有强制拥有者的存储桶

以下 create-bucket 示例创建一个名为 my-bucket 的存储桶，该存储桶对于 S3 对象所有权使用强制存储桶所有者设置。

```
aws s3api create-bucket \
  --bucket my-bucket \
  --region us-east-1 \
  --object-ownership BucketOwnerEnforced
```

输出：

```
{
  "Location": "/my-bucket"
}
```

有关更多信息，请参阅《Amazon S3 用户指南》中的[控制对象所有权和禁用 ACL](#)。

示例 3：在“us-east-1”区域之外创建存储桶

以下 create-bucket 示例在 eu-west-1 区域中创建名为 my-bucket 的存储桶。us-east-1 之外的区域需要指定相应的 LocationConstraint 才能在所需区域创建存储桶。

```
aws s3api create-bucket \
  --bucket my-bucket \
  --region eu-west-1 \
  --create-bucket-configuration LocationConstraint=eu-west-1
```

输出：

```
{
  "Location": "http://my-bucket.s3.amazonaws.com/"
}
```

```
}
```

有关更多信息，请参阅《Amazon S3 用户指南》中的[创建存储桶](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateBucket](#)中的。

## create-multipart-upload

以下代码示例演示了如何使用 create-multipart-upload。

### AWS CLI

以下命令在存储桶 my-bucket 中创建带有密钥 multipart/01 的分段上传：

```
aws s3api create-multipart-upload --bucket my-bucket --key 'multipart/01'
```

输出：

```
{
  "Bucket": "my-bucket",
  "UploadId":
  "dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3URC
  "Key": "multipart/01"
}
```

完成的文件在存储桶 my-bucket 中名为 multipart 文件夹中将命名为 01。保存上传 ID、密钥和存储桶名称以供 upload-part 命令使用。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateMultipartUpload](#)中的。

## delete-bucket-analytics-configuration

以下代码示例演示了如何使用 delete-bucket-analytics-configuration。

### AWS CLI

删除存储桶的分析配置

以下 delete-bucket-analytics-configuration 示例移除指定存储桶和 ID 的分析配置。

```
aws s3api delete-bucket-analytics-configuration \
  --bucket my-bucket \
```

```
--id 1
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketAnalyticsConfiguration](#)中的。

## **delete-bucket-cors**

以下代码示例演示了如何使用 delete-bucket-cors。

### AWS CLI

以下命令从名为 my-bucket 的存储桶中删除跨源资源共享配置：

```
aws s3api delete-bucket-cors --bucket my-bucket
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketCors](#)中的。

## **delete-bucket-encryption**

以下代码示例演示了如何使用 delete-bucket-encryption。

### AWS CLI

删除存储桶的服务器端加密配置

以下 delete-bucket-encryption 示例删除指定存储桶的服务器端加密配置。

```
aws s3api delete-bucket-encryption \  
  --bucket my-bucket
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketEncryption](#)中的。

## **delete-bucket-intelligent-tiering-configuration**

以下代码示例演示了如何使用 delete-bucket-intelligent-tiering-configuration。

### AWS CLI

移除存储桶上的 S3 智能分层配置

以下delete-bucket-intelligent-tiering-configuration示例删除了存储桶上名ExampleConfig为的 S3 智能分层配置。

```
aws s3api delete-bucket-intelligent-tiering-configuration \  
  --bucket DOC-EXAMPLE-BUCKET \  
  --id ExampleConfig
```

此命令不生成任何输出。

有关更多信息，请参阅 [Amazon S3 用户指南中的使用 S3 智能分层](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketIntelligentTieringConfiguration](#)中的。

## delete-bucket-inventory-configuration

以下代码示例演示了如何使用 delete-bucket-inventory-configuration。

### AWS CLI

删除存储桶的清单配置

以下 delete-bucket-inventory-configuration 示例删除指定存储桶的 ID 为 1 的清单配置。

```
aws s3api delete-bucket-inventory-configuration \  
  --bucket my-bucket \  
  --id 1
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketInventoryConfiguration](#)中的。

## delete-bucket-lifecycle

以下代码示例演示了如何使用 delete-bucket-lifecycle。

### AWS CLI

以下命令从名为 my-bucket 的存储桶中删除生命周期配置：

```
aws s3api delete-bucket-lifecycle --bucket my-bucket
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketLifecycle](#)中的。

## delete-bucket-metrics-configuration

以下代码示例演示了如何使用 delete-bucket-metrics-configuration。

### AWS CLI

#### 删除存储桶的指标配置

以下 delete-bucket-metrics-configuration 示例移除指定存储桶和 ID 的指标配置。

```
aws s3api delete-bucket-metrics-configuration \  
  --bucket my-bucket \  
  --id 123
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketMetricsConfiguration](#)中的。

## delete-bucket-ownership-controls

以下代码示例演示了如何使用 delete-bucket-ownership-controls。

### AWS CLI

#### 移除存储桶的存储桶所有权设置

以下delete-bucket-ownership-controls示例删除了存储桶的存储桶所有权设置。

```
aws s3api delete-bucket-ownership-controls \  
  --bucket DOC-EXAMPLE-BUCKET
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon S3 用户指南中的[为现有存储桶设置对象所有权](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketOwnershipControls](#)中的。

## delete-bucket-policy

以下代码示例演示了如何使用 delete-bucket-policy。

## AWS CLI

以下命令从名为 my-bucket 的存储桶中删除存储桶策略：

```
aws s3api delete-bucket-policy --bucket my-bucket
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketPolicy](#)中的。

### **delete-bucket-replication**

以下代码示例演示了如何使用 delete-bucket-replication。

## AWS CLI

以下命令从名为 my-bucket 的存储桶中删除复制配置：

```
aws s3api delete-bucket-replication --bucket my-bucket
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketReplication](#)中的。

### **delete-bucket-tagging**

以下代码示例演示了如何使用 delete-bucket-tagging。

## AWS CLI

以下命令从名为 my-bucket 的存储桶中删除标记配置：

```
aws s3api delete-bucket-tagging --bucket my-bucket
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketTagging](#)中的。

### **delete-bucket-website**

以下代码示例演示了如何使用 delete-bucket-website。

## AWS CLI

以下命令从名为 my-bucket 的存储桶中删除网站配置：

```
aws s3api delete-bucket-website --bucket my-bucket
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucketWebsite](#)中的。

## delete-bucket

以下代码示例演示了如何使用 delete-bucket。

### AWS CLI

以下命令删除名为 my-bucket 的存储桶：

```
aws s3api delete-bucket --bucket my-bucket --region us-east-1
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucket](#)中的。

## delete-object-tagging

以下代码示例演示了如何使用 delete-object-tagging。

### AWS CLI

删除对象的标签集

以下 delete-object-tagging 示例从对象 doc1.rtf 中删除带有指定键的标签。

```
aws s3api delete-object-tagging \  
  --bucket my-bucket \  
  --key doc1.rtf
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteObjectTagging](#)中的。

## delete-object

以下代码示例演示了如何使用 delete-object。

### AWS CLI

以下命令从名为 my-bucket 的存储桶中删除名为 test.txt 的对象：

```
aws s3api delete-object --bucket my-bucket --key test.txt
```

如果启用了存储桶版本控制，则输出将包含删除标记的版本 ID：

```
{
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tNDlWlGq",
  "DeleteMarker": true
}
```

有关删除对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“删除对象”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteObject](#)中的。

## delete-objects

以下代码示例演示了如何使用 delete-objects。

### AWS CLI

以下命令从名为 my-bucket 的存储桶中删除对象：

```
aws s3api delete-objects --bucket my-bucket --delete file://delete.json
```

delete.json 是当前目录中指定要删除的对象的 JSON 文档：

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

输出：

```
{
  "Deleted": [
    {
      "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUP01koHwzU",
      "Key": "test1.txt",

```



```
        "DeleteMarker": true
      }
    ]
  }
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteObjects](#)中的。

## delete-public-access-block

以下代码示例演示了如何使用 delete-public-access-block。

### AWS CLI

删除存储桶的屏蔽公共访问权限配置

以下 delete-public-access-block 示例移除指定存储桶上的屏蔽公共访问权限配置。

```
aws s3api delete-public-access-block \
  --bucket my-bucket
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePublicAccessBlock](#)中的。

## get-bucket-accelerate-configuration

以下代码示例演示了如何使用 get-bucket-accelerate-configuration。

### AWS CLI

检索存储桶的加速配置

以下 get-bucket-accelerate-configuration 示例检索指定存储桶的加速配置。

```
aws s3api get-bucket-accelerate-configuration \
  --bucket my-bucket
```

输出：

```
{
  "Status": "Enabled"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketAccelerateConfiguration](#)中的。

## get-bucket-acl

以下代码示例演示了如何使用 `get-bucket-acl`。

### AWS CLI

以下命令检索名为 `my-bucket` 的存储桶的访问控制列表：

```
aws s3api get-bucket-acl --bucket my-bucket
```

输出：

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
      },
      "Permission": "FULL_CONTROL"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketAcl](#)中的。

## get-bucket-analytics-configuration

以下代码示例演示了如何使用 `get-bucket-analytics-configuration`。

### AWS CLI

检索具有特定 ID 的存储桶的分析配置

以下 `get-bucket-analytics-configuration` 示例显示了指定存储桶和 ID 的分析配置。

```
aws s3api get-bucket-analytics-configuration \  
  --bucket my-bucket \  
  --id 1
```

输出：

```
{  
  "AnalyticsConfiguration": {  
    "StorageClassAnalysis": {},  
    "Id": "1"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketAnalyticsConfiguration](#)中的。

## get-bucket-cors

以下代码示例演示了如何使用 `get-bucket-cors`。

### AWS CLI

以下命令检索名为 `my-bucket` 的存储桶的跨源资源共享配置：

```
aws s3api get-bucket-cors --bucket my-bucket
```

输出：

```
{  
  "CORSRules": [  
    {  
      "AllowedHeaders": [  
        "*"   
      ],  
      "ExposeHeaders": [  
        "x-amz-server-side-encryption"  
      ],  
      "AllowedMethods": [  
        "PUT",  
        "POST",  
        "DELETE"  
      ]  
    }  
  ]  
}
```

```
    ],
    "MaxAgeSeconds": 3000,
    "AllowedOrigins": [
      "http://www.example.com"
    ]
  },
  {
    "AllowedHeaders": [
      "Authorization"
    ],
    "MaxAgeSeconds": 3000,
    "AllowedMethods": [
      "GET"
    ],
    "AllowedOrigins": [
      "*"
    ]
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketCors](#)中的。

## get-bucket-encryption

以下代码示例演示了如何使用 get-bucket-encryption。

### AWS CLI

检索存储桶的服务器端加密配置

以下 get-bucket-encryption 示例检索存储桶 my-bucket 的服务器端加密配置。

```
aws s3api get-bucket-encryption \
  --bucket my-bucket
```

输出：

```
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
```

```
        "ApplyServerSideEncryptionByDefault": {
            "SSEAlgorithm": "AES256"
        }
    }
]
}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketEncryption](#)中的。

## get-bucket-intelligent-tiering-configuration

以下代码示例演示了如何使用 `get-bucket-intelligent-tiering-configuration`。

### AWS CLI

检索存储桶上的 S3 智能分层配置

以下`get-bucket-intelligent-tiering-configuration`示例检索存储桶上名 `ExampleConfig` 为的 S3 智能分层配置。

```
aws s3api get-bucket-intelligent-tiering-configuration \
  --bucket DOC-EXAMPLE-BUCKET \
  --id ExampleConfig
```

输出：

```
{
  "IntelligentTieringConfiguration": {
    "Id": "ExampleConfig2",
    "Filter": {
      "Prefix": "images"
    },
    "Status": "Enabled",
    "Tierings": [
      {
        "Days": 90,
        "AccessTier": "ARCHIVE_ACCESS"
      },
      {
        "Days": 180,
        "AccessTier": "DEEP_ARCHIVE_ACCESS"
      }
    ]
  }
}
```

```
    }  
  ]  
}  
}
```

有关更多信息，请参阅 [Amazon S3 用户指南中的使用 S3 智能分层](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetBucketIntelligentTieringConfiguration](#) 中的。

## get-bucket-inventory-configuration

以下代码示例演示了如何使用 `get-bucket-inventory-configuration`。

### AWS CLI

检索存储桶的清单配置

以下 `get-bucket-inventory-configuration` 示例检索 ID 为 1 的指定存储桶的清单配置。

```
aws s3api get-bucket-inventory-configuration \  
  --bucket my-bucket \  
  --id 1
```

输出：

```
{  
  "InventoryConfiguration": {  
    "IsEnabled": true,  
    "Destination": {  
      "S3BucketDestination": {  
        "Format": "ORC",  
        "Bucket": "arn:aws:s3:::my-bucket",  
        "AccountId": "123456789012"  
      }  
    },  
    "IncludedObjectVersions": "Current",  
    "Id": "1",  
    "Schedule": {  
      "Frequency": "Weekly"  
    }  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketInventoryConfiguration](#)中的。

## get-bucket-lifecycle-configuration

以下代码示例演示了如何使用 `get-bucket-lifecycle-configuration`。

### AWS CLI

以下命令检索名为 `my-bucket` 的存储桶的生命周期配置：

```
aws s3api get-bucket-lifecycle-configuration --bucket my-bucket
```

输出：

```
{
  "Rules": [
    {
      "ID": "Move rotated logs to Glacier",
      "Prefix": "rotated/",
      "Status": "Enabled",
      "Transitions": [
        {
          "Date": "2015-11-10T00:00:00.000Z",
          "StorageClass": "GLACIER"
        }
      ]
    },
    {
      "Status": "Enabled",
      "Prefix": "",
      "NoncurrentVersionTransitions": [
        {
          "NoncurrentDays": 0,
          "StorageClass": "GLACIER"
        }
      ],
      "ID": "Move old versions to Glacier"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketLifecycleConfiguration](#)中的。

## get-bucket-lifecycle

以下代码示例演示了如何使用 `get-bucket-lifecycle`。

### AWS CLI

以下命令检索名为 `my-bucket` 的存储桶的生命周期配置：

```
aws s3api get-bucket-lifecycle --bucket my-bucket
```

输出：

```
{
  "Rules": [
    {
      "ID": "Move to Glacier after sixty days (objects in logs/2015/)",
      "Prefix": "logs/2015/",
      "Status": "Enabled",
      "Transition": {
        "Days": 60,
        "StorageClass": "GLACIER"
      }
    },
    {
      "Expiration": {
        "Date": "2016-01-01T00:00:00.000Z"
      },
      "ID": "Delete 2014 logs in 2016.",
      "Prefix": "logs/2014/",
      "Status": "Enabled"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketLifecycle](#)中的。

## get-bucket-location

以下代码示例演示了如何使用 `get-bucket-location`。

### AWS CLI

如果存在约束条件，则以下命令会检索名为 `my-bucket` 的存储桶的位置约束：



```
aws s3api get-bucket-location --bucket my-bucket
```

输出：

```
{
  "LocationConstraint": "us-west-2"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketLocation](#)中的。

## get-bucket-logging

以下代码示例演示了如何使用 `get-bucket-logging`。

AWS CLI

检索存储桶的日志记录状态

以下 `get-bucket-logging` 示例检索指定存储桶的日志记录状态。

```
aws s3api get-bucket-logging \
  --bucket my-bucket
```

输出：

```
{
  "LoggingEnabled": {
    "TargetPrefix": "",
    "TargetBucket": "my-bucket-logs"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketLogging](#)中的。

## get-bucket-metrics-configuration

以下代码示例演示了如何使用 `get-bucket-metrics-configuration`。

AWS CLI

检索具有特定 ID 的存储桶的指标配置

以下 `get-bucket-metrics-configuration` 示例显示了指定存储桶和 ID 的指标配置。

```
aws s3api get-bucket-metrics-configuration \  
  --bucket my-bucket \  
  --id 123
```

输出：

```
{  
  "MetricsConfiguration": {  
    "Filter": {  
      "Prefix": "logs"  
    },  
    "Id": "123"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketMetricsConfiguration](#)中的。

## get-bucket-notification-configuration

以下代码示例演示了如何使用 `get-bucket-notification-configuration`。

AWS CLI

以下命令检索名为 `my-bucket` 的存储桶的通知配置：

```
aws s3api get-bucket-notification-configuration --bucket my-bucket
```

输出：

```
{  
  "TopicConfigurations": [  
    {  
      "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNI",  
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",  
      "Events": [  
        "s3:ObjectCreated:*"  
      ]  
    }  
  ]  
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketNotificationConfiguration](#)中的。

## get-bucket-notification

以下代码示例演示了如何使用 get-bucket-notification。

### AWS CLI

以下命令检索名为 my-bucket 的存储桶的通知配置：

```
aws s3api get-bucket-notification --bucket my-bucket
```

输出：

```
{
  "TopicConfiguration": {
    "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
    "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWw1",
    "Event": "s3:ObjectCreated:*",
    "Events": [
      "s3:ObjectCreated:*"
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketNotification](#)中的。

## get-bucket-ownership-controls

以下代码示例演示了如何使用 get-bucket-ownership-controls。

### AWS CLI

检索存储桶的存储桶所有权设置

以下get-bucket-ownership-controls示例检索存储桶的存储桶所有权设置。

```
aws s3api get-bucket-ownership-controls \
  --bucket DOC-EXAMPLE-BUCKET
```

输出：

```
{
  "OwnershipControls": {
    "Rules": [
      {
        "ObjectOwnership": "BucketOwnerEnforced"
      }
    ]
  }
}
```

有关更多信息，请参阅 Amazon S3 用户指南中的[查看 S3 存储桶的对象所有权设置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketOwnershipControls](#)中的。

## get-bucket-policy-status

以下代码示例演示了如何使用 get-bucket-policy-status。

### AWS CLI

检索存储桶的策略状态，此状态指示存储桶是否为公有存储桶

以下 get-bucket-policy-status 示例检索存储桶 my-bucket 的策略状态。

```
aws s3api get-bucket-policy-status \
  --bucket my-bucket
```

输出：

```
{
  "PolicyStatus": {
    "IsPublic": false
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketPolicyStatus](#)中的。

## get-bucket-policy

以下代码示例演示了如何使用 get-bucket-policy。

## AWS CLI

以下命令检索名为 `my-bucket` 的存储桶的存储桶策略：

```
aws s3api get-bucket-policy --bucket my-bucket
```

输出：

```
{
  "Policy": "{\n\"Version\": \"2008-10-17\", \"Statement\": [\n{\n\"Sid\": \"\", \"Effect\": \"Allow\", \"Principal\": \"*\", \"Action\": \"s3:GetObject\", \"Resource\": \"arn:aws:s3:::my-bucket/*\"}, {\n\"Sid\": \"\", \"Effect\": \"Deny\", \"Principal\": \"*\", \"Action\": \"s3:GetObject\", \"Resource\": \"arn:aws:s3:::my-bucket/secret/*\"}]]\"
}
```

**获取并放置存储桶策略** 以下示例演示了如何下载 Amazon S3 存储桶策略，修改文件，然后使用 `put-bucket-policy` 来应用修改后的存储桶策略。要将存储桶策略下载到文件中，您可以运行：

```
aws s3api get-bucket-policy --bucket mybucket --query Policy --output text >
policy.json
```

然后，您可以根据需要修改 `policy.json` 文件。最后，您可以通过运行以下对象，将此修改后的策略应用回 S3 存储桶：

`policy.json` 文件（根据需要）。最后，您可以通过运行以下对象，将此修改后的策略应用回 S3 存储桶：

文件（根据需要）。最后，您可以通过运行以下对象，将此修改后的策略应用回 S3 存储桶：

```
aws s3api put-bucket-policy --bucket mybucket --policy file://policy.json
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetBucketPolicy](#) 中的。

## get-bucket-replication

以下代码示例演示了如何使用 `get-bucket-replication`。

## AWS CLI

以下命令检索名为 `my-bucket` 的存储桶的复制配置：

```
aws s3api get-bucket-replication --bucket my-bucket
```

输出：

```
{
  "ReplicationConfiguration": {
    "Rules": [
      {
        "Status": "Enabled",
        "Prefix": "",
        "Destination": {
          "Bucket": "arn:aws:s3:::my-bucket-backup",
          "StorageClass": "STANDARD"
        },
        "ID": "ZmUwNzE4ZmQ4tMjVhOS00MTlkLOGI4NDkzZTIWJjNTUtYTA1"
      }
    ],
    "Role": "arn:aws:iam::123456789012:role/s3-replication-role"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketReplication](#)中的。

## get-bucket-request-payment

以下代码示例演示了如何使用 get-bucket-request-payment。

AWS CLI

检索存储桶的请求付款配置

以下 get-bucket-request-payment 示例检索指定存储桶的申请方付款配置。

```
aws s3api get-bucket-request-payment \
  --bucket my-bucket
```

输出：

```
{
  "Payer": "BucketOwner"
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketRequestPayment](#)中的。

## get-bucket-tagging

以下代码示例演示了如何使用 `get-bucket-tagging`。

### AWS CLI

以下命令检索名为 `my-bucket` 的存储桶的标记配置：

```
aws s3api get-bucket-tagging --bucket my-bucket
```

输出：

```
{
  "TagSet": [
    {
      "Value": "marketing",
      "Key": "organization"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketTagging](#)中的。

## get-bucket-versioning

以下代码示例演示了如何使用 `get-bucket-versioning`。

### AWS CLI

以下命令检索名为 `my-bucket` 的存储桶的版本控制配置：

```
aws s3api get-bucket-versioning --bucket my-bucket
```

输出：

```
{
```

```
"Status": "Enabled"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketVersioning](#)中的。

## get-bucket-website

以下代码示例演示了如何使用 `get-bucket-website`。

### AWS CLI

以下命令检索名为 `my-bucket` 的存储桶的静态网站配置：

```
aws s3api get-bucket-website --bucket my-bucket
```

输出：

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetBucketWebsite](#)中的。

## get-object-acl

以下代码示例演示了如何使用 `get-object-acl`。

### AWS CLI

以下命令检索名为 `my-bucket` 的存储桶中对象的访问控制列表：

```
aws s3api get-object-acl --bucket my-bucket --key index.html
```

输出：

```
{
```



```

"Owner": {
  "DisplayName": "my-username",
  "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
},
"Grants": [
  {
    "Grantee": {
      "DisplayName": "my-username",
      "ID":
"7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
    },
    "Permission": "FULL_CONTROL"
  },
  {
    "Grantee": {
      "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
    },
    "Permission": "READ"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetObjectAcl](#)中的。

## get-object-attributes

以下代码示例演示了如何使用 `get-object-attributes`。

### AWS CLI

从对象检索元数据而不返回对象本身

以下`get-object-attributes`示例从对象`doc1.rtf`检索元数据。

```

aws s3api get-object-attributes \
  --bucket my-bucket \
  --key doc1.rtf \
  --object-attributes "StorageClass" "ETag" "ObjectSize"

```

输出：

```
{
```

```
"LastModified": "2022-03-15T19:37:31+00:00",
"VersionId": "IuCPjXTDzHNfldAuitVBIKJpF2p1fg4P",
"ETag": "b662d79adeb7c8d787ea7eafb9ef6207",
"StorageClass": "STANDARD",
"ObjectSize": 405
}
```

有关更多信息，请参阅 Amazon S3 API 参考[GetObjectAttributes](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetObjectAttributes](#)中的。

## get-object-legal-hold

以下代码示例演示了如何使用 get-object-legal-hold。

### AWS CLI

检索对象的法定保留状态

以下 get-object-legal-hold 示例检索指定对象的法定保留状态。

```
aws s3api get-object-legal-hold \
  --bucket my-bucket-with-object-lock \
  --key doc1.rtf
```

输出：

```
{
  "LegalHold": {
    "Status": "ON"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetObjectLegalHold](#)中的。

## get-object-lock-configuration

以下代码示例演示了如何使用 get-object-lock-configuration。

### AWS CLI

检索存储桶的对象锁定配置

以下 `get-object-lock-configuration` 示例检索指定存储桶的对象锁定配置。

```
aws s3api get-object-lock-configuration \  
  --bucket my-bucket-with-object-lock
```

输出：

```
{  
  "ObjectLockConfiguration": {  
    "ObjectLockEnabled": "Enabled",  
    "Rule": {  
      "DefaultRetention": {  
        "Mode": "COMPLIANCE",  
        "Days": 50  
      }  
    }  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetObjectLockConfiguration](#)中的。

## get-object-retention

以下代码示例演示了如何使用 `get-object-retention`。

AWS CLI

检索对象的对象保留配置

以下 `get-object-retention` 示例检索指定对象的对象保留配置。

```
aws s3api get-object-retention \  
  --bucket my-bucket-with-object-lock \  
  --key doc1.rtf
```

输出：

```
{  
  "Retention": {  
    "Mode": "GOVERNANCE",
```

```
    "RetainUntilDate": "2025-01-01T00:00:00.000Z"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetObjectRetention](#)中的。

## get-object-tagging

以下代码示例演示了如何使用 `get-object-tagging`。

### AWS CLI

检索附加到对象的标签

以下 `get-object-tagging` 示例从指定的对象中检索指定键的值。

```
aws s3api get-object-tagging \  
  --bucket my-bucket \  
  --key doc1.rtf
```

输出：

```
{  
  "TagSet": [  
    {  
      "Value": "confidential",  
      "Key": "designation"  
    }  
  ]  
}
```

以下 `get-object-tagging` 示例尝试检索没有标签的对象 `doc2.rtf` 的标签集。

```
aws s3api get-object-tagging \  
  --bucket my-bucket \  
  --key doc2.rtf
```

输出：

```
{
```

```
"TagSet": []
}
```

以下 `get-object-tagging` 示例检索具有多个标签的对象 `doc3.rtf` 的标签集。

```
aws s3api get-object-tagging \
  --bucket my-bucket \
  --key doc3.rtf
```

输出：

```
{
  "TagSet": [
    {
      "Value": "confidential",
      "Key": "designation"
    },
    {
      "Value": "finance",
      "Key": "department"
    },
    {
      "Value": "payroll",
      "Key": "team"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetObjectTagging](#)中的。

## get-object-torrent

以下代码示例演示了如何使用 `get-object-torrent`。

### AWS CLI

以下命令为名为 `torrentmy-bucket` 的存储桶中的对象创建 `torrentmy-bucket`：

```
aws s3api get-object-torrent --bucket my-bucket --key large-video-file.mp4 large-
video-file.torrent
```

torrent 文件保存在本地当前文件夹中。请注意，指定输出 filename (large-video-file.torrent) 时没有选项名称，并且必须是命令中的最后一个参数。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetObjectTorrent](#)中的。

## get-object

以下代码示例演示了如何使用 get-object。

### AWS CLI

以下示例使用 get-object 命令从 Amazon S3 下载对象：

```
aws s3api get-object --bucket text-content --key dir/my_images.tar.bz2
my_images.tar.bz2
```

请注意，指定 outfile 参数时没有诸如"--outfile"之类的选项名称。输出文件的名称必须是命令中的最后一个参数。

以下示例演示了如何使用 --range 从对象下载特定字节范围。请注意，字节范围需要以“bytes=”为前缀：

```
aws s3api get-object --bucket text-content --key dir/my_data --range bytes=8888-9999
my_data_range
```

有关检索对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“获取对象”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetObject](#)中的。

## get-public-access-block

以下代码示例演示了如何使用 get-public-access-block。

### AWS CLI

设置或修改存储桶的屏蔽公共访问权限配置

以下 get-public-access-block 示例显示了指定存储桶的屏蔽公共访问权限配置。

```
aws s3api get-public-access-block \
```

```
--bucket my-bucket
```

输出：

```
{
  "PublicAccessBlockConfiguration": {
    "IgnorePublicAcls": true,
    "BlockPublicPolicy": true,
    "BlockPublicAcls": true,
    "RestrictPublicBuckets": true
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPublicAccessBlock](#)中的。

## head-bucket

以下代码示例演示了如何使用 head-bucket。

### AWS CLI

以下命令验证对名为 my-bucket 的存储桶的访问权限：

```
aws s3api head-bucket --bucket my-bucket
```

如果存储桶存在并且您可以访问它，则不返回任何输出。否则，会显示错误消息。例如：

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[HeadBucket](#)中的。

## head-object

以下代码示例演示了如何使用 head-object。

### AWS CLI

以下命令检索名为 my-bucket 的存储桶中对象的元数据：

```
aws s3api head-object --bucket my-bucket --key index.html
```

输出：

```
{
  "AcceptRanges": "bytes",
  "ContentType": "text/html",
  "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
  "ContentLength": 77,
  "VersionId": "null",
  "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
  "Metadata": {}
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[HeadObject](#)中的。

## list-bucket-analytics-configurations

以下代码示例演示了如何使用 list-bucket-analytics-configurations。

### AWS CLI

检索存储桶的分析配置列表

下面的 list-bucket-analytics-configurations 检索指定存储桶的分析配置列表。

```
aws s3api list-bucket-analytics-configurations \
  --bucket my-bucket
```

输出：

```
{
  "AnalyticsConfigurationList": [
    {
      "StorageClassAnalysis": {},
      "Id": "1"
    }
  ],
  "IsTruncated": false
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBucketAnalyticsConfigurations](#)中的。



## list-bucket-intelligent-tiering-configurations

以下代码示例演示了如何使用 list-bucket-intelligent-tiering-configurations。

### AWS CLI

检索存储桶上的所有 S3 智能分层配置

以下list-bucket-intelligent-tiering-configurations示例检索存储桶上的所有 S3 智能分层配置。

```
aws s3api list-bucket-intelligent-tiering-configurations \
  --bucket DOC-EXAMPLE-BUCKET
```

输出：

```
{
  "IsTruncated": false,
  "IntelligentTieringConfigurationList": [
    {
      "Id": "ExampleConfig",
      "Filter": {
        "Prefix": "images"
      },
      "Status": "Enabled",
      "Tierings": [
        {
          "Days": 90,
          "AccessTier": "ARCHIVE_ACCESS"
        },
        {
          "Days": 180,
          "AccessTier": "DEEP_ARCHIVE_ACCESS"
        }
      ]
    },
    {
      "Id": "ExampleConfig2",
      "Status": "Disabled",
      "Tierings": [
        {
          "Days": 730,
          "AccessTier": "ARCHIVE_ACCESS"
        }
      ]
    }
  ]
}
```

```
    }
  ]
},
{
  "Id": "ExampleConfig3",
  "Filter": {
    "Tag": {
      "Key": "documents",
      "Value": "taxes"
    }
  },
  "Status": "Enabled",
  "Tierings": [
    {
      "Days": 90,
      "AccessTier": "ARCHIVE_ACCESS"
    },
    {
      "Days": 365,
      "AccessTier": "DEEP_ARCHIVE_ACCESS"
    }
  ]
}
]
```

有关更多信息，请参阅 [Amazon S3 用户指南中的使用 S3 智能分层](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListBucketIntelligentTieringConfigurations](#) 中的。

## list-bucket-inventory-configurations

以下代码示例演示了如何使用 list-bucket-inventory-configurations。

### AWS CLI

检索存储桶的清单配置列表

以下 list-bucket-inventory-configurations 示例列出了指定存储桶的清单配置。

```
aws s3api list-bucket-inventory-configurations \
  --bucket my-bucket
```

输出：

```
{
  "InventoryConfigurationList": [
    {
      "IsEnabled": true,
      "Destination": {
        "S3BucketDestination": {
          "Format": "ORC",
          "Bucket": "arn:aws:s3::my-bucket",
          "AccountId": "123456789012"
        }
      },
      "IncludedObjectVersions": "Current",
      "Id": "1",
      "Schedule": {
        "Frequency": "Weekly"
      }
    },
    {
      "IsEnabled": true,
      "Destination": {
        "S3BucketDestination": {
          "Format": "CSV",
          "Bucket": "arn:aws:s3::my-bucket",
          "AccountId": "123456789012"
        }
      },
      "IncludedObjectVersions": "Current",
      "Id": "2",
      "Schedule": {
        "Frequency": "Daily"
      }
    }
  ],
  "IsTruncated": false
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBucketInventoryConfigurations](#)中的。

## list-bucket-metrics-configurations

以下代码示例演示了如何使用 list-bucket-metrics-configurations。

## AWS CLI

### 检索存储桶的指标配置列表

以下`list-bucket-metrics-configurations`示例检索指定存储桶的指标配置列表。

```
aws s3api list-bucket-metrics-configurations \  
  --bucket my-bucket
```

输出：

```
{  
  "IsTruncated": false,  
  "MetricsConfigurationList": [  
    {  
      "Filter": {  
        "Prefix": "logs"  
      },  
      "Id": "123"  
    },  
    {  
      "Filter": {  
        "Prefix": "tmp"  
      },  
      "Id": "234"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBucketMetricsConfigurations](#)中的。

## list-buckets

以下代码示例演示了如何使用 `list-buckets`。

### AWS CLI

以下命令使用 `list-buckets` 命令显示所有 Amazon S3 存储桶的名称（跨所有区域）：

```
aws s3api list-buckets --query "Buckets[].Name"
```

查询选项会筛选 `list-buckets` 的输出，使其范围缩小到仅限存储桶名称。

有关存储桶的更多信息，请参阅《Amazon S3 开发人员指南》中的“使用 Amazon S3 存储桶”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListBuckets](#)中的。

## list-multipart-uploads

以下代码示例演示了如何使用 `list-multipart-uploads`。

### AWS CLI

以下命令列出了名为 `my-bucket` 的存储桶的所有活动分段上传：

```
aws s3api list-multipart-uploads --bucket my-bucket
```

输出：

```
{
  "Uploads": [
    {
      "Initiator": {
        "DisplayName": "username",
        "ID": "arn:aws:iam::0123456789012:user/username"
      },
      "Initiated": "2015-06-02T18:01:30.000Z",
      "UploadId":
      "dfRtDYU0WWCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3URC
      "StorageClass": "STANDARD",
      "Key": "multipart/01",
      "Owner": {
        "DisplayName": "aws-account-name",
        "ID":
        "100719349fc3b6dcd7c820a124bf7aec408092c3d7b51b38494939801fc248b"
      }
    }
  ],
  "CommonPrefixes": []
}
```

正在进行的分段上传会产生 Amazon S3 存储费用。完成或中止活动分段上传，可将其从您的账户中移除。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListMultipartUploads](#)中的。

## list-object-versions

以下代码示例演示了如何使用 list-object-versions。

### AWS CLI

以下命令检索名为 my-bucket 的存储桶中对象的版本信息：

```
aws s3api list-object-versions --bucket my-bucket --prefix index.html
```

输出：

```
{
  "DeleteMarkers": [
    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": true,
      "VersionId": "B2VsEK5saUNNHKc0AJj7hIE86RozToyq",
      "Key": "index.html",
      "LastModified": "2015-11-10T00:57:03.000Z"
    },
    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
      "Key": "index.html",
      "LastModified": "2015-11-09T23:32:20.000Z"
    }
  ],
  "Versions": [
    {
      "LastModified": "2015-11-10T00:20:11.000Z",
```

```

    "VersionId": "Rb_l2T8UHDkFEwCgJjhlgPOZC0qJ.vpD",
    "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
    "StorageClass": "STANDARD",
    "Key": "index.html",
    "Owner": {
      "DisplayName": "my-username",
      "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
    },
    "IsLatest": false,
    "Size": 38
  },
  {
    "LastModified": "2015-11-09T23:26:41.000Z",
    "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAff",
    "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
    "StorageClass": "STANDARD",
    "Key": "index.html",
    "Owner": {
      "DisplayName": "my-username",
      "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
    },
    "IsLatest": false,
    "Size": 38
  },
  {
    "LastModified": "2015-11-09T22:50:50.000Z",
    "VersionId": "null",
    "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
    "StorageClass": "STANDARD",
    "Key": "index.html",
    "Owner": {
      "DisplayName": "my-username",
      "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
    },
    "IsLatest": false,
    "Size": 533823
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListObjectVersions](#)中的。

## list-objects-v2

以下代码示例演示了如何使用 list-objects-v2。

### AWS CLI

获取存储桶中对象的列表

以下 list-objects-v2 示例列出了指定存储桶中的对象。

```
aws s3api list-objects-v2 \  
  --bucket my-bucket
```

输出：

```
{  
  "Contents": [  
    {  
      "LastModified": "2019-11-05T23:11:50.000Z",  
      "ETag": "\"621503c373607d548b37cff8778d992c\"",  
      "StorageClass": "STANDARD",  
      "Key": "doc1.rtf",  
      "Size": 391  
    },  
    {  
      "LastModified": "2019-11-05T23:11:50.000Z",  
      "ETag": "\"a2cecc36ab7c7fe3a71a273b9d45b1b5\"",  
      "StorageClass": "STANDARD",  
      "Key": "doc2.rtf",  
      "Size": 373  
    },  
    {  
      "LastModified": "2019-11-05T23:11:50.000Z",  
      "ETag": "\"08210852f65a2e9cb999972539a64d68\"",  
      "StorageClass": "STANDARD",  
      "Key": "doc3.rtf",  
      "Size": 399  
    },  
    {  
      "LastModified": "2019-11-05T23:11:50.000Z",  
      "ETag": "\"d1852dd683f404306569471af106988e\"",  
      "StorageClass": "STANDARD",  
      "Key": "doc4.rtf",  
      "Size": 6225  
    }  
  ]  
}
```



```
    }
  ]
}
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [ListObjectsV2](#)。

## list-objects

以下代码示例演示了如何使用 `list-objects`。

### AWS CLI

以下示例使用 `list-objects` 命令显示指定存储桶中所有对象的名称：

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

该示例使用 `--query` 参数筛选 `list-objects` 的输出，使其范围缩小到每个对象的键值和大小。有关对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“使用 Amazon S3 对象”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListObjects](#) 中的。

## list-parts

以下代码示例演示了如何使用 `list-parts`。

### AWS CLI

以下命令列出了为分段上传而上传的所有分段，密钥 `multipart/01` 位于存储桶 `my-bucket` 中：

```
aws s3api list-parts --bucket my-bucket --key 'multipart/01' --upload-id dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3URCS
```

输出：

```
{
  "Owner": {
    "DisplayName": "aws-account-name",
    "ID": "100719349fc3b6dcd7c820a124bf7aecd408092c3d7b51b38494939801fc248b"
  },
  "Initiator": {
```

```

    "DisplayName": "username",
    "ID": "arn:aws:iam::0123456789012:user/username"
  },
  "Parts": [
    {
      "LastModified": "2015-06-02T18:07:35.000Z",
      "PartNumber": 1,
      "ETag": "\"e868e0f4719e394144ef36531ee6824c\"",
      "Size": 5242880
    },
    {
      "LastModified": "2015-06-02T18:07:42.000Z",
      "PartNumber": 2,
      "ETag": "\"6bb2b12753d66fe86da4998aa33fffb0\"",
      "Size": 5242880
    },
    {
      "LastModified": "2015-06-02T18:07:47.000Z",
      "PartNumber": 3,
      "ETag": "\"d0a0112e841abec9c9ec83406f0159c8\"",
      "Size": 5242880
    }
  ],
  "StorageClass": "STANDARD"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListParts](#)中的。

## ls

以下代码示例演示了如何使用 `ls`。

### AWS CLI

#### 示例 1：列出所有用户拥有的存储桶

以下`ls`命令列出了用户拥有的所有存储桶。在此示例中，用户拥有存储桶`mybucket`和`mybucket2`。时间戳是存储桶的创建日期，以计算机的时区显示。更改存储桶（例如编辑存储桶策略）时，此日期可能会更改。请注意`s3://`，如果用于 `path` 参数 `<S3Uri>`，它也会列出所有存储桶。

```
aws s3 ls
```

输出：

```
2013-07-11 17:08:50 mybucket
2013-07-24 14:55:44 mybucket2
```

示例 2：列出存储桶中的所有前缀和对象

以下`ls`命令列出指定存储桶和前缀下的对象和常用前缀。在此示例中，用户拥有`mybucket`包含`test.txt`和对象的存储桶`somePrefix/test.txt`。`LastWriteTime`和`Length`是任意的。请注意，由于该`ls`命令与本地文件系统没有交互，因此不需要使用 `s3://` URI 方案来解决歧义，因此可以省略 URI 方案。

```
aws s3 ls s3://mybucket
```

输出：

```
                PRE somePrefix/
2013-07-25 17:06:27      88 test.txt
```

示例 3：列出特定存储桶和前缀中的所有前缀和对象

以下`ls`命令列出指定存储桶和前缀下的对象和常用前缀。但是，指定的存储桶和前缀下没有对象，也没有常用前缀。

```
aws s3 ls s3://mybucket/noExistPrefix
```

输出：

```
None
```

示例 4：以递归方式列出存储桶中的所有前缀和对象

以下`ls`命令将递归列出存储桶中的对象。存储桶`PRE dirname/`中的所有内容将按顺序列出，而不是显示在输出中。

```
aws s3 ls s3://mybucket \
  --recursive
```

输出：

```

2013-09-02 21:37:53      10 a.txt
2013-09-02 21:37:53 2863288 foo.zip
2013-09-02 21:32:57      23 foo/bar/.baz/a
2013-09-02 21:32:58      41 foo/bar/.baz/b
2013-09-02 21:32:57     281 foo/bar/.baz/c
2013-09-02 21:32:57      73 foo/bar/.baz/d
2013-09-02 21:32:57     452 foo/bar/.baz/e
2013-09-02 21:32:57     896 foo/bar/.baz/hooks/bar
2013-09-02 21:32:57     189 foo/bar/.baz/hooks/foo
2013-09-02 21:32:57     398 z.txt

```

### 示例 5：汇总存储桶中的所有前缀和对象

以下`ls`命令使用`--human-readable`和`--summary`选项演示了相同的命令。`--human-readable`以字节/mib/kib/kib/Gib/tib/pib/EiB 为单位显示文件大小。`--summary`在结果列表的末尾显示对象的总数和总大小：

```

aws s3 ls s3://mybucket \
  --recursive \
  --human-readable \
  --summarize

```

### 输出：

```

2013-09-02 21:37:53   10 Bytes a.txt
2013-09-02 21:37:53  2.9 MiB foo.zip
2013-09-02 21:32:57   23 Bytes foo/bar/.baz/a
2013-09-02 21:32:58   41 Bytes foo/bar/.baz/b
2013-09-02 21:32:57  281 Bytes foo/bar/.baz/c
2013-09-02 21:32:57   73 Bytes foo/bar/.baz/d
2013-09-02 21:32:57  452 Bytes foo/bar/.baz/e
2013-09-02 21:32:57  896 Bytes foo/bar/.baz/hooks/bar
2013-09-02 21:32:57  189 Bytes foo/bar/.baz/hooks/foo
2013-09-02 21:32:57  398 Bytes z.txt

Total Objects: 10
  Total Size: 2.9 MiB

```

### 示例 6：从 S3 接入点列出

以下`ls`命令列出来自接入点 (myaccesspoint) 的对象：

```
aws s3 ls s3://arn:aws:s3:us-west-2:123456789012:accesspoint/myaccesspoint/
```

输出：

```
2013-07-25 17:06:27          PRE somePrefix/
                        88 test.txt
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [Ls](#)。

## mb

以下代码示例演示了如何使用 mb。

### AWS CLI

#### 示例 1：创建存储桶

以下mb命令创建存储桶。在此示例中，用户创建了存储桶mybucket。存储桶是在用户配置文件中指定的区域创建的：

```
aws s3 mb s3://mybucket
```

输出：

```
make_bucket: s3://mybucket
```

#### 示例 2：在指定区域创建存储桶

以下mb命令在--region参数指定的区域中创建存储桶。在此示例中，用户mybucket在以下区域创建存储桶us-west-1：

```
aws s3 mb s3://mybucket \  
  --region us-west-1
```

输出：

```
make_bucket: s3://mybucket
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [Mb](#)。

## mv

以下代码示例演示了如何使用 mv。

### AWS CLI

示例 1：将本地文件移动到指定的存储桶

以下mv命令将单个文件移动到指定的存储桶和密钥。

```
aws s3 mv test.txt s3://mybucket/test2.txt
```

输出：

```
move: test.txt to s3://mybucket/test2.txt
```

示例 2：将对象移至指定的存储桶和密钥

以下mv命令将单个 s3 对象移动到指定的存储桶和密钥。

```
aws s3 mv s3://mybucket/test.txt s3://mybucket/test2.txt
```

输出：

```
move: s3://mybucket/test.txt to s3://mybucket/test2.txt
```

示例 3：将 S3 对象移动到本地目录

以下mv命令将单个对象移动到本地的指定文件中。

```
aws s3 mv s3://mybucket/test.txt test2.txt
```

输出：

```
move: s3://mybucket/test.txt to test2.txt
```

示例 4：将具有原始名称的对象移动到指定的存储桶

以下mv命令将单个对象移动到指定的存储桶，同时保留其原始名称：

```
aws s3 mv s3://mybucket/test.txt s3://mybucket2/
```

输出：

```
move: s3://mybucket/test.txt to s3://mybucket2/test.txt
```

示例 5：将存储桶中的所有对象和前缀移至本地目录

当与参数一起传递时`--recursive`，以下`mv`命令会递归地将指定前缀下的所有对象和存储桶移动到指定目录。在此示例中`mybucket`，存储桶包含对象`test1.txt`和`test2.txt`。

```
aws s3 mv s3://mybucket . \  
--recursive
```

输出：

```
move: s3://mybucket/test1.txt to test1.txt  
move: s3://mybucket/test2.txt to test2.txt
```

示例 6：将存储桶中的所有对象和前缀移到本地目录，“.jpg”文件除外

当与参数一起传递时`--recursive`，以下`mv`命令将指定目录下的所有文件递归移动到指定的存储桶和前缀，同时使用`--exclude`参数排除某些文件。在此示例中`myDir`，目录包含文件`test1.txt`和`test2.jpg`。

```
aws s3 mv myDir s3://mybucket/ \  
--recursive \  
--exclude "*.jpg"
```

输出：

```
move: myDir/test1.txt to s3://mybucket2/test1.txt
```

示例 7：将存储桶中的所有对象和前缀移到本地目录（指定前缀除外）

当与参数一起传递时`--recursive`，以下`mv`命令会递归地将指定存储桶下的所有对象移动到另一个存储桶，同时使用`--exclude`参数排除某些对象。在此示例中`mybucket`，存储桶包含对象`test1.txt`和`another/test1.txt`。

```
aws s3 mv s3://mybucket/ s3://mybucket2/ \  
--recursive \  
--exclude another/
```

```
--exclude "mybucket/another/*"
```

输出：

```
move: s3://mybucket/test1.txt to s3://mybucket2/test1.txt
```

示例 8：将对象移至指定存储桶并设置 ACL

以下mv命令将单个对象移动到指定的存储桶和密钥，同时将 ACL 设置为public-read-write。

```
aws s3 mv s3://mybucket/test.txt s3://mybucket/test2.txt \  
--acl public-read-write
```

输出：

```
move: s3://mybucket/test.txt to s3://mybucket/test2.txt
```

示例 9：将本地文件移至指定存储桶并授予权限

以下mv命令说明了如何使用该--grants选项向所有用户授予读取权限，并向通过其电子邮件地址标识的特定用户授予完全控制权。

```
aws s3 mv file.txt s3://mybucket/ \  
--grants read=uri=http://acs.amazonaws.com/groups/global/AllUsers  
full=emailaddress=user@example.com
```

输出：

```
move: file.txt to s3://mybucket/file.txt
```

示例 10：将文件移动到 S3 接入点

以下mv命令将名为的单个文件移动mydoc.txt到以名为的密钥命名的myaccesspoint接入点mykey。

```
aws s3 mv mydoc.txt s3://arn:aws:s3:us-west-2:123456789012:accesspoint/  
myaccesspoint/mykey
```

输出：



```
move: mydoc.txt to s3://arn:aws:s3:us-west-2:123456789012:accesspoint/myaccesspoint/
mykey
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [Mv](#)。

## presign

以下代码示例演示了如何使用 presign。

### AWS CLI

示例 1：创建默认生命周期为一小时的预签名 URL，用于链接到 S3 存储桶中的对象

以下presign命令为指定的存储桶和密钥生成一个预签名 URL，有效期为一小时。

```
aws s3 presign s3://DOC-EXAMPLE-BUCKET/test2.txt
```

输出：

```
https://DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com/key?X-Amz-Algorithm=AWS4-
HMAC-SHA256&X-Amz-Credential=AKIAEXAMPLE123456789%2F20210621%2Fus-
west-2%2Fs3%2Faws4_request&X-Amz-Date=20210621T041609Z&X-Amz-Expires=3600&X-Amz-
SignedHeaders=host&X-Amz-
Signature=EXAMBLE1234494d5fba3fed607f98018e1dfc62e2529ae96d844123456
```

示例 2：创建具有自定义生命周期的预签名 URL，该网址链接到 S3 存储桶中的对象

以下presign命令为指定的存储桶和密钥生成一个预签名 URL，有效期为一周。

```
aws s3 presign s3://DOC-EXAMPLE-BUCKET/test2.txt \
--expires-in 604800
```

输出：

```
https://DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com/key?X-Amz-Algorithm=AWS4-
HMAC-SHA256&X-Amz-Credential=AKIAEXAMPLE123456789%2F20210621%2Fus-
west-2%2Fs3%2Faws4_request&X-Amz-Date=20210621T041609Z&X-Amz-Expires=604800&X-Amz-
SignedHeaders=host&X-Amz-
Signature=EXAMBLE1234494d5fba3fed607f98018e1dfc62e2529ae96d844123456
```

有关更多信息，请参阅《S3 开发人员指南》指南中的[与其他人共享对象](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [Presign](#)。

## put-bucket-accelerate-configuration

以下代码示例演示了如何使用 `put-bucket-accelerate-configuration`。

### AWS CLI

#### 设置存储桶的加速配置

以下 `put-bucket-accelerate-configuration` 示例启用指定存储桶的加速配置。

```
aws s3api put-bucket-accelerate-configuration \  
  --bucket my-bucket \  
  --accelerate-configuration Status=Enabled
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketAccelerateConfiguration](#)中的。

## put-bucket-acl

以下代码示例演示了如何使用 `put-bucket-acl`。

### AWS CLI

此示例 `full control` 向两个 AWS 用户 ( `user1@example.com` 和 `user2@example.com` ) 授予权限，并向所有人 `read` 授予权限：

```
aws s3api put-bucket-acl --bucket MyBucket --grant-full-control  
  emailaddress=user1@example.com,emailaddress=user2@example.com --grant-read  
  uri=http://acs.amazonaws.com/groups/global/AllUsers
```

有关自定义 ACL ( `s3api ACL 命令` ( 如 `put-bucket-acl` ) 使用相同的速记参数表示法 ) 的详细信息，请参阅 <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html>。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketAcl](#)中的。

## put-bucket-analytics-configuration

以下代码示例演示了如何使用 `put-bucket-analytics-configuration`。

## AWS CLI

### 为存储桶设置分析配置

以下put-bucket-analytics-configuration示例为指定存储桶配置分析。

```
aws s3api put-bucket-analytics-configuration \  
  --bucket my-bucket --id 1 \  
  --analytics-configuration '{"Id": "1","StorageClassAnalysis": {}}'
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketAnalyticsConfiguration](#)中的。

## put-bucket-cors

以下代码示例演示了如何使用 put-bucket-cors。

## AWS CLI

以下示例启用来自 www.example.com 的 PUT、POST 和 DELETE 请求，并启用来自任何域的 GET 请求：

```
aws s3api put-bucket-cors --bucket MyBucket --cors-configuration file://cors.json  
  
cors.json:  
{  
  "CORSRules": [  
    {  
      "AllowedOrigins": ["http://www.example.com"],  
      "AllowedHeaders": ["*"],  
      "AllowedMethods": ["PUT", "POST", "DELETE"],  
      "MaxAgeSeconds": 3000,  
      "ExposeHeaders": ["x-amz-server-side-encryption"]  
    },  
    {  
      "AllowedOrigins": ["*"],  
      "AllowedHeaders": ["Authorization"],  
      "AllowedMethods": ["GET"],  
      "MaxAgeSeconds": 3000  
    }  
  ]  
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketCors](#)中的。

## put-bucket-encryption

以下代码示例演示了如何使用 put-bucket-encryption。

### AWS CLI

#### 配置存储桶的服务器端加密

以下 put-bucket-encryption 示例将 AES256 加密设置为指定存储桶的默认值。

```
aws s3api put-bucket-encryption \  
  --bucket my-bucket \  
  --server-side-encryption-configuration '{"Rules":  
  [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketEncryption](#)中的。

## put-bucket-intelligent-tiering-configuration

以下代码示例演示了如何使用 put-bucket-intelligent-tiering-configuration。

### AWS CLI

#### 更新存储桶上的 S3 智能分层配置

以下put-bucket-intelligent-tiering-configuration示例更新了存储桶上名 ExampleConfig为的 S3 智能分层配置。该配置将在 90 天后将未在前缀图像下访问的对象转换为存档访问权限和 180 天后深度存档访问权限。

```
aws s3api put-bucket-intelligent-tiering-configuration \  
  --bucket DOC-EXAMPLE-BUCKET \  
  --id "ExampleConfig" \  
  --intelligent-tiering-configuration file://intelligent-tiering-  
  configuration.json
```

intelligent-tiering-configuration.json 的内容：

```
{
  "Id": "ExampleConfig",
  "Status": "Enabled",
  "Filter": {
    "Prefix": "images"
  },
  "Tierings": [
    {
      "Days": 90,
      "AccessTier": "ARCHIVE_ACCESS"
    },
    {
      "Days": 180,
      "AccessTier": "DEEP_ARCHIVE_ACCESS"
    }
  ]
}
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon S3 用户指南中的[为现有存储桶设置对象所有权](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[PutBucketIntelligentTieringConfiguration](#)中的。

## put-bucket-inventory-configuration

以下代码示例演示了如何使用 put-bucket-inventory-configuration。

### AWS CLI

#### 示例 1：为存储桶设置清单配置

以下 put-bucket-inventory-configuration 示例为存储桶设置了每周 ORC 格式的库存报告。my-bucket

```
aws s3api put-bucket-inventory-configuration \
  --bucket my-bucket \
  --id 1 \
  --inventory-configuration '{"Destination": { "S3BucketDestination":
{ "AccountId": "123456789012", "Bucket": "arn:aws:s3:::my-bucket", "Format":
"ORC" }}, "IsEnabled": true, "Id": "1", "IncludedObjectVersions": "Current",
"Schedule": { "Frequency": "Weekly" } }'
```

此命令不生成任何输出。

示例 2：为存储桶设置清单配置

以下 `put-bucket-inventory-configuration` 示例为存储桶设置 CSV 格式的每日库存报告。 `my-bucket`

```
aws s3api put-bucket-inventory-configuration \  
  --bucket my-bucket \  
  --id 2 \  
  --inventory-configuration '{"Destination": { "S3BucketDestination":  
  { "AccountId": "123456789012", "Bucket": "arn:aws:s3:::my-bucket", "Format":  
  "CSV" }}, "IsEnabled": true, "Id": "2", "IncludedObjectVersions": "Current",  
  "Schedule": { "Frequency": "Daily" } }'
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutBucketInventoryConfiguration](#) 中的。

## put-bucket-lifecycle-configuration

以下代码示例演示了如何使用 `put-bucket-lifecycle-configuration`。

### AWS CLI

以下命令将生命周期配置应用于名为 `my-bucket` 的存储桶：

```
aws s3api put-bucket-lifecycle-configuration --bucket my-bucket --lifecycle-  
configuration file://lifecycle.json
```

文件 `lifecycle.json` 是当前文件夹中指定两个规则的 JSON 文档：

```
{  
  "Rules": [  
    {  
      "ID": "Move rotated logs to Glacier",  
      "Prefix": "rotated/",  
      "Status": "Enabled",  
      "Transitions": [  
        {  
          "Date": "2015-11-10T00:00:00.000Z",
```

```

        "StorageClass": "GLACIER"
      }
    ]
  },
  {
    "Status": "Enabled",
    "Prefix": "",
    "NoncurrentVersionTransitions": [
      {
        "NoncurrentDays": 2,
        "StorageClass": "GLACIER"
      }
    ],
    "ID": "Move old versions to Glacier"
  }
]
}

```

第一条规则在指定日期将带有前缀 `rotated` 的文件移动到 Glacier。第二条规则在旧对象版本不再是最新版本时将其移至 Glacier。有关可接受时间戳格式的信息，请参阅《AWS CLI 用户指南》中的“指定参数值”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutBucketLifecycleConfiguration](#) 中的。

## put-bucket-lifecycle

以下代码示例演示了如何使用 `put-bucket-lifecycle`。

### AWS CLI

以下命令将生命周期配置应用于存储桶 `my-bucket`：

```
aws s3api put-bucket-lifecycle --bucket my-bucket --lifecycle-configuration file://lifecycle.json
```

文件 `lifecycle.json` 是当前文件夹中指定两个规则的 JSON 文档：

```

{
  "Rules": [
    {
      "ID": "Move to Glacier after sixty days (objects in logs/2015/)",
      "Prefix": "logs/2015/",

```

```

    "Status": "Enabled",
    "Transition": {
      "Days": 60,
      "StorageClass": "GLACIER"
    }
  },
  {
    "Expiration": {
      "Date": "2016-01-01T00:00:00.000Z"
    },
    "ID": "Delete 2014 logs in 2016.",
    "Prefix": "logs/2014/",
    "Status": "Enabled"
  }
]
}

```

第一条规则是在六十天后将文件移至 Amazon Glacier。第二条规则在指定日期从 Amazon S3 中删除文件。有关可接受时间戳格式的信息，请参阅《AWS CLI 用户指南》中的“指定参数值”。

上面示例中的每条规则都指定了其适用的策略 ( Transition或Expiration ) 和文件前缀 ( 文件夹名称 )。您还可以通过指定空白前缀来创建适用于整个存储桶的规则：

```

{
  "Rules": [
    {
      "ID": "Move to Glacier after sixty days (all objects in bucket)",
      "Prefix": "",
      "Status": "Enabled",
      "Transition": {
        "Days": 60,
        "StorageClass": "GLACIER"
      }
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketLifecycle](#)中的。

## put-bucket-logging

以下代码示例演示了如何使用 put-bucket-logging。



## AWS CLI

### 示例 1：设置存储桶策略日志记录

以下put-bucket-logging示例为设置了日志策略MyBucket。首先，使用 put-bucket-policy 命令在存储桶策略中向日志记录服务主体授予权限。

```
aws s3api put-bucket-policy \  
  --bucket MyBucket \  
  --policy file://policy.json
```

policy.json 的内容：

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "S3ServerAccessLogsPolicy",  
      "Effect": "Allow",  
      "Principal": {"Service": "logging.s3.amazonaws.com"},  
      "Action": "s3:PutObject",  
      "Resource": "arn:aws:s3:::MyBucket/Logs/*",  
      "Condition": {  
        "ArnLike": {"aws:SourceARN": "arn:aws:s3:::SOURCE-BUCKET-NAME"},  
        "StringEquals": {"aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"}  
      }  
    }  
  ]  
}
```

要应用日志记录策略，请使用 put-bucket-logging。

```
aws s3api put-bucket-logging \  
  --bucket MyBucket \  
  --bucket-logging-status file://logging.json
```

logging.json 的内容：

```
{  
  "LoggingEnabled": {  
    "TargetBucket": "MyBucket",
```

```
    "TargetPrefix": "Logs/"
  }
}
```

向日志记录服务主体授予 `s3:PutObject` 权限需要使用 `put-bucket-policy` 命令。

有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 [Amazon S3 服务器访问日志记录](#)。

示例 2：设置仅向单个用户授予日志访问权限的存储桶策略

以下 `put-bucket-logging` 示例为设置了日志策略 `MyBucket`。AWS 用户 `bob@example.com` 将完全控制日志文件，其他人没有任何访问权限。首先，使用 `put-bucket-acl` 授予 S3 权限。

```
aws s3api put-bucket-acl \
  --bucket MyBucket \
  --grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery \
  --grant-read-acp URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

然后，使用 `put-bucket-logging` 应用日志记录策略。

```
aws s3api put-bucket-logging \
  --bucket MyBucket \
  --bucket-logging-status file://logging.json
```

`logging.json` 的内容：

```
{
  "LoggingEnabled": {
    "TargetBucket": "MyBucket",
    "TargetPrefix": "MyBucketLogs/",
    "TargetGrants": [
      {
        "Grantee": {
          "Type": "AmazonCustomerByEmail",
          "EmailAddress": "bob@example.com"
        },
        "Permission": "FULL_CONTROL"
      }
    ]
  }
}
```

```
}
```

必须使用 `put-bucket-acl` 命令向 S3 的日志传输系统授予必要的权限 ( `write-acp` 和 `read-acp` 权限 )。

有关更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的 [Amazon S3 服务器访问日志记录](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketLogging](#)中的。

## **put-bucket-metrics-configuration**

以下代码示例演示了如何使用 `put-bucket-metrics-configuration`。

### AWS CLI

为存储桶设置指标配置

以下`put-bucket-metrics-configuration`示例为指定存储桶设置 ID 为 123 的指标配置。

```
aws s3api put-bucket-metrics-configuration \  
  --bucket my-bucket \  
  --id 123 \  
  --metrics-configuration '{"Id": "123", "Filter": {"Prefix": "logs"}}'
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketMetricsConfiguration](#)中的。

## **put-bucket-notification-configuration**

以下代码示例演示了如何使用 `put-bucket-notification-configuration`。

### AWS CLI

启用存储桶的指定通知

以下 `put-bucket-notification-configuration` 示例将通知配置应用于名为 `my-bucket` 的存储桶。文件 `notification.json` 是当前文件夹中的一个 JSON 文件，用于指定 SNS 主题和要监控的事件类型。

```
aws s3api put-bucket-notification-configuration \  
  --bucket my-bucket \  
  --notification-configuration notification.json
```

```
--bucket my-bucket \  
--notification-configuration file://notification.json
```

notification.json 的内容：

```
{  
  "TopicConfigurations": [  
    {  
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic",  
      "Events": [  
        "s3:ObjectCreated:*"  
      ]  
    }  
  ]  
}
```

SNS 主题必须附加一个 IAM 策略，以允许 Amazon S3 向其发布。

```
{  
  "Version": "2008-10-17",  
  "Id": "example-ID",  
  "Statement": [  
    {  
      "Sid": "example-statement-ID",  
      "Effect": "Allow",  
      "Principal": {  
        "Service": "s3.amazonaws.com"  
      },  
      "Action": [  
        "SNS:Publish"  
      ],  
      "Resource": "arn:aws:sns:us-west-2:123456789012::s3-notification-topic",  
      "Condition": {  
        "ArnLike": {  
          "aws:SourceArn": "arn:aws:s3:*:*:my-bucket"  
        }  
      }  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketNotificationConfiguration](#)中的。

## put-bucket-notification

以下代码示例演示了如何使用 `put-bucket-notification`。

### AWS CLI

将通知配置应用于名为 `my-bucket` 的存储桶：

```
aws s3api put-bucket-notification --bucket my-bucket --notification-configuration
file://notification.json
```

文件 `notification.json` 是当前文件夹中的一个 JSON 文件，用于指定 SNS 主题和要监控的事件类型：

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

SNS 主题必须附加一个 IAM 策略，以允许 Amazon S3 向其发布：

```
{
  "Version": "2008-10-17",
  "Id": "example-ID",
  "Statement": [
    {
      "Sid": "example-statement-ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": [
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:us-west-2:123456789012:my-bucket",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:*:*:my-bucket"
        }
      }
    }
  ]
}
```

```
}  
]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketNotification](#)中的。

## put-bucket-ownership-controls

以下代码示例演示了如何使用 `put-bucket-ownership-controls`。

### AWS CLI

更新存储桶的存储桶所有权设置

以下`put-bucket-ownership-controls`示例更新了存储桶的存储桶所有权设置。

```
aws s3api put-bucket-ownership-controls \  
  --bucket DOC-EXAMPLE-BUCKET \  
  --ownership-controls="Rules=[{ObjectOwnership=BucketOwnerEnforced}]"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon S3 用户指南中的[为现有存储桶设置对象所有权](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketOwnershipControls](#)中的。

## put-bucket-policy

以下代码示例演示了如何使用 `put-bucket-policy`。

### AWS CLI

此示例允许所有用户检索中的任何对象，MyBucket但中的对象除外MySecretFolder。它还向 AWS 账户的根用户授予`put`和`delete`权限1234-5678-9012：

```
aws s3api put-bucket-policy --bucket MyBucket --policy file://policy.json  
  
policy.json:  
{  
  "Statement": [  
    {
```

```
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::MyBucket/*"
  },
  {
    "Effect": "Deny",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::MyBucket/MySecretFolder/*"
  },
  {
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::123456789012:root"
    },
    "Action": [
      "s3:DeleteObject",
      "s3:PutObject"
    ],
    "Resource": "arn:aws:s3:::MyBucket/*"
  }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketPolicy](#)中的。

## put-bucket-replication

以下代码示例演示了如何使用 put-bucket-replication。

### AWS CLI

为 S3 存储桶配置复制

以下 put-bucket-replication 示例将复制配置应用于指定的 S3 存储桶。

```
aws s3api put-bucket-replication \  
  --bucket AWSDOC-EXAMPLE-BUCKET1 \  
  --replication-configuration file://replication.json
```

replication.json 的内容：

```
{
  "Role": "arn:aws:iam::123456789012:role/s3-replication-role",
  "Rules": [
    {
      "Status": "Enabled",
      "Priority": 1,
      "DeleteMarkerReplication": { "Status": "Disabled" },
      "Filter" : { "Prefix": ""},
      "Destination": {
        "Bucket": "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET2"
      }
    }
  ]
}
```

目标存储桶必须已启用版本控制。指定的角色必须具有写入目标存储桶的权限，并且必须建立允许 Amazon S3 代入角色的信任关系。

示例角色权限策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetReplicationConfiguration",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET1"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET1/*"
      ]
    }
  ]
}
```



```

    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete",
        "s3:ReplicateTags"
      ],
      "Resource": "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET2/*"
    }
  ]
}

```

示例信任关系策略：

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Simple Storage Service 控制台用户指南》中的[这是主题标题](#)：

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketReplication](#)中的。

## put-bucket-request-payment

以下代码示例演示了如何使用 put-bucket-request-payment。

### AWS CLI

示例 1：为存储桶启用“申请方付款”配置

以下 put-bucket-request-payment 示例为指定的存储桶启用 requester pays。

```
aws s3api put-bucket-request-payment \  
  --bucket my-bucket \  
  --request-payment-configuration '{"Payer":"Requester"}'
```

此命令不生成任何输出。

示例 2：为存储桶禁用“申请方付款”配置

以下 `put-bucket-request-payment` 示例为指定的存储桶禁用 `requester pays`。

```
aws s3api put-bucket-request-payment \  
  --bucket my-bucket \  
  --request-payment-configuration '{"Payer":"BucketOwner"}'
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutBucketRequestPayment](#) 中的。

## put-bucket-tagging

以下代码示例演示了如何使用 `put-bucket-tagging`。

### AWS CLI

以下命令将标记配置应用于名为 `my-bucket` 的存储桶：

```
aws s3api put-bucket-tagging --bucket my-bucket --tagging file://tagging.json
```

文件 `tagging.json` 是当前文件夹中指定标签的 JSON 文档：

```
{  
  "TagSet": [  
    {  
      "Key": "organization",  
      "Value": "marketing"  
    }  
  ]  
}
```

或者，直接从命令行将标记配置应用于 `my-bucket`：

```
aws s3api put-bucket-tagging --bucket my-bucket --tagging
  'TagSet=[{Key=organization,Value=marketing}]'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketTagging](#)中的。

## put-bucket-versioning

以下代码示例演示了如何使用 put-bucket-versioning。

### AWS CLI

以下命令对于名为 my-bucket 的存储桶启用版本控制。

```
aws s3api put-bucket-versioning --bucket my-bucket --versioning-configuration
  Status=Enabled
```

以下命令启用版本控制，并使用 mfa 代码

```
aws s3api put-bucket-versioning --bucket my-bucket --versioning-configuration
  Status=Enabled --mfa "SERIAL 123456"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketVersioning](#)中的。

## put-bucket-website

以下代码示例演示了如何使用 put-bucket-website。

### AWS CLI

将静态网站配置应用于名为 my-bucket 的存储桶：

```
aws s3api put-bucket-website --bucket my-bucket --website-configuration file://
  website.json
```

文件 website.json 是当前文件夹中的一个 JSON 文档，用于指定网站的索引和错误页面：

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  }
}
```

```
    },  
    "ErrorDocument": {  
      "Key": "error.html"  
    }  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutBucketWebsite](#)中的。

## put-object-acl

以下代码示例演示了如何使用 `put-object-acl`。

### AWS CLI

以下命令 `full control` 向两个 AWS 用户 ( `user1@example.com` 和 `user2@example.com` ) 授予权限，向所有人 `read` 授予权限：

```
aws s3api put-object-acl --bucket MyBucket --key file.txt --grant-full-control  
emailaddress=user1@example.com,emailaddress=user2@example.com --grant-read  
uri=http://acs.amazonaws.com/groups/global/AllUsers
```

有关自定义 ACL ( `s3api` ACL 命令 ( 如 `put-object-acl` ) 使用相同的速记参数表示法 ) 的详细信息，请参阅 <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html>。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutObjectAcl](#)中的。

## put-object-legal-hold

以下代码示例演示了如何使用 `put-object-legal-hold`。

### AWS CLI

对对象应用法定保留

以下 `put-object-legal-hold` 示例在 `doc1.rtf` 对象上设置了法定保留。

```
aws s3api put-object-legal-hold \  
  --bucket my-bucket-with-object-lock \  
  --key doc1.rtf \  
  --legal-hold Status=ON
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutObjectLegalHold](#)中的。

## put-object-lock-configuration

以下代码示例演示了如何使用 `put-object-lock-configuration`。

### AWS CLI

在存储桶上设置对象锁定配置

以下 `put-object-lock-configuration` 示例在指定存储桶上设置了 50 天的对象锁定。

```
aws s3api put-object-lock-configuration \  
  --bucket my-bucket-with-object-lock \  
  --object-lock-configuration '{ "ObjectLockEnabled": "Enabled", "Rule":  
  { "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 50 } } }'
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutObjectLockConfiguration](#)中的。

## put-object-retention

以下代码示例演示了如何使用 `put-object-retention`。

### AWS CLI

为对象设置对象保留配置

以下 `put-object-retention` 示例为指定对象设置直到 2025 年 1 月 1 日的对象保留配置。

```
aws s3api put-object-retention \  
  --bucket my-bucket-with-object-lock \  
  --key doc1.rtf \  
  --retention '{ "Mode": "GOVERNANCE", "RetainUntilDate": "2025-01-01T00:00:00" }'
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutObjectRetention](#)中的。

## put-object-tagging

以下代码示例演示了如何使用 `put-object-tagging`。

### AWS CLI

在对象上设置标签

以下 `put-object-tagging` 示例在指定对象 `confidential` 上设置带有键 `designation` 和值的标签。

```
aws s3api put-object-tagging \  
  --bucket my-bucket \  
  --key doc1.rtf \  
  --tagging '{"TagSet": [{ "Key": "designation", "Value": "confidential" } ]}'
```

此命令不生成任何输出。

以下 `put-object-tagging` 示例在指定对象上设置多个标签集。

```
aws s3api put-object-tagging \  
  --bucket my-bucket-example \  
  --key doc3.rtf \  
  --tagging '{"TagSet": [{ "Key": "designation", "Value": "confidential" },  
  { "Key": "department", "Value": "finance" }, { "Key": "team", "Value":  
  "payroll" } ]}'
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutObjectTagging](#) 中的。

## put-object

以下代码示例演示了如何使用 `put-object`。

### AWS CLI

以下示例使用 `put-object` 命令将对象上传到 Amazon S3：

```
aws s3api put-object --bucket text-content --key dir-1/my_images.tar.bz2 --body  
my_images.tar.bz2
```

以下示例演示了如何上传视频文件（该视频文件是使用 Windows 文件系统语法指定的。）：

```
aws s3api put-object --bucket text-content --key dir-1/big-video-file.mp4 --body e:\media\videos\f-sharp-3-data-services.mp4
```

有关上传对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“上传对象”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutObject](#) 中的。

## put-public-access-block

以下代码示例演示了如何使用 put-public-access-block。

### AWS CLI

为存储桶设置阻止公有访问配置

以下 put-public-access-block 示例为指定存储桶设置限制性封禁公共访问配置。

```
aws s3api put-public-access-block \  
  --bucket my-bucket \  
  --public-access-block-configuration  
  "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutPublicAccessBlock](#) 中的。

## rb

以下代码示例演示了如何使用 rb。

### AWS CLI

示例 1：删除存储桶

以下 rb 命令删除存储桶。在此示例中，用户的存储桶是 mybucket。请注意，存储桶必须为空才能删除：

```
aws s3 rb s3://mybucket
```

输出：

```
remove_bucket: mybucket
```

## 示例 2：强制删除存储桶

以下 `rb` 命令使用 `--force` 参数首先删除存储桶中的所有对象，然后移除存储桶本身。在此示例中，用户的存储桶为 `mybucket`，其中的对象 `mybucket` 为 `test1.txt` 和 `test2.txt`：

```
aws s3 rb s3://mybucket \  
  --force
```

输出：

```
delete: s3://mybucket/test1.txt  
delete: s3://mybucket/test2.txt  
remove_bucket: mybucket
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [Rb](#)。

## restore-object

以下代码示例演示了如何使用 `restore-object`。

### AWS CLI

为对象创建还原请求

以下 `restore-object` 示例将存储桶 `my-glacier-bucket` 的指定 Amazon S3 Glacier 对象还原为 10 天。

```
aws s3api restore-object \  
  --bucket my-glacier-bucket \  
  --key doc1.rtf \  
  --restore-request Days=10
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RestoreObject](#) 中的。

## rm

以下代码示例演示了如何使用 `rm`。



## AWS CLI

### 示例 1：删除 S3 对象

以下rm命令删除单个 s3 对象：

```
aws s3 rm s3://mybucket/test2.txt
```

输出：

```
delete: s3://mybucket/test2.txt
```

### 示例 2：删除存储桶中的所有内容

以下rm命令在传递参数--recursive时递归删除指定存储桶和前缀下的所有对象。在此示例中，存储桶mybucket包含对象test1.txt和test2.txt：

```
aws s3 rm s3://mybucket \  
--recursive
```

输出：

```
delete: s3://mybucket/test1.txt  
delete: s3://mybucket/test2.txt
```

### 示例 3：删除存储桶中的所有内容，“.jpg”文件除外

以下rm命令在传递参数时递归删除指定存储桶和前缀下的所有对象，--recursive同时使用参数排除某些对象。--exclude在此示例中mybucket，存储桶包含对象test1.txt和test2.jpg：

```
aws s3 rm s3://mybucket/ \  
--recursive \  
--exclude "*.jpg"
```

输出：

```
delete: s3://mybucket/test1.txt
```

### 示例 4：删除存储桶中的所有内容，但指定前缀下的对象除外

以下`rm`命令在传递参数时递归删除指定存储桶和前缀下的所有对象，`--recursive`同时使用参数排除特定前缀下的所有对象`--exclude`。在此示例中`mybucket`，存储桶包含对象`test1.txt`和`another/test.txt`：

```
aws s3 rm s3://mybucket/ \  
  --recursive \  
  --exclude "another/*"
```

输出：

```
delete: s3://mybucket/test1.txt
```

示例 5：从 S3 接入点删除对象

以下`rm`命令从接入点 (`mykey`) 中删除单个对象 (`myaccesspoint`)。:: 以下`rm`命令从接入点 (`mykey`) 中删除单个对象 (`myaccesspoint`)。

```
aws s3 rm s3://arn:aws:s3:us-west-2:123456789012:accesspoint/myaccesspoint/mykey
```

输出：

```
delete: s3://arn:aws:s3:us-west-2:123456789012:accesspoint/myaccesspoint/mykey
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [Rm](#)。

## select-object-content

以下代码示例演示了如何使用 `select-object-content`。

### AWS CLI

基于 SQL 语句筛选 Amazon S3 对象的内容

以下 `select-object-content` 示例使用指定的 SQL 语句筛选 `my-data-file.csv` 对象并将输出发送到文件。

```
aws s3api select-object-content \  
  --bucket my-bucket \  
  --key my-data-file.csv \  
  --expression "select * from s3object limit 100" \  
  --output my-output-file
```

```
--expression-type 'SQL' \  
--input-serialization '{"CSV": {}, "CompressionType": "NONE"}' \  
--output-serialization '{"CSV": {}}' "output.csv"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SelectObjectContent](#)中的。

## sync

以下代码示例演示了如何使用 sync。

### AWS CLI

#### 示例 1：将所有本地对象同步到指定存储桶

以下sync命令通过将本地文件上传到 S3，将本地目录中的对象同步到指定的前缀和存储桶。如果本地文件的大小与 S3 对象的大小不同，本地文件的上次修改时间晚于 S3 对象的上次修改时间，或者本地文件在指定的存储桶和前缀下不存在，则需要上传本地文件。在此示例中，用户将存储桶同步mybucket到本地当前目录。本地当前目录包含文件test.txt和test2.txt。存储桶不mybucket包含任何对象。

```
aws s3 sync . s3://mybucket
```

输出：

```
upload: test.txt to s3://mybucket/test.txt  
upload: test2.txt to s3://mybucket/test2.txt
```

#### 示例 2：将指定的 S3 存储桶中的所有 S3 对象同步到另一个存储桶

以下sync命令通过复制 S3 对象，将指定前缀下的对象和存储桶同步到另一个指定前缀下的对象和存储桶。如果两个 S3 对象的大小不同，源的最后修改时间晚于目标的上次修改时间，或者 S3 对象在指定的存储桶和前缀目标下不存在，则需要复制 S3 对象。

在此示例中，用户将存储桶同步mybucket到存储桶mybucket2。存储桶mybucket包含对象test.txt和test2.txt。存储桶不mybucket2包含任何对象：

```
aws s3 sync s3://mybucket s3://mybucket2
```

输出：

```
copy: s3://mybucket/test.txt to s3://mybucket2/test.txt
copy: s3://mybucket/test2.txt to s3://mybucket2/test2.txt
```

### 示例 3：将指定 S3 存储桶中的所有 S3 对象同步到本地目录

以下sync命令通过下载 S3 对象将文件从指定的 S3 存储桶同步到本地目录。如果 S3 对象的大小与本地文件的大小不同，S3 对象的上次修改时间晚于本地文件的上次修改时间，或者本地目录中不存在 S3 对象，则需要下载。请注意，从 S3 下载对象时，本地文件的最后修改时间将更改为 S3 对象的上次修改时间。在此示例中，用户将存储桶同步mybucket到当前本地目录。存储桶mybucket包含对象test.txt和test2.txt。当前本地目录中没有文件：

```
aws s3 sync s3://mybucket .
```

输出：

```
download: s3://mybucket/test.txt to test.txt
download: s3://mybucket/test2.txt to test2.txt
```

### 示例 4：将所有本地对象同步到指定存储桶并删除所有不匹配的文件

以下sync命令通过将本地文件上传到 S3，将指定前缀和存储桶下的对象同步到本地目录中的文件。由于--delete参数的原因，任何存在于指定前缀和存储桶下但不存在于本地目录中的文件都将被删除。在此示例中，用户将存储桶同步mybucket到本地当前目录。本地当前目录包含文件test.txt和test2.txt。存储桶mybucket包含对象test3.txt：

```
aws s3 sync . s3://mybucket \
  --delete
```

输出：

```
upload: test.txt to s3://mybucket/test.txt
upload: test2.txt to s3://mybucket/test2.txt
delete: s3://mybucket/test3.txt
```

### 示例 5：将除.jpg`文件之外的所有本地对象同步到指定存储桶

以下sync命令通过将本地文件上传到 S3，将指定前缀和存储桶下的对象同步到本地目录中的文件。由于--exclude参数的原因，所有与 S3 和本地模式相匹配的文件都将被排除在同

步之外。在此示例中，用户将存储桶同步mybucket到本地当前目录。本地当前目录包含文件test.jpg和test2.txt。该存储桶mybucket包含test.jpg与本地存储桶大小不同的对象test.jpg：

```
aws s3 sync . s3://mybucket \  
  --exclude "*.jpg"
```

输出：

```
upload: test2.txt to s3://mybucket/test2.txt
```

示例 6：将除.jpg`文件之外的所有本地对象同步到指定存储桶

以下sync命令通过下载 S3 对象将本地目录下的文件同步到指定前缀下的对象和存储桶。此示例使用--exclude参数标志从sync命令中排除指定的目录和 S3 前缀。在此示例中，用户将本地当前目录同步到存储桶mybucket。本地当前目录包含文件test.txt和another/test2.txt。存储桶mybucket包含对象another/test5.txt和test1.txt：

```
aws s3 sync s3://mybucket/ . \  
  --exclude "*another/*"
```

输出：

```
download: s3://mybucket/test1.txt to test1.txt
```

示例 7：在不同区域的存储桶之间同步所有对象

以下sync命令可在不同区域的两个存储桶之间同步文件：

```
aws s3 sync s3://my-us-west-2-bucket s3://my-us-east-1-bucket \  
  --source-region us-west-2 \  
  --region us-east-1
```

输出：

```
download: s3://my-us-west-2-bucket/test1.txt to s3://my-us-east-1-bucket/test1.txt
```

示例 8：同步到 S3 接入点

以下sync命令将当前目录同步到接入点 (myaccesspoint) :

```
aws s3 sync . s3://arn:aws:s3:us-west-2:123456789012:accesspoint/myaccesspoint/
```

输出 :

```
upload: test.txt to s3://arn:aws:s3:us-west-2:123456789012:accesspoint/
myaccesspoint/test.txt
upload: test2.txt to s3://arn:aws:s3:us-west-2:123456789012:accesspoint/
myaccesspoint/test2.txt
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的“[同步](#)”。

## upload-part-copy

以下代码示例演示了如何使用 upload-part-copy。

### AWS CLI

通过从现有对象中复制数据作为数据源来上传部分对象

以下upload-part-copy示例通过从现有对象复制数据作为数据源来上传部件。

```
aws s3api upload-part-copy \
  --bucket my-bucket \
  --key "Map_Data_June.mp4" \
  --copy-source "my-bucket/copy_of_Map_Data_June.mp4" \
  --part-number 1 \
  --upload-id
  "bq0tdE1CDpWQYRPLHuNG50xAT6pA5D.m_RiBy0gg0H6b13pVRY7QjvL1f75iFdJqp_2wztk5hvpUM2SesXgrzbehG5"
```

输出 :

```
{
  "CopyPartResult": {
    "LastModified": "2019-12-13T23:16:03.000Z",
    "ETag": "\"711470fc377698c393d94aed6305e245\""
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UploadPartCopy](#)中的。

## upload-part

以下代码示例演示了如何使用 `upload-part`。

### AWS CLI

以下命令上传使用 `create-multipart-upload` 命令启动的分段上传中的第一个分段：

```
aws s3api upload-part --bucket my-bucket --key 'multipart/01' --part-number 1 --body
part01 --upload-id
"dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3URC
```

`body` 选项采用本地文件的名称或路径进行上传（不要使用 `file://` 前缀）。最小分段大小为 5 MB。上传 ID 由 `create-multipart-upload` 返回，也可以使用 `list-multipart-uploads` 进行检索。存储桶和键是在您创建分段上传时指定的。

输出：

```
{
  "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

保存每个分段的 ETag 值以备后用。需要这些值才能完成分段上传。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UploadPart](#) 中的。

## website

以下代码示例演示了如何使用 `website`。

### AWS CLI

将 S3 存储桶配置为静态网站

以下命令将名为静态网站的存储桶配置 `my-bucket` 为静态网站。索引文档选项指定了访问者 `my-bucket` 在导航到网站 URL 时将被定向到的文件。在本例中，存储桶位于 `us-west-2` 区域，因此该网站将显示在 `http://my-bucket.s3-website-us-west-2.amazonaws.com`

存储桶中显示在静态站点上的所有文件都必须配置为允许访问者打开它们。文件权限是与存储桶网站配置分开配置的。

```
aws s3 website s3://my-bucket/ \
```

```
--index-document index.html \  
--error-document error.html
```

有关在 Amazon S3 中托管静态网站的信息，请参阅《亚马逊简单存储服务开发者指南》中的[托管静态网站](#)。

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的“[网站](#)”。

## 使用 Amazon S3 控制示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon S3 Control 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **create-access-point**

以下代码示例演示了如何使用 create-access-point。

#### AWS CLI

##### 创建接入点

以下 create-access-point 示例为账户 123456789012 business-records 中的存储桶创建了一个名为 finance-ap 的接入点。在运行此示例之前，请将接入点名称、存储桶名称和账号替换为适合您的用例的值。

```
aws s3control create-access-point \  
--account-id 123456789012 \  

```



```
--bucket business-records \  
--name finance-ap
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon 简单存储服务开发者指南》中的[创建接入点](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAccessPoint](#)中的。

## create-job

以下代码示例演示了如何使用 create-job。

### AWS CLI

#### 创建 Amazon S3 批量操作任务

以下create-job示例创建了一个 Amazon S3 批处理操作任务，将对象标记为confidential`  
in the bucket ``employee-records。

```
aws s3control create-job \  
  --account-id 123456789012 \  
  --operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"confidential",  
"Value":"true"}] }}' \  
  --report '{"Bucket":"arn:aws:s3:::employee-records-logs","Prefix":"batch-op-  
create-job",  
"Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \  
  --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":  
["Bucket","Key"]},"Location":{"ObjectArn":"arn:aws:s3:::employee-records-logs/inv-  
report/7a6a9be4-072c-407e-85a2-  
ec3e982f773e.csv"},"ETag":"69f52a4e9f797e987155d9c8f5880897"}' \  
  --priority 42 \  
  --role-arn arn:aws:iam::123456789012:role/S3BatchJobRole
```

输出：

```
{  
  "JobId": "93735294-df46-44d5-8638-6356f335324e"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateJob](#)中的。

## delete-access-point-policy

以下代码示例演示了如何使用 delete-access-point-policy。

### AWS CLI

#### 删除接入点策略

以下delete-access-point-policy示例从账户 123456789012 中命名的接入点finance-ap中删除接入点策略。在运行此示例之前，请将接入点名称和账号替换为适合您的用例的值。

```
aws s3control delete-access-point-policy \  
  --account-id 123456789012 \  
  --name finance-ap
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊简单存储服务开发人员指南](#)》中的[使用 Amazon S3 接入点管理数据访问](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAccessPointPolicy](#)中的。

## delete-access-point

以下代码示例演示了如何使用 delete-access-point。

### AWS CLI

#### 删除接入点

以下delete-access-point示例删除了账户 123456789012 finance-ap 中名为的接入点。在运行此示例之前，请将接入点名称和账号替换为适合您的用例的值。

```
aws s3control delete-access-point \  
  --account-id 123456789012 \  
  --name finance-ap
```

此命令不生成任何输出。

有关更多信息，请参阅《[亚马逊简单存储服务开发人员指南](#)》中的[使用 Amazon S3 接入点管理数据访问](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAccessPoint](#)中的。

## delete-public-access-block

以下代码示例演示了如何使用 delete-public-access-block。

### AWS CLI

删除账户的封锁公共访问设置

以下delete-public-access-block示例删除了指定账户的封禁公共访问设置。

```
aws s3control delete-public-access-block \  
  --account-id 123456789012
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePublicAccessBlock](#)中的。

## describe-job

以下代码示例演示了如何使用 describe-job。

### AWS CLI

描述 Amazon S3 批量操作任务

以下内容describe-job提供了指定批处理作业的配置参数和状态。

```
aws s3control describe-job \  
  --account-id 123456789012 \  
  --job-id 93735294-df46-44d5-8638-6356f335324e
```

输出：

```
{  
  "Job": {  
    "TerminationDate": "2019-10-03T21:49:53.944Z",  
    "JobId": "93735294-df46-44d5-8638-6356f335324e",  
    "FailureReasons": [],  
    "Manifest": {  
      "Spec": {
```

```

        "Fields": [
            "Bucket",
            "Key"
        ],
        "Format": "S3BatchOperations_CSV_20180820"
    },
    "Location": {
        "ETag": "69f52a4e9f797e987155d9c8f5880897",
        "ObjectArn": "arn:aws:s3:::employee-records-logs/inv-report/7a6a9be4-072c-407e-85a2-ec3e982f773e.csv"
    }
},
"Operation": {
    "S3PutObjectTagging": {
        "TagSet": [
            {
                "Value": "true",
                "Key": "confidential"
            }
        ]
    }
},
"RoleArn": "arn:aws:iam::123456789012:role/S3BatchJobRole",
"ProgressSummary": {
    "TotalNumberOfTasks": 8,
    "NumberOfTasksFailed": 0,
    "NumberOfTasksSucceeded": 8
},
"Priority": 42,
"Report": {
    "ReportScope": "AllTasks",
    "Format": "Report_CSV_20180820",
    "Enabled": true,
    "Prefix": "batch-op-create-job",
    "Bucket": "arn:aws:s3:::employee-records-logs"
},
"JobArn": "arn:aws:s3:us-west-2:123456789012:job/93735294-df46-44d5-8638-6356f335324e",
"CreationTime": "2019-10-03T21:48:48.048Z",
"Status": "Complete"
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeJob](#)中的。

## get-access-point-policy-status

以下代码示例演示了如何使用 `get-access-point-policy-status`。

### AWS CLI

#### 检索接入点策略状态

以下`get-access-point-policy-status`示例检索账户 123456789012 `finance-ap` 中命名的接入点的接入点策略状态。接入点策略状态表明接入点的策略是否允许公共访问。在运行此示例之前，请将接入点名称和账号替换为适合您的用例的值。

```
aws s3control get-access-point-policy-status \  
  --account-id 123456789012 \  
  --name finance-ap
```

#### 输出：

```
{  
  "PolicyStatus": {  
    "IsPublic": false  
  }  
}
```

有关何时将接入点策略视为公开的更多信息，请参阅 [《Amazon 简单存储服务开发者指南》](#) 中的“公开”的含义。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAccessPointPolicyStatus](#)中的。

## get-access-point-policy

以下代码示例演示了如何使用 `get-access-point-policy`。

### AWS CLI

#### 检索接入点策略

以下`get-access-point-policy`示例从账户 123456789012 中命名的接入点检索接`finance-ap`入点策略。在运行此示例之前，请将接入点名称和账号替换为适合您的用例的值。

```
aws s3control get-access-point-policy \  
  --account-id 123456789012 \  
  --name finance-ap
```

```
--account-id 123456789012 \  
--name finance-ap
```

输出：

```
{  
  "Policy": "{\n\"Version\":\n\"2012-10-17\",  
\"Statement\":[[{\n\"Effect\":\n\"Allow\",  
\"Principal\":{\n\"AWS\":\n\"arn:aws:iam::123456789012:role/Admin\"},  
\"Action\":\n\"s3:GetObject\",  
\"Resource\":\n\"arn:aws:s3:us-west-2:123456789012:accesspoint/  
finance-ap/object/records/*\"}]]}"  
}
```

有关更多信息，请参阅 [《亚马逊简单存储服务开发人员指南》](#) 中的使用 Amazon S3 接入点管理数据访问。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetAccessPointPolicy](#) 中的。

## get-access-point

以下代码示例演示了如何使用 get-access-point。

### AWS CLI

检索接入点配置详细信息

以下 get-access-point 示例检索账户 123456789012 finance-ap 中命名的接入点的配置详细信息。在运行此示例之前，请将接入点名称和账号替换为适合您的用例的值。

```
aws s3control get-access-point \  
--account-id 123456789012 \  
--name finance-ap
```

输出：

```
{  
  "Name": "finance-ap",  
  "Bucket": "business-records",  
  "NetworkOrigin": "Internet",  
  "PublicAccessBlockConfiguration": {  
    "BlockPublicAcls": false,  
    "IgnorePublicAcls": false,
```

```
    "BlockPublicPolicy": false,
    "RestrictPublicBuckets": false
  },
  "CreationDate": "2020-01-01T00:00:00Z"
}
```

有关更多信息，请参阅《[亚马逊简单存储服务开发人员指南](#)》中的使用 Amazon S3 接入点管理数据访问。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetAccessPoint](#) 中的。

## get-multi-region-access-point-routes

以下代码示例演示了如何使用 `get-multi-region-access-point-routes`。

### AWS CLI

查询当前的多区域接入点路由配置

以下 `get-multi-region-access-point-routes` 示例返回指定多区域接入点的当前路由配置。

```
aws s3control get-multi-region-access-point-routes \
  --region Region \
  --account-id 111122223333 \
  --mrap MultiRegionAccessPoint_ARN
```

输出：

```
{
  "Mrap": "arn:aws:s3::111122223333:accesspoint/0000000000000000.mrap",
  "Routes": [
    {
      "Bucket": "DOC-EXAMPLE-BUCKET-1",
      "Region": "ap-southeast-2",
      "TrafficDialPercentage": 100
    },
    {
      "Bucket": "DOC-EXAMPLE-BUCKET-2",
      "Region": "us-west-1",
      "TrafficDialPercentage": 0
    }
  ]
}
```

```
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMultiRegionAccessPointRoutes](#)中的。

## get-public-access-block

以下代码示例演示了如何使用 `get-public-access-block`。

### AWS CLI

列出账户的公开封禁访问设置

以下`get-public-access-block`示例显示了指定账户的封锁公共访问设置。

```
aws s3control get-public-access-block \
  --account-id 123456789012
```

输出：

```
{
  "PublicAccessBlockConfiguration": {
    "BlockPublicPolicy": true,
    "RestrictPublicBuckets": true,
    "IgnorePublicAcls": true,
    "BlockPublicAcls": true
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPublicAccessBlock](#)中的。

## list-access-points

以下代码示例演示了如何使用 `list-access-points`。

### AWS CLI

示例 1：检索账户的所有接入点列表

以下`list-access-points`示例显示了挂载到账户 123456789012 拥有的存储桶的所有接入点的列表。



```
aws s3control list-access-points \  
--account-id 123456789012
```

输出：

```
{  
  "AccessPointList": [  
    {  
      "Name": "finance-ap",  
      "NetworkOrigin": "Internet",  
      "Bucket": "business-records"  
    },  
    {  
      "Name": "managers-ap",  
      "NetworkOrigin": "Internet",  
      "Bucket": "business-records"  
    },  
    {  
      "Name": "private-network-ap",  
      "NetworkOrigin": "VPC",  
      "VpcConfiguration": {  
        "VpcId": "1a2b3c"  
      },  
      "Bucket": "business-records"  
    },  
    {  
      "Name": "customer-ap",  
      "NetworkOrigin": "Internet",  
      "Bucket": "external-docs"  
    },  
    {  
      "Name": "public-ap",  
      "NetworkOrigin": "Internet",  
      "Bucket": "external-docs"  
    }  
  ]  
}
```

示例 2：检索存储桶的所有接入点列表

以下list-access-points示例检索挂载到账户 123456789012 external-docs 拥有的存储桶的所有接入点的列表。

```
aws s3control list-access-points \  
  --account-id 123456789012 \  
  --bucket external-docs
```

输出：

```
{  
  "AccessPointList": [  
    {  
      "Name": "customer-ap",  
      "NetworkOrigin": "Internet",  
      "Bucket": "external-docs"  
    },  
    {  
      "Name": "public-ap",  
      "NetworkOrigin": "Internet",  
      "Bucket": "external-docs"  
    }  
  ]  
}
```

有关更多信息，请参阅《[亚马逊简单存储服务开发人员指南](#)》中的使用 Amazon S3 接入点管理数据访问。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAccessPoints](#)中的。

## list-jobs

以下代码示例演示了如何使用 list-jobs。

### AWS CLI

要列出账户 Amazon S3 批量操作任务

以下list-jobs示例列出了指定账户最近的所有批量操作任务。

```
aws s3control list-jobs \  
  --account-id 123456789012
```

输出：

```
{
  "Jobs": [
    {
      "Operation": "S3PutObjectTagging",
      "ProgressSummary": {
        "NumberOfTasksFailed": 0,
        "NumberOfTasksSucceeded": 8,
        "TotalNumberOfTasks": 8
      },
      "CreationTime": "2019-10-03T21:48:48.048Z",
      "Status": "Complete",
      "JobId": "93735294-df46-44d5-8638-6356f335324e",
      "Priority": 42
    },
    {
      "Operation": "S3PutObjectTagging",
      "ProgressSummary": {
        "NumberOfTasksFailed": 0,
        "NumberOfTasksSucceeded": 0,
        "TotalNumberOfTasks": 0
      },
      "CreationTime": "2019-10-03T21:46:07.084Z",
      "Status": "Failed",
      "JobId": "3f3c7619-02d3-4779-97f6-1d98dd313108",
      "Priority": 42
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListJobs](#)中的。

## put-access-point-policy

以下代码示例演示了如何使用 put-access-point-policy。

### AWS CLI

#### 设置接入点策略

以下put-access-point-policy示例将接入点的指定接入点策略置于账户 123456789012 finance-ap 中。如果接入点finance-ap已有策略，则此命令将现有策略替换为该命令中指定的策略。在运行此示例之前，请将账号、接入点名称和策略声明替换为适合您的用例的值。

```
aws s3control put-access-point-policy \  
  --account-id 123456789012 \  
  --name finance-ap \  
  --policy file://ap-policy.json
```

ap-policy.json 的内容：

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::123456789012:user/Alice"  
      },  
      "Action": "s3:GetObject",  
      "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/finance-ap/  
object/Alice/*"  
    }  
  ]  
}
```

此命令不生成任何输出。

有关更多信息，请参阅 [《亚马逊简单存储服务开发人员指南》](#) 中的 [使用 Amazon S3 接入点管理数据访问](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutAccessPointPolicy](#) 中的。

## put-public-access-block

以下代码示例演示了如何使用 put-public-access-block。

### AWS CLI

#### 编辑账户的封锁公共访问设置

以下 put-public-access-block 示例将指定账户的所有封禁公共访问 true 设置切换为。

```
aws s3control put-public-access-block \  
  --account-id 123456789012 \  
  --policy file://ap-policy.json
```

```
--public-access-block-configuration '{"BlockPublicAcls": true,
"IgnorePublicAcls": true, "BlockPublicPolicy": true, "RestrictPublicBuckets":
true}'
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutPublicAccessBlock](#)中的。

## submit-multi-region-access-point-routes

以下代码示例演示了如何使用 submit-multi-region-access-point-routes。

### AWS CLI

更新您的多区域接入点路由配置

以下submit-multi-region-access-point-routes示例更新了您的多区域接入点的ap-southeast-2区域DOC-EXAMPLE-BUCKET-2中DOC-EXAMPLE-BUCKET-1和区域中的路由状态。

```
aws s3control submit-multi-region-access-point-routes \
  --region ap-southeast-2 \
  --account-id 111122223333 \
  --mrap MultiRegionAccessPoint_ARN \
  --route-updates Bucket=DOC-EXAMPLE-BUCKET-1,TrafficDialPercentage=100
Bucket=DOC-EXAMPLE-BUCKET-2,TrafficDialPercentage=0
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SubmitMultiRegionAccessPointRoutes](#)中的。

## update-job-priority

以下代码示例演示了如何使用 update-job-priority。

### AWS CLI

更新 Amazon S3 批处理操作任务的任务优先级

以下update-job-priority示例将指定的任务更新为新的优先级。

```
aws s3control update-job-priority \
```

```
--account-id 123456789012 \  
--job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \  
--priority 52
```

输出：

```
{  
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386",  
  "Priority": 52  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateJobPriority](#)中的。

## update-job-status

以下代码示例演示了如何使用 update-job-status。

### AWS CLI

更新 Amazon S3 批处理操作任务的状态

以下update-job-status示例取消了正在等待批准的指定任务。

```
aws s3control update-job-status \  
--account-id 123456789012 \  
--job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \  
--requested-job-status Cancelled
```

输出：

```
{  
  "Status": "Cancelled",  
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386"  
}
```

以下update-job-status示例确认并运行正在等待批准的指定内容。

```
aws s3control update-job-status \  
--account-id 123456789012 \  
--requested-job-status InProgress
```

```
--job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \  
--requested-job-status Ready
```

Output::

```
{  
  "Status": "Ready",  
  "JobId": "5782949f-3301-4fb3-be34-8d5bab54dbca"  
}
```

以下update-job-status示例取消正在运行的指定作业。

```
aws s3control update-job-status \  
  --account-id 123456789012 \  
  --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \  
  --requested-job-status Cancelled
```

Output::

```
{  
  "Status": "Cancelling",  
  "JobId": "5782949f-3301-4fb3-be34-8d5bab54dbca"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateJobStatus](#)中的。

## S3 Glacier 示例使用 AWS CLI

以下代码示例向您展示了如何在 S3 Glacier 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

## 操作

### **abort-multipart-upload**

以下代码示例演示了如何使用 `abort-multipart-upload`。

#### AWS CLI

以下命令删除正在进行的分段上传到名为的文件库：`my-vault`

```
aws glacier abort-multipart-upload --account-id - --vault-name my-vault
--upload-id 19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-
0ssZtLqyFu7sY1_lR7vgFuJV6NtcV5zpsJ
```

此命令不生成任何输出。Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。上传 ID 由 `aws glacier initiate-multipart-upload` 命令返回，也可以使用 `aws glacier list-multipart-uploads` 获取它。

有关使用 CLI 分段上传到 Amazon Glacier 的更多信息，请参阅 AWS CL AWS I 用户指南中的使用亚马逊 Glacier。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AbortMultipartUpload](#) 中的。

### **abort-vault-lock**

以下代码示例演示了如何使用 `abort-vault-lock`。

#### AWS CLI

中止正在进行的文件库锁定进程

以下 `abort-vault-lock` 示例从指定文件库中删除文件库锁定策略，并将文件库锁定的锁定状态重置为已解锁。

```
aws glacier abort-vault-lock \
--account-id - \
--vault-name MyVaultName
```

此命令不生成任何输出。

有关更多信息，请参阅 [Amazon Glacier API 开发者指南中的中止文件库锁定（删除锁定策略）](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AbortVaultLock](#)中的。

## add-tags-to-vault

以下代码示例演示了如何使用 `add-tags-to-vault`。

### AWS CLI

以下命令向名为 `my-vault` 的文件库中添加两个标签：

```
aws glacier add-tags-to-vault --account-id - --vault-name my-vault --tags
id=1234,date=july2015
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddTagsToVault](#)中的。

## complete-multipart-upload

以下代码示例演示了如何使用 `complete-multipart-upload`。

### AWS CLI

以下命令完成 3 MiB 档案的分段上传：

```
aws glacier complete-multipart-upload --archive-size 3145728 --checksum
9628195fcdcbbe76cdde456d4646fa7de5f219fb39823836d81f0cc0e18aa67
--upload-id 19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-
0ssZtLqyFu7sY1_lR7vgFuJV6NtcV5zpsJ --account-id - --vault-name my-vault
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

上传 ID 由 `aws glacier initiate-multipart-upload` 命令返回，也可以使用 `aws glacier list-multipart-uploads` 获取它。校验和参数采用十六进制格式的档案的 SHA-256 树形哈希。

有关使用 CLI 分段上传到 Amazon Glacier 的更多信息，包括计算树形哈希的说明，请参阅 AWS CL AWS I 用户指南中的使用 Amazon Glacier。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CompleteMultipartUpload](#)中的。

## complete-vault-lock

以下代码示例演示了如何使用 `complete-vault-lock`。

### AWS CLI

完成正在进行的文件库锁定流程

以下`complete-vault-lock`示例完成了指定文件库的锁定进度，并将文件库锁定的锁定状态设置为Locked。当你运行时，你会得到`lock-id`参数的值`initiate-lock-process`。

```
aws glacier complete-vault-lock \  
  --account-id - \  
  --vault-name MyVaultName \  
  --lock-id 9QZgEXAMPLEPhvL6xEXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Glacier API 开发者指南中的[完成文件库锁定 \( POST LockID \)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CompleteVaultLock](#)中的。

## create-vault

以下代码示例演示了如何使用 `create-vault`。

### AWS CLI

以下命令创建名为 `my-vault` 的新文件库：

```
aws glacier create-vault --vault-name my-vault --account-id -
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVault](#)中的。

## delete-archive

以下代码示例演示了如何使用 `delete-archive`。

## AWS CLI

从文件库中删除存档

以下 `delete-archive` 示例从 `example_vault` 中删除指定存档。

```
aws glacier delete-archive \  
  --account-id 111122223333 \  
  --vault-name example_vault \  
  --archive-id Sc0u9ZP8yaWkmh-XG1IvAVprtLhaLCGnNwN15I5x9HqPIkX5mjc0DrId3Ln-  
  Gi_k2HzmlIDZUz117KSdVMdMXLuFWi9PJUitxW073edQ43eT1MwkH0pd9zVSAuV_XXZBVhKhyGhJ7w
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteArchive](#)中的。

## `delete-vault-access-policy`

以下代码示例演示了如何使用 `delete-vault-access-policy`。

## AWS CLI

移除文件库的访问策略

以下`delete-vault-access-policy`示例删除了指定文件库的访问策略。

```
aws glacier delete-vault-access-policy \  
  --account-id 111122223333 \  
  --vault-name example_vault
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVaultAccessPolicy](#)中的。

## `delete-vault-notifications`

以下代码示例演示了如何使用 `delete-vault-notifications`。

## AWS CLI

删除文件库的 SNS 通知

以下 `delete-vault-notifications` 示例演示了如何删除 Amazon Simple Notification Service (Amazon SNS) 针对指定文件库发送的通知。

```
aws glacier delete-vault-notifications \  
  --account-id 111122223333 \  
  --vault-name example_vault
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVaultNotifications](#)中的。

## **delete-vault**

以下代码示例演示了如何使用 `delete-vault`。

### AWS CLI

以下命令删除名为 `my-vault` 的文件库：

```
aws glacier delete-vault --vault-name my-vault --account-id -
```

此命令不生成任何输出。Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVault](#)中的。

## **describe-job**

以下代码示例演示了如何使用 `describe-job`。

### AWS CLI

以下命令检索名为 `my-vault` 的文件库中有关库存检索任务的信息：

```
aws glacier describe-job --account-id - --vault-name my-  
vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3R1oGduS7Eg-  
R047Yc6FxsdsGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61ds04CnMW
```

输出：

```
{
```

```
"InventoryRetrievalParameters": {
  "Format": "JSON"
},
"VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-vault",
"Completed": false,
"JobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
R047Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_Xq1NHS61ds04CnMW",
"Action": "InventoryRetrieval",
"CreationDate": "2015-07-17T20:23:41.616Z",
"StatusCode": "InProgress"
}
```

任务 ID 可以在 `aws glacier initiate-job` 和 `aws glacier list-jobs` 的输出中找到。Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeJob](#) 中的。

## describe-vault

以下代码示例演示了如何使用 `describe-vault`。

### AWS CLI

以下命令检索名为 `my-vault` 的文件库的相关数据：

```
aws glacier describe-vault --vault-name my-vault --account-id -
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeVault](#) 中的。

## get-data-retrieval-policy

以下代码示例演示了如何使用 `get-data-retrieval-policy`。

### AWS CLI

以下命令获取正在使用的账户的数据检索策略：

```
aws glacier get-data-retrieval-policy --account-id -
```

输出：

```
{
  "Policy": {
    "Rules": [
      {
        "BytesPerHour": 10737418240,
        "Strategy": "BytesPerHour"
      }
    ]
  }
}
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDataRetrievalPolicy](#) 中的。

## get-job-output

以下代码示例演示了如何使用 get-job-output。

### AWS CLI

以下命令将文件库清单任务的输出保存到名为 output.json 的当前目录中的某个文件中：

```
aws glacier get-job-output --account-id - --vault-name my-
vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
R047Yc6FxsdBgf_Q2DK5Ejh18CnTS5XW4_Xq1NHS61ds04CnMW output.json
```

可在 aws glacier list-jobs 输出中找到 job-id。请注意，输出文件名是一个位置参数，不以选项名称作为前缀。Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

输出：

```
{
  "status": 200,
  "acceptRanges": "bytes",
  "contentType": "application/json"
}
```

output.json:

```
{
  "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-vault",
  "InventoryDate": "2015-04-07T00:26:18Z",
  "ArchiveList": [
    {
      "ArchiveId": "kKB7ymWJVpPSwhGP6ycS0Aekp9ZYe_--zM_mw6k76ZFGEIWQX-ybtRDvc2VkpSDtFkmQrj0IRQLSGsNuDp-AJVlu2ccmDSyDUmZwKwbpAdGATGDiB3hH00bjbGehXTcApVud_wyDw",
      "ArchiveDescription": "multipart upload test",
      "CreationDate": "2015-04-06T22:24:34Z",
      "Size": 3145728,
      "SHA256TreeHash": "9628195fcdcbcb"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetJobOutput](#)中的。

## get-vault-access-policy

以下代码示例演示了如何使用 get-vault-access-policy。

### AWS CLI

检索文件库的访问策略

以下get-vault-access-policy示例检索指定文件库的访问策略。

```
aws glacier get-vault-access-policy \
  --account-id 111122223333 \
  --vault-name example_vault
```

输出：

```
{
  "policy": {
    "Policy": "{\n\"Version\": \"2012-10-17\", \"Statement\": [\n\n{\n\n\"Effect\": \"Allow\", \"Principal\": {\n\n\"AWS\": \"arn:aws:iam::444455556666:root\"},\n\n\"Action\": \"glacier:ListJobs\", \"Resource\": \"arn:aws:glacier:us-east-1:111122223333:vaults/example_vault\"},\n\n{\n\n\"Effect\": \"Allow\", \"Principal\": {\n\n\"AWS\": \"arn:aws:iam::444455556666:root\"},\n\n\"Action\": \"glacier:UploadArchive\", \"Resource\": \"arn:aws:glacier:us-east-1:111122223333:vaults/example_vault\"}]]}"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVaultAccessPolicy](#)中的。

## get-vault-lock

以下代码示例演示了如何使用 `get-vault-lock`。

### AWS CLI

获取文件库锁的详细信息

以下`get-vault-lock`示例检索了有关指定文件库的锁的详细信息。

```
aws glacier get-vault-lock \  
  --account-id - \  
  --vault-name MyVaultName
```

输出：

```
{  
  "Policy": "{\n\"Version\": \"2012-10-17\", \"Statement\": [\n{\n\"Sid\": \"Define-vault-lock\", \"Effect\": \"Deny\", \"Principal\": {\n\"AWS\": \"arn:aws:iam::999999999999:root\"}, \"Action\": \"glacier:DeleteArchive\", \"Resource\": \"arn:aws:glacier:us-west-2:999999999999:vaults/MyVaultName\", \"Condition\": {\n\"NumericLessThanEquals\": {\n\"glacier:ArchiveAgeinDays\": \"365\"}}}}]",  
  "State": "Locked",  
  "CreationDate": "2019-07-29T22:25:28.640Z"  
}
```

有关更多信息，请参阅 Amazon Glacier API 开发者[指南中的获取文件库锁定（获取锁定策略）](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVaultLock](#)中的。

## get-vault-notifications

以下代码示例演示了如何使用 `get-vault-notifications`。

### AWS CLI

以下命令获取名为 `my-vault` 的文件库的通知配置的描述：

```
aws glacier get-vault-notifications --account-id - --vault-name my-vault
```

输出：



```
{
  "vaultNotificationConfig": {
    "Events": [
      "InventoryRetrievalCompleted",
      "ArchiveRetrievalCompleted"
    ],
    "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault"
  }
}
```

如果没有为该文件库配置任何通知，则会返回错误。Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVaultNotifications](#)中的。

## initiate-job

以下代码示例演示了如何使用 `initiate-job`。

### AWS CLI

以下命令启动任务以获取文件my-vault库清单：

```
aws glacier initiate-job --account-id - --vault-name my-vault --job-parameters
'{"Type": "inventory-retrieval"}
```

输出：

```
{
  "location": "/0123456789012/vaults/my-vault/jobs/
zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
R047Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_Xq1NHS61ds04CnMW",
  "jobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
R047Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_Xq1NHS61ds04CnMW"
}
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

以下命令启动从文件库my-vault中取回档案的任务：

```
aws glacier initiate-job --account-id - --vault-name my-vault --job-parameters
file://job-archive-retrieval.json
```

`job-archive-retrieval.json` 是本地文件夹中的一个 JSON 文件，用于指定任务类型、档案 ID 和一些可选参数：

```
{
  "Type": "archive-retrieval",
  "ArchiveId": "kKB7ymWJVpPSwhGP6ycS0Aekp9ZYe_--zM_mw6k76ZFGElWQX-
ybtRDvc2VkpSDtfKmQrj0IRQLSGsNuDp-
AJV1u2ccmDSyDUMZwKbwbpAdGATGDiB3hH00bjbGehXTcApVud_wyDw",
  "Description": "Retrieve archive on 2015-07-17",
  "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-topic"
}
```

档案 ID 可在 `aws glacier upload-archive` 和的输出中找到 `aws glacier get-job-output`。

输出：

```
{
  "location": "/011685312445/vaults/mwunderl/jobs/17IL5-
EkXyEY9Ws95fClzIbk205uLYaFdAY0i-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJ059-8N9aHWav",
  "jobId": "17IL5-EkXy205uLYaFdAY0iEY9Ws95fClzIbk-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJ059-8N9aHWav"
}
```

有关任务参数格式的详细信息，请参阅 Amazon Glacier API 参考中的启动任务。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [InitiateJob](#) 中的。

## initiate-multipart-upload

以下代码示例演示了如何使用 `initiate-multipart-upload`。

### AWS CLI

以下命令启动分段上传到名为的文件库，每个文件的分段大小为 `my-vault` 1 MiB ( 1024 x 1024 字节 )：

```
aws glacier initiate-multipart-upload --account-id - --part-size 1048576 --vault-name my-vault --archive-description "multipart upload test"
```

档案描述参数是可选的。Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

成功后，此命令会输出上传 ID。使用上传档案的每个部分时，请使用上传 ID `aws glacier upload-multipart-part`。有关使用 CLI 分段上传到 Amazon Glacier 的更多信息，请参阅 AWS CL AWS I 用户指南中的使用亚马逊 Glacier。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [InitiateMultipartUpload](#) 中的。

## initiate-vault-lock

以下代码示例演示了如何使用 `initiate-vault-lock`。

### AWS CLI

#### 启动文件库锁定过程

以下 `initiate-vault-lock` 示例在指定的文件库上安装文件库锁定策略，并将文件库锁定的锁定状态设置为 `InProgress`。您必须 `complete-vault-lock` 在 24 小时内致电以将文件库锁定状态设置为，从而完成该过程 `Locked`。

```
aws glacier initiate-vault-lock \  
  --account-id - \  
  --vault-name MyVaultName \  
  --policy file://vault_lock_policy.json
```

`vault_lock_policy.json` 的内容：

```
{"Policy": "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Sid\": \"Define-vault-lock\", \"Effect\": \"Deny\", \"Principal\": {\"AWS\": \"arn:aws:iam::999999999999:root\"}, \"Action\": \"glacier:DeleteArchive\", \"Resource\": \"arn:aws:glacier:us-west-2:999999999999:vaults/examplevault\", \"Condition\": {\"NumericLessThanEquals\": {\"glacier:ArchiveAgeInDays\": \"365\"}}}]\"}
```

输出是可用于完成文件库锁定过程的文件库锁定 ID。

```
{
```

```
"lockId": "9QZgEXAMPLEPhvL6xEXAMPLE"
}
```

有关更多信息，请参阅 Amazon Glacier API 开发者[指南中的启动文件库锁定 \( POST 锁定策略 \)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[InitiateVaultLock](#)中的。

## list-jobs

以下代码示例演示了如何使用 list-jobs。

### AWS CLI

以下命令列出了名为 my-vault 的文件库的正在进行和最近完成的任务：

```
aws glacier list-jobs --account-id - --vault-name my-vault
```

输出：

```
{
  "JobList": [
    {
      "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-vault",
      "RetrievalByteRange": "0-3145727",
      "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault",
      "Completed": false,
      "SHA256TreeHash":
"9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67",
      "JobId": "17IL5-EkXyEY9Ws95fClzIbk205uLYaFdAY0i-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJ059-8N9aHWav",
      "ArchiveId": "kKB7ymWJVpPSwhGP6ycS0Aekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkpSDtFkmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKwbwAdGATGDiB3hH00bjbGehXTcApVud_wyDw",
      "JobDescription": "Retrieve archive on 2015-07-17",
      "ArchiveSizeInBytes": 3145728,
      "Action": "ArchiveRetrieval",
      "ArchiveSHA256TreeHash":
"9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67",
      "CreationDate": "2015-07-17T21:16:13.840Z",
      "StatusCode": "InProgress"
    },
  ],
}
```

```
{
  "InventoryRetrievalParameters": {
    "Format": "JSON"
  },
  "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-vault",
  "Completed": false,
  "JobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
R047Yc6FxsdBGgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61ds04CnMW",
  "Action": "InventoryRetrieval",
  "CreationDate": "2015-07-17T20:23:41.616Z",
  "StatusCode": ""InProgress""
}
]
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListJobs](#)中的。

## list-multipart-uploads

以下代码示例演示了如何使用 list-multipart-uploads。

### AWS CLI

以下命令显示名为的文件库的所有正在进行的分段上传：my-vault

```
aws glacier list-multipart-uploads --account-id - --vault-name my-vault
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

有关使用 CLI 分段上传到 Amazon Glacier 的更多信息，请参阅 AWS CL AWS I 用户指南中的使用亚马逊 Glacier。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListMultipartUploads](#)中的。

## list-parts

以下代码示例演示了如何使用 list-parts。

## AWS CLI

以下命令列出了分段上传到名为my-vault为的文件库的分段：

```
aws glacier list-parts --account-id - --vault-name my-vault --upload-id "SYZi7qnL-
YGqGwAm8Kn3BLP2E1NCvnB-5961R09CSaPmPwkYGH0qeN_nX3-Vhnd2yF0KfB5FkmbnBU9GubbdRcs8ut-D"
```

输出：

```
{
  "MultipartUploadId": "SYZi7qnL-
YGqGwAm8Kn3BLP2E1NCvnB-5961R09CSaPmPwkYGH0qeN_nX3-Vhnd2yF0KfB5FkmbnBU9GubbdRcs8ut-
D",
  "Parts": [
    {
      "RangeInBytes": "0-1048575",
      "SHA256TreeHash":
"e1f2a7cd6e047350f69b9f8cfa60fa606fe2f02802097a9a026360a7edc1f553"
    },
    {
      "RangeInBytes": "1048576-2097151",
      "SHA256TreeHash":
"43cf3061fb95796aed99a11a6aa3cd8f839eed15e655ab0a597126210636aee6"
    }
  ],
  "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-vault",
  "CreationDate": "2015-07-18T00:05:23.830Z",
  "PartSizeInBytes": 1048576
}
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

有关使用 CLI 分段上传到 Amazon Glacier 的更多信息，请参阅 AWS CLI 用户指南中的使用亚马逊 Glacier。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListParts](#) 中的。

### list-provisioned-capacity

以下代码示例演示了如何使用 list-provisioned-capacity。

## AWS CLI

### 检索已配置的容量单位

以下 `list-provisioned-capacity` 示例检索指定账户的所有预配置容量单位的详细信息。

```
aws glacier list-provisioned-capacity \  
  --account-id 111122223333
```

输出：

```
{  
  "ProvisionedCapacityList": [  
    {  
      "CapacityId": "HpASAUvfRFiVDb0jMfEIcr8K",  
      "ExpirationDate": "2020-03-18T19:59:24.000Z",  
      "StartDate": "2020-02-18T19:59:24.912Z"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListProvisionedCapacity](#) 中的。

### `list-tags-for-vault`

以下代码示例演示了如何使用 `list-tags-for-vault`。

## AWS CLI

以下命令列出应用于名为 `my-vault` 的文件库的标签：

```
aws glacier list-tags-for-vault --account-id - --vault-name my-vault
```

输出：

```
{  
  "Tags": {  
    "date": "july2015",  
    "id": "1234"  
  }  
}
```

```
}
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForVault](#)中的。

## list-vaults

以下代码示例演示了如何使用 list-vaults。

### AWS CLI

以下命令列出默认账户和区域中的文件库：

```
aws glacier list-vaults --account-id -
```

输出：

```
{
  "VaultList": [
    {
      "SizeInBytes": 3178496,
      "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-vault",
      "LastInventoryDate": "2015-04-07T00:26:19.028Z",
      "VaultName": "my-vault",
      "NumberOfArchives": 1,
      "CreationDate": "2015-04-06T21:23:45.708Z"
    }
  ]
}
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListVaults](#)中的。

## purchase-provisioned-capacity

以下代码示例演示了如何使用 purchase-provisioned-capacity。



## AWS CLI

### 购买预置容量单位

以下purchase-provisioned-capacity示例购买了预配置容量单位。

```
aws glacier purchase-provisioned-capacity \  
  --account-id 111122223333
```

输出：

```
{  
  "capacityId": "HpASAUvfRFiVDb0jMfEIcr8K"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PurchaseProvisionedCapacity](#)中的。

### remove-tags-from-vault

以下代码示例演示了如何使用 remove-tags-from-vault。

## AWS CLI

以下命令date从名为的文件库中删除带有密钥的标签my-vault：

```
aws glacier remove-tags-from-vault --account-id - --vault-name my-vault --tag-keys  
date
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveTagsFromVault](#)中的。

### set-data-retrieval-policy

以下代码示例演示了如何使用 set-data-retrieval-policy。

## AWS CLI

以下命令为正在使用的账户配置数据检索策略：

```
aws glacier set-data-retrieval-policy --account-id - --policy file://data-retrieval-policy.json
```

`data-retrieval-policy.json`是当前文件夹中的一个 JSON 文件，用于指定数据检索策略：

```
{
  "Rules": [
    {
      "Strategy": "BytesPerHour",
      "BytesPerHour": 10737418240
    }
  ]
}
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

以下命令将数据检索策略设置为FreeTier使用内联 JSON：

```
aws glacier set-data-retrieval-policy --account-id - --policy '{"Rules": [{"Strategy": "FreeTier"}]}'
```

有关策略格式的详细信息，请参阅 Amazon Glacier API 参考中的设置数据检索策略。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetDataRetrievalPolicy](#)中的。

## set-vault-access-policy

以下代码示例演示了如何使用 `set-vault-access-policy`。

### AWS CLI

#### 设置文件库的访问策略

以下`set-vault-access-policy`示例将权限策略附加到指定的文件库。

```
aws glacier set-vault-access-policy \
  --account-id 111122223333 \
  --vault-name example_vault
  --policy '{"Policy": "{\"Version\": \"2012-10-17\", \"Statement\": [{"Effect\": \"Allow\", \"Principal\": {\"AWS\": \"arn:aws:iam::444455556666:root
```

```
\"}, {"Action": "glacier:ListJobs"}, {"Resource": "arn:aws:glacier:us-east-1:111122223333:vaults/example_vault"}], [{"Effect": "Allow"}, {"Principal": {"AWS": "arn:aws:iam::444455556666:root"}}, {"Action": "glacier:UploadArchive"}, {"Resource": "arn:aws:glacier:us-east-1:111122223333:vaults/example_vault"}]]}'
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SetVaultAccessPolicy](#) 中的。

## set-vault-notifications

以下代码示例演示了如何使用 set-vault-notifications。

### AWS CLI

以下命令为名为 my-vault 的文件库配置 SNS 通知：

```
aws glacier set-vault-notifications --account-id - --vault-name my-vault --vault-notification-config file://notificationconfig.json
```

notificationconfig.json 是当前文件夹中的一个 JSON 文件，用于指定 SNS 主题和要发布的事件：

```
{
  "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault",
  "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"]
}
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SetVaultNotifications](#) 中的。

## upload-archive

以下代码示例演示了如何使用 upload-archive。

### AWS CLI

以下命令将名为 archive.zip 的当前文件夹中的存档上传到名为 my-vault 的文件库：

```
aws glacier upload-archive --account-id - --vault-name my-vault --body archive.zip
```

输出：

```
{
  "archiveId": "kKB7ymWJVpPSwhGP6ycS0Aekp9ZYe_--zM_mw6k76ZFGIEWQX-
ybtRDvc2VkJPSDtfKmQrj0IRQLSGsNuDp-
AJV1u2ccmDSyDUmZwKbwbpAdGATGDiB3hH00bjbGehXTcApVud_wyDw",
  "checksum": "969fb39823836d81f0cc028195fcdbcbbe76cdde932d4646fa7de5f21e18aa67",
  "location": "/0123456789012/vaults/my-vault/archives/
kKB7ymWJVpPSwhGP6ycS0Aekp9ZYe_--zM_mw6k76ZFGIEWQX-ybtRDvc2VkJPSDtfKmQrj0IRQLSGsNuDp-
AJV1u2ccmDSyDUmZwKbwbpAdGATGDiB3hH00bjbGehXTcApVud_wyDw"
}
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

要检索上传的存档，可使用 `aws glacier initiate-job` 命令启动检索任务。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UploadArchive](#) 中的。

## upload-multipart-part

以下代码示例演示了如何使用 `upload-multipart-part`。

### AWS CLI

以下命令上传存档的前 1 MiB ( 1024 x 1024 字节 ) 部分：

```
aws glacier upload-multipart-part --body part1 --range 'bytes
0-1048575/*' --account-id - --vault-name my-vault --upload-
id 19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-
0ssZtLqyFu7sY1_lR7vgFuJV6NtcV5zpsJ
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

正文参数采用本地文件系统上分段文件的路径。范围参数采用 HTTP 内容范围，指示该分段在已完成的存档中占用的字节。上传 ID 由 `aws glacier initiate-multipart-upload` 命令返回，也可以使用 `aws glacier list-multipart-uploads` 获取它。

有关使用 CLI 分段上传到 Amazon Glacier 的更多信息，请参阅 AWS CL AWS I 用户指南中的使用亚马逊 Glacier。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UploadMultipartPart](#)中的。

## 使用的 Secrets Manager 示例 AWS CLI

以下代码示例向您展示了如何使用 with Secrets Manager 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### batch-get-secret-value

以下代码示例演示了如何使用 batch-get-secret-value。

AWS CLI

示例 1：检索按名称列出的一组密钥的密钥值

以下batch-get-secret-value示例获取了三个密钥的秘密值机密。

```
aws secretsmanager batch-get-secret-value \  
  --secret-id-list MySecret1 MySecret2 MySecret3
```

输出：

```
{  
  "SecretValues": [  
    {
```

```

    "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MySecret1-
a1b2c3",
    "Name": "MySecret1",
    "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "SecretString": "{\"username\":\"diego_ramirez\",\"password\":\"EXAMPLE-
PASSWORD\",\"engine\":\"mysql\",\"host\":\"secretsmanagertutorial.cluster.us-
west-2.rds.amazonaws.com\",\"port\":3306,\"dbClusterIdentifier\":
\"secretsmanagertutorial\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreateDate": "1523477145.729"
  },
  {
    "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MySecret2-
a1b2c3",
    "Name": "MySecret2",
    "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbbb",
    "SecretString": "{\"username\":\"akua_mansa\",\"password\":\"EXAMPLE-
PASSWORD\""}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreateDate": "1673477781.275"
  },
  {
    "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MySecret3-
a1b2c3",
    "Name": "MySecret3",
    "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEccccc",
    "SecretString": "{\"username\":\"jie_liu\",\"password\":\"EXAMPLE-
PASSWORD\""}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreateDate": "1373477721.124"
  }
],
"Errors": []
}

```

有关更多信息，请参阅 [S ecrets Manager 用户指南中的批量检索一组AWS密钥](#)。

示例 2：检索筛选器选择的一组密钥的密钥值

以下batch-get-secret-value示例获取您账户中名称中包含的机密值机密。MySecret按名称筛选区分大小写。

```
aws secretsmanager batch-get-secret-value \  
  --filters Key="name",Values="MySecret"
```

输出：

```
{  
  "SecretValues": [  
    {  
      "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MySecret1-  
a1b2c3",  
      "Name": "MySecret1",  
      "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",  
      "SecretString": "{\"username\":\"diego_ramirez\",\"password\":\"EXAMPLE-  
PASSWORD\",\"engine\":\"mysql\",\"host\":\"secretsmanagertutorial.cluster.us-  
west-2.rds.amazonaws.com\",\"port\":3306,\"dbClusterIdentifier\":  
\"secretsmanagertutorial\"}",  
      "VersionStages": [  
        "AWSCURRENT"  
      ],  
      "CreateDate": "1523477145.729"  
    },  
    {  
      "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MySecret2-  
a1b2c3",  
      "Name": "MySecret2",  
      "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbbb",  
      "SecretString": "{\"username\":\"akua_mansa\",\"password\":\"EXAMPLE-  
PASSWORD\""}",  
      "VersionStages": [  
        "AWSCURRENT"  
      ],  
      "CreateDate": "1673477781.275"  
    },  
    {  
      "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MySecret3-  
a1b2c3",  
      "Name": "MySecret3",  
      "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEccccc",  
      "SecretString": "{\"username\":\"jie_liu\",\"password\":\"EXAMPLE-  
PASSWORD\""}",  
    }  
  ]  
}
```

```
        "VersionStages": [  
            "AWSCURRENT"  
        ],  
        "CreateDate": "1373477721.124"  
    }  
],  
"Errors": []  
}
```

有关更多信息，请参阅 [Secrets Manager 用户指南中的批量检索一组AWS密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetSecretValue](#)中的。

## cancel-rotate-secret

以下代码示例演示了如何使用 `cancel-rotate-secret`。

### AWS CLI

关闭密钥的自动轮换

以下`cancel-rotate-secret`示例关闭了密钥的自动轮换。要恢复轮换，请致电`rotate-secret`。

```
aws secretsmanager cancel-rotate-secret \  
    --secret-id MyTestSecret
```

输出：

```
{  
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-a1b2c3",  
  "Name": "MyTestSecret"  
}
```

有关更多信息，请参阅 [Secrets Manager 用户指南中的轮换密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelRotateSecret](#)中的。

## create-secret

以下代码示例演示了如何使用 `create-secret`。



## AWS CLI

### 示例 1：创建密钥

以下 `create-secret` 示例将创建包含两个键值对的密钥。

```
aws secretsmanager create-secret \  
  --name MyTestSecret \  
  --description "My test secret created with the CLI." \  
  --secret-string "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}"
```

输出：

```
{  
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-a1b2c3",  
  "Name": "MyTestSecret",  
  "VersionId": "EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE"  
}
```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[创建密钥](#)。

### 示例 2：根据 JSON 文件中的凭证创建密钥

以下 `create-secret` 示例将根据文件中的凭证创建密钥。有关更多信息，请参阅 [AWS CLI 用户指南中的从文件加载 AWS CLI 参数](#)。

```
aws secretsmanager create-secret \  
  --name MyTestSecret \  
  --secret-string file://mycreds.json
```

`mycreds.json` 的内容：

```
{  
  "engine": "mysql",  
  "username": "saanvis",  
  "password": "EXAMPLE-PASSWORD",  
  "host": "my-database-endpoint.us-west-2.rds.amazonaws.com",  
  "dbname": "myDatabase",  
  "port": "3306"  
}
```

输出：

```
{
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-a1b2c3",
  "Name": "MyTestSecret",
  "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[创建密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSecret](#)中的。

## delete-resource-policy

以下代码示例演示了如何使用 delete-resource-policy。

AWS CLI

删除附加到密钥的基于资源的策略

以下 delete-resource-policy 示例将删除附加到密钥的基于资源的策略。

```
aws secretsmanager delete-resource-policy \
  --secret-id MyTestSecret
```

输出：

```
{
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-a1b2c3",
  "Name": "MyTestSecret"
}
```

有关更多信息，请参阅 S ecrets Manager 用户指南中的[身份验证和访问控制](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteResourcePolicy](#)中的。

## delete-secret

以下代码示例演示了如何使用 delete-secret。

## AWS CLI

### 示例 1：删除密钥

以下 `delete-secret` 示例将删除密钥。您可以使用 `DeletionDate` 恢复密钥，直到 `restore-secret` 响应字段中的日期和时间。要删除复制到其他区域的密钥，请先使用 `remove-regions-from-replication` 删除其副本，然后调用 `delete-secret`。

```
aws secretsmanager delete-secret \  
  --secret-id MyTestSecret \  
  --recovery-window-in-days 7
```

输出：

```
{  
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-  
a1b2c3",  
  "Name": "MyTestSecret",  
  "DeletionDate": 1524085349.095  
}
```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[删除密钥](#)。

### 示例 2：立即删除密钥

以下 `delete-secret` 示例将立即删除密钥而没有恢复时段。您无法恢复此密钥。

```
aws secretsmanager delete-secret \  
  --secret-id MyTestSecret \  
  --force-delete-without-recovery
```

输出：

```
{  
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-  
a1b2c3",  
  "Name": "MyTestSecret",  
  "DeletionDate": 1508750180.309  
}
```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[删除密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSecret](#)中的。

## describe-secret

以下代码示例演示了如何使用 describe-secret。

### AWS CLI

检索密钥的详细信息

以下 describe-secret 示例显示密钥的详细信息。

```
aws secretsmanager describe-secret \  
  --secret-id MyTestSecret
```

输出：

```
{  
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-  
Ca8JGt",  
  "Name": "MyTestSecret",  
  "Description": "My test secret",  
  "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/EXAMPLE1-90ab-cdef-fedc-  
ba987EXAMPLE",  
  "RotationEnabled": true,  
  "RotationLambdaARN": "arn:aws:lambda:us-  
west-2:123456789012:function:MyTestRotationLambda",  
  "RotationRules": {  
    "AutomaticallyAfterDays": 2,  
    "Duration": "2h",  
    "ScheduleExpression": "cron(0 16 1,15 * ? *)"  
  },  
  "LastRotatedDate": 1525747253.72,  
  "LastChangedDate": 1523477145.729,  
  "LastAccessedDate": 1524572133.25,  
  "Tags": [  
    {  
      "Key": "SecondTag",  
      "Value": "AnotherValue"  
    },  
    {  
      "Key": "FirstTag",
```

```
        "Value": "SomeValue"
      }
    ],
    "VersionIdsToStages": {
      "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111": [
        "AWSPREVIOUS"
      ],
      "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222": [
        "AWSCURRENT"
      ],
      "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333": [
        "AWSPENDING"
      ]
    },
    "CreateDate": 1521534252.66,
    "PrimaryRegion": "us-west-2",
    "ReplicationStatus": [
      {
        "Region": "eu-west-3",
        "KmsKeyId": "alias/aws/secretsmanager",
        "Status": "InSync",
        "StatusMessage": "Replication succeeded"
      }
    ]
  }
}
```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSecret](#)中的。

## get-random-password

以下代码示例演示了如何使用 `get-random-password`。

### AWS CLI

#### 生成随机密码

以下`get-random-password`示例生成一个长度为 20 个字符的随机密码，其中至少包含一个大写字母、小写字母、数字和标点符号。

```
aws secretsmanager get-random-password \
  --require-each-included-type \
```

```
--password-length 20
```

输出：

```
{
  "RandomPassword": "EXAMPLE-PASSWORD"
}
```

有关更多信息，请参阅 [Secrets Manager 用户指南中的创建和管理密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetRandomPassword](#) 中的。

## get-resource-policy

以下代码示例演示了如何使用 `get-resource-policy`。

### AWS CLI

检索附加到密钥的基于资源的策略

以下 `get-resource-policy` 示例将检索附加到密钥的基于资源的策略。

```
aws secretsmanager get-resource-policy \
  --secret-id MyTestSecret
```

输出：

```
{
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-a1b2c3",
  "Name": "MyTestSecret",
  "ResourcePolicy": "{\n  \"Version\": \"2012-10-17\",\n  \"Statement\": [\n    {\n      \"Effect\": \"Allow\",\n      \"Principal\": {\n        \"AWS\": \"arn:aws:iam::123456789012:root\"\n      },\n      \"Action\": \"secretsmanager:GetSecretValue\",\n      \"Resource\": \"*\"\n    }\n  ]\n}"
```

有关更多信息，请参阅 [Secrets Manager 用户指南中的身份验证和访问控制](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetResourcePolicy](#) 中的。

## get-secret-value

以下代码示例演示了如何使用 `get-secret-value`。

### AWS CLI

#### 示例 1：检索密钥的加密密钥值

以下 `get-secret-value` 示例获取当前密钥值。

```
aws secretsmanager get-secret-value \  
  --secret-id MyTestSecret
```

输出：

```
{  
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-  
a1b2c3",  
  "Name": "MyTestSecret",  
  "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
  "SecretString": "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}",  
  "VersionStages": [  
    "AWSCURRENT"  
  ],  
  "CreateDate": 1523477145.713  
}
```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[检索密钥](#)。

#### 示例 2：检索之前的密钥值

以下 `get-secret-value` 示例获取之前的密钥值。

```
aws secretsmanager get-secret-value \  
  --secret-id MyTestSecret  
  --version-stage AWSPREVIOUS
```

输出：

```
{  
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-  
a1b2c3",
```

```
"Name": "MyTestSecret",
"VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
"SecretString": "{\"user\":\"diegor\",\"password\":\"PREVIOUS-EXAMPLE-PASSWORD
\"}",
"VersionStages": [
  "AWSPREVIOUS"
],
"CreateDate": 1523477145.713
}
```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[检索密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSecretValue](#)中的。

## list-secret-version-ids

以下代码示例演示了如何使用 list-secret-version-ids。

### AWS CLI

列出与密钥关联的所有密钥版本

以下list-secret-version-ids示例获取密钥所有版本的列表。

```
aws secretsmanager list-secret-version-ids \
  --secret-id MyTestSecret
```

输出：

```
{
  "Versions": [
    {
      "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "VersionStages": [
        "AWSPREVIOUS"
      ],
      "LastAccessedDate": 1523477145.713,
      "CreateDate": 1523477145.713
    },
    {
      "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "VersionStages": [
```



```

    "AWSCURRENT"
  ],
  "LastAccessedDate": 1523477145.713,
  "CreatedDate": 1523486221.391
},
{
  "CreatedDate": 1.51197446236E9,
  "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333;"
}
],
"ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-
a1b2c3",
  "Name": "MyTestSecret"
}

```

有关更多信息，请参阅 S ecrets Manager 用户指南中的[版本](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSecretVersionIds](#)中的。

## list-secrets

以下代码示例演示了如何使用 list-secrets。

### AWS CLI

示例 1：列出您账户中的密钥

以下 list-secrets 示例获取了您账户中的密钥列表。

```
aws secretsmanager list-secrets
```

输出：

```

{
  "SecretList": [
    {
      "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MyTestSecret-a1b2c3",
      "Name": "MyTestSecret",
      "LastChangedDate": 1523477145.729,
      "SecretVersionsToStages": {
        "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111": [

```

```

        "AWSCURRENT"
      ]
    }
  },
  {
    "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:AnotherSecret-d4e5f6",
    "Name": "AnotherSecret",
    "LastChangedDate": 1523482025.685,
    "SecretVersionsToStages": {
      "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222": [
        "AWSCURRENT"
      ]
    }
  }
]
}

```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[查找密钥](#)。

#### 示例 2：筛选您账户中的密钥列表

以下 `list-secrets` 示例将获取您的账户中名称包含 `Test` 的密钥列表。按名称筛选区分大小写。

```

aws secretsmanager list-secrets \
  --filter Key="name",Values="Test"

```

输出：

```

{
  "SecretList": [
    {
      "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MyTestSecret-a1b2c3",
      "Name": "MyTestSecret",
      "LastChangedDate": 1523477145.729,
      "SecretVersionsToStages": {
        "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111": [
          "AWSCURRENT"
        ]
      }
    }
  ]
}

```

```
]
}
```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[查找密钥](#)。

示例 3：列出您账户中由其他服务管理的密钥

以下 `list-secrets` 示例返回您账户中由 Amazon RDS 管理的密钥。

```
aws secretsmanager list-secrets \
  --filter Key="owning-service",Values="rds"
```

输出：

```
{
  "SecretList": [
    {
      "Name": "rds!cluster-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "Tags": [
        {
          "Value": "arn:aws:rds:us-
west-2:123456789012:cluster:database-1",
          "Key": "aws:rds:primaryDBClusterArn"
        },
        {
          "Value": "rds",
          "Key": "aws:secretsmanager:owningService"
        }
      ],
      "RotationRules": {
        "AutomaticallyAfterDays": 1
      },
      "LastChangedDate": 1673477781.275,
      "LastRotatedDate": 1673477781.26,
      "SecretVersionsToStages": {
        "a1b2c3d4-5678-90ab-cdef-EXAMPLEEaaaaa": [
          "AWSPREVIOUS"
        ],
        "a1b2c3d4-5678-90ab-cdef-EXAMPLEEbbbbbb": [
          "AWSCURRENT",
          "AWSPENDING"
        ]
      }
    }
  ],
}
```

```

        "OwningService": "rds",
        "RotationEnabled": true,
        "CreatedDate": 1673467300.7,
        "LastAccessedDate": 1673395200.0,
        "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:rds!
cluster-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111-a1b2c3",
        "Description": "Secret associated with primary RDS DB cluster:
arn:aws:rds:us-west-2:123456789012:cluster:database-1"
    }
]
}

```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[由其他服务管理的密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListSecrets](#) 中的。

## put-resource-policy

以下代码示例演示了如何使用 `put-resource-policy`。

### AWS CLI

向密钥添加基于资源的策略

以下 `put-resource-policy` 示例将向密钥添加权限策略，首先检查该策略是否不提供对该密钥的广泛访问权限。该策略是从文件中读取的。有关更多信息，请参阅 [AWS CLI 用户指南中的从文件加载 AWS CLI 参数](#)。

```

aws secretsmanager put-resource-policy \
  --secret-id MyTestSecret \
  --resource-policy file://mypolicy.json \
  --block-public-policy

```

`mypolicy.json` 的内容：

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/MyRole"
      }
    }
  ]
}

```

```

        },
        "Action": "secretsmanager:GetSecretValue",
        "Resource": "*"
    }
]
}

```

输出：

```

{
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-
a1b2c3",
  "Name": "MyTestSecret"
}

```

有关更多信息，请参阅 [Secrets Manager 用户指南中的为密钥附加权限策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutResourcePolicy](#) 中的。

## put-secret-value

以下代码示例演示了如何使用 `put-secret-value`。

### AWS CLI

示例 1：在密钥中存储新的密钥值

以下 `put-secret-value` 示例将创建包含两个键值对的新版本密钥。

```

aws secretsmanager put-secret-value \
  --secret-id MyTestSecret \
  --secret-string "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}"

```

输出：

```

{
  "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MyTestSecret-1a2b3c",
  "Name": "MyTestSecret",
  "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "VersionStages": [

```

```
    "AWSCURRENT"  
  ]  
}
```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[修改密钥](#)。

示例 2：将凭证中的新密钥值存储在 JSON 文件中

以下 `put-secret-value` 示例将根据文件中的凭证创建新版本密钥。有关更多信息，请参阅[AWS CLI 用户指南中的从文件加载 AWS CLI 参数](#)。

```
aws secretsmanager put-secret-value \  
  --secret-id MyTestSecret \  
  --secret-string file://mycreds.json
```

`mycreds.json` 的内容：

```
{  
  "engine": "mysql",  
  "username": "saanvis",  
  "password": "EXAMPLE-PASSWORD",  
  "host": "my-database-endpoint.us-west-2.rds.amazonaws.com",  
  "dbname": "myDatabase",  
  "port": "3306"  
}
```

输出：

```
{  
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-a1b2c3",  
  "Name": "MyTestSecret",  
  "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
  "VersionStages": [  
    "AWSCURRENT"  
  ]  
}
```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[修改密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutSecretValue](#) 中的。

## remove-regions-from-replication

以下代码示例演示了如何使用 `remove-regions-from-replication`。

### AWS CLI

#### 删除副本密钥

以下 `remove-regions-from-replication` 示例将删除 `eu-west-3` 中的副本密钥。要删除复制到其他区域的主密钥，请先删除副本，然后调用 `delete-secret`。

```
aws secretsmanager remove-regions-from-replication \  
  --secret-id MyTestSecret \  
  --remove-replica-regions eu-west-3
```

#### 输出：

```
{  
  "ARN": "arn:aws:secretsmanager:us-  
west-2:123456789012:secret:MyTestSecret-1a2b3c",  
  "ReplicationStatus": []  
}
```

有关更多信息，请参阅 [Secrets Manager 用户指南中的删除副本密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RemoveRegionsFromReplication](#) 中的。

## replicate-secret-to-regions

以下代码示例演示了如何使用 `replicate-secret-to-regions`。

### AWS CLI

#### 将密钥复制到另一个区域

以下 `replicate-secret-to-regions` 示例将密钥复制到 `eu-west-3`。副本使用 AWS 托管密钥加密 `aws/secretsmanager`。

```
aws secretsmanager replicate-secret-to-regions \  
  --secret-id MyTestSecret \  
  --add-replica-regions Region=eu-west-3
```

输出：

```
{
  "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MyTestSecret-1a2b3c",
  "ReplicationStatus": [
    {
      "Region": "eu-west-3",
      "KmsKeyId": "alias/aws/secretsmanager",
      "Status": "InProgress"
    }
  ]
}
```

有关更多信息，请参阅 [S ecrets Manager 用户指南中的将密钥复制到其他区域](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReplicateSecretToRegions](#)中的。

## restore-secret

以下代码示例演示了如何使用 restore-secret。

### AWS CLI

恢复之前删除的密钥

以下 restore-secret 示例恢复了先前计划删除的密钥。

```
aws secretsmanager restore-secret \
  --secret-id MyTestSecret
```

输出：

```
{
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-
a1b2c3",
  "Name": "MyTestSecret"
}
```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[删除密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RestoreSecret](#)中的。



## rotate-secret

以下代码示例演示了如何使用 rotate-secret。

### AWS CLI

#### 示例 1：配置并启动密钥的自动轮换

以下 rotate-secret 示例配置并启动密钥的自动轮换。Secrets Manager 会立即轮换一次密钥，然后在两小时内每八小时轮换一次密钥。输出显示 VersionId 了通过轮换创建的新密钥版本的。

```
aws secretsmanager rotate-secret \  
  --secret-id MyTestDatabaseSecret \  
  --rotation-lambda-arn arn:aws:lambda:us-  
west-2:1234566789012:function:SecretsManagerTestRotationLambda \  
  --rotation-rules "{\"ScheduleExpression\": \"cron(0 8/8 * * ? *)\"}, {\"Duration\  
\": \"2h\"}"
```

输出：

```
{  
  "ARN": "aws:arn:secretsmanager:us-  
west-2:123456789012:secret:MyTestDatabaseSecret-a1b2c3",  
  "Name": "MyTestDatabaseSecret",  
  "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE111111"  
}
```

有关更多信息，请参阅 [Secrets Manager 用户指南中的轮换密钥](#)。

#### 示例 2：配置并开始按轮换间隔自动旋转

以下 rotate-secret 示例配置并启动密钥的自动轮换。Secrets Manager 会立即轮换一次密钥，然后每 10 天轮换一次。输出显示 VersionId 了通过轮换创建的新密钥版本的。

```
aws secretsmanager rotate-secret \  
  --secret-id MyTestDatabaseSecret \  
  --rotation-lambda-arn arn:aws:lambda:us-  
west-2:1234566789012:function:SecretsManagerTestRotationLambda \  
  --rotation-rules "{\"ScheduleExpression\": \"rate(10 days)\"}"
```

输出：

```
{
  "ARN": "aws:arn:secretsmanager:us-
west-2:123456789012:secret:MyTestDatabaseSecret-a1b2c3",
  "Name": "MyTestDatabaseSecret",
  "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

有关更多信息，请参阅 [Secrets Manager 用户指南中的轮换密钥](#)。

### 示例 3：立即轮换密钥

以下 `rotate-secret` 示例将立即开始轮换。输出显示 `VersionId` 了通过轮换创建的新密钥版本的。密钥必须已配置轮换。

```
aws secretsmanager rotate-secret \
  --secret-id MyTestDatabaseSecret
```

输出：

```
{
  "ARN": "aws:arn:secretsmanager:us-
west-2:123456789012:secret:MyTestDatabaseSecret-a1b2c3",
  "Name": "MyTestDatabaseSecret",
  "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

有关更多信息，请参阅 [Secrets Manager 用户指南中的轮换密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RotateSecret](#) 中的。

## stop-replication-to-replica

以下代码示例演示了如何使用 `stop-replication-to-replica`。

### AWS CLI

将副本密钥提升为主密钥

以下 `stop-replication-to-replica` 示例将删除副本密钥与主密钥之间的链接。副本密钥在副本区域中被提升为主密钥。您必须从副本区域内调用 `stop-replication-to-replica`。

```
aws secretsmanager stop-replication-to-replica \
```

```
--secret-id MyTestSecret
```

输出：

```
{
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-
a1b2c3"
}
```

有关更多信息，请参阅 [Secrets Manager 用户指南中的提升副本密钥](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StopReplicationToReplica](#) 中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

示例 1：为密钥添加标签

以下示例说明了如何使用速记语法附加标签。

```
aws secretsmanager tag-resource \
  --secret-id MyTestSecret \
  --tags Key=FirstTag,Value=FirstValue
```

此命令不生成任何输出。

有关更多信息，请参阅 Secrets Manager 用户指南中的 [标记您的密钥](#)。

示例 2：为密钥添加多个标签

以下 tag-resource 示例将向密钥附加两个键值标签。

```
aws secretsmanager tag-resource \
  --secret-id MyTestSecret \
  --tags '[{"Key": "FirstTag", "Value": "FirstValue"}, {"Key": "SecondTag",
"Value": "SecondValue"}]'
```

此命令不生成任何输出。

有关更多信息，请参阅 [Secrets Manager 用户指南中的标签](#) 密钥。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从密钥中移除标签

以下 untag-resource 示例将从密钥中删除两个标签。对于每个标签，键和值都会被删除。

```
aws secretsmanager untag-resource \  
  --secret-id MyTestSecret \  
  --tag-keys '[ "FirstTag", "SecondTag"]'
```

此命令不生成任何输出。

有关更多信息，请参阅 [Secrets Manager 用户指南中的标签](#) 密钥。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-secret-version-stage

以下代码示例演示了如何使用 update-secret-version-stage。

### AWS CLI

示例 1：将密钥还原为先前版本

以下 update-secret-version-stage 示例将 CURRENT AWS 暂存标签移动到密钥的先前版本，这会将密钥还原为先前的版本。要查找先前版本的 ID，请使用 list-secret-version-ids。在本示例中，带有 AWS 当前标签的版本是 a1b2c3d4-5678-90ab-CDEF-example11111，带有先前标签的版本是 a1b2c3d4-5678-90ab-cdef-example22222。AWS 在本示例中，您将 AWS 当前标签从版本 11111 移动到 22222。由于 AWS 当前标签已从版本中移除，因此 update-secret-version-stage 会自动将 AWS 之前的标签移至该版本 (11111)。结果是交换了 AWS 当前版本和 AWS 以前的版本。

```
aws secretsmanager update-secret-version-stage \  
  --secret-id MyTestSecret \  
  --previous-version-id a1b2c3d4-5678-90ab-cdef-example22222 \  
  --current-version-id a1b2c3d4-5678-90ab-CDEF-example11111
```

```
--version-stage AWSCURRENT \  
--move-to-version-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222 \  
--remove-from-version-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-  
a1b2c3",  
  "Name": "MyTestSecret"  
}
```

有关更多信息，请参阅 S ecrets Manager 用户指南中的[版本](#)。

示例 2：添加附加到密钥版本的暂存标签

以下update-secret-version-stage示例为密钥的某个版本添加暂存标签。您可以通过运行list-secret-version-ids和查看受影响版本的VersionStages响应字段来查看结果。

```
aws secretsmanager update-secret-version-stage \  
  --secret-id MyTestSecret \  
  --version-stage STAGINGLABEL1 \  
  --move-to-version-id EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE
```

输出：

```
{  
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-  
a1b2c3",  
  "Name": "MyTestSecret"  
}
```

有关更多信息，请参阅 S ecrets Manager 用户指南中的[版本](#)。

示例 3：删除附加到密钥版本的暂存标签

以下update-secret-version-stage示例删除了附加到密钥版本的暂存标签。您可以通过运行list-secret-version-ids和查看受影响版本的VersionStages响应字段来查看结果。

```
aws secretsmanager update-secret-version-stage \  
  --secret-id MyTestSecret \  
  --version-stage STAGINGLABEL1 \  
  --remove-from-version-id EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE
```

```
--remove-from-version-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-
a1b2c3",
  "Name": "MyTestSecret"
}
```

有关更多信息，请参阅 [S ecrets Manager 用户指南中的版本](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 UpdateSecretVersionStage](#) 中的。

## update-secret

以下代码示例演示了如何使用 `update-secret`。

### AWS CLI

示例 1：更新密钥的描述

以下 `update-secret` 示例将更新密钥的描述。

```
aws secretsmanager update-secret \
  --secret-id MyTestSecret \
  --description "This is a new description for the secret."
```

输出：

```
{
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-
a1b2c3",
  "Name": "MyTestSecret"
}
```

有关更多信息，请参阅《[Secrets Manager 用户指南](#)》中的 [修改密钥](#)。

示例 2：更新与密钥关联的加密密钥

以下 `update-secret` 示例将更新用于加密密钥值的 KMS 密钥。该 KMS 密钥必须与加密密钥位于同一区域中。

```
aws secretsmanager update-secret \  
  --secret-id MyTestSecret \  
  --kms-key-id arn:aws:kms:us-west-2:123456789012:key/EXAMPLE1-90ab-cdef-fedc-  
ba987EXAMPLE
```

输出：

```
{  
  "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-  
a1b2c3",  
  "Name": "MyTestSecret"  
}
```

有关更多信息，请参阅《Secrets Manager 用户指南》中的[修改密钥](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateSecret](#)中的。

## validate-resource-policy

以下代码示例演示了如何使用 validate-resource-policy。

### AWS CLI

#### 验证资源策略

以下validate-resource-policy示例检查资源策略是否未授予对密钥的广泛访问权限。策略是从磁盘上的文件中读取的。有关更多信息，请参阅[AWS CLI 用户指南中的从文件加载AWS CLI 参数](#)。

```
aws secretsmanager validate-resource-policy \  
  --resource-policy file://mypolicy.json
```

mypolicy.json 的内容：

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {
```

```
        "AWS": "arn:aws:iam::123456789012:role/MyRole"
      },
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "*"
    }
  ]
}
```

输出：

```
{
  "PolicyValidationPassed": true,
  "ValidationErrors": []
}
```

有关更多信息，请参阅 [Secrets Manager 用户指南中的 Secrets Manager 权限参考](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 ValidateResourcePolicy](#) 中的。

## 使用的 Security Hub 示例 AWS CLI

以下代码示例向您展示了如何使用 `awscli` 与 Security Hub 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 [GitHub](#)，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **accept-administrator-invitation**

以下代码示例演示了如何使用 `accept-administrator-invitation`。



## AWS CLI

### 接受管理员账户的邀请

以下accept-administrator-invitation示例接受来自指定管理员账户的指定邀请。

```
aws securityhub accept-invitation \  
  --administrator-id 123456789012 \  
  --invitation-id 7ab938c5d52d7904ad09f9e7c20cc4eb
```

此命令不生成任何输出。

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AcceptAdministratorInvitation](#)中的。

## accept-invitation

以下代码示例演示了如何使用 accept-invitation。

## AWS CLI

### 接受管理员账户的邀请

以下accept-invitation示例接受来自指定管理员账户的指定邀请。

```
aws securityhub accept-invitation \  
  --master-id 123456789012 \  
  --invitation-id 7ab938c5d52d7904ad09f9e7c20cc4eb
```

此命令不生成任何输出。

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AcceptInvitation](#)中的。

## batch-delete-automation-rules

以下代码示例演示了如何使用 batch-delete-automation-rules。

## AWS CLI

### 删除自动化规则

以下 `batch-delete-automation-rules` 示例删除了指定的自动化规则。您可以使用单个命令删除一条或多条规则。只有 Security Hub 管理员帐户可以运行此命令。

```
aws securityhub batch-delete-automation-rules \  
  --automation-rules-arns '["arn:aws:securityhub:us-  
east-1:123456789012:automation-rule/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"]'
```

输出：

```
{  
  "ProcessedAutomationRules": [  
    "arn:aws:securityhub:us-east-1:123456789012:automation-rule/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"  
  ],  
  "UnprocessedAutomationRules": []  
}
```

有关更多信息，请参阅《Sec AWS urity Hub 用户指南》中的[删除自动化规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[BatchDeleteAutomationRules](#)中的。

## batch-disable-standards

以下代码示例演示了如何使用 `batch-disable-standards`。

### AWS CLI

#### 禁用标准

以下 `batch-disable-standards` 示例禁用与指定订阅 ARN 关联的标准。

```
aws securityhub batch-disable-standards \  
  --standards-subscription-arns "arn:aws:securityhub:us-  
west-1:123456789012:subscription/pci-dss/v/3.2.1"
```

输出：

```
{  
  "StandardsSubscriptions": [  
    {  
      "StandardsArn": "arn:aws:securityhub:eu-central-1::standards/pci-dss/  
v/3.2.1",
```

```

        "StandardsInput": { },
        "StandardsStatus": "DELETING",
        "StandardsSubscriptionArn": "arn:aws:securityhub:us-
west-1:123456789012:subscription/pci-dss/v/3.2.1"
    }
]
}

```

有关更多信息，请参阅《Security Hub 用户指南》中的禁用或启用AWS 安全[标准](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchDisableStandards](#)中的。

## batch-enable-standards

以下代码示例演示了如何使用 batch-enable-standards。

### AWS CLI

#### 启用标准

以下batch-enable-standards示例为请求的账户启用 PCI DSS 标准。

```

aws securityhub batch-enable-standards \
  --standards-subscription-requests '{"StandardsArn":"arn:aws:securityhub:us-
west-1::standards/pci-dss/v/3.2.1"}'

```

输出：

```

{
  "StandardsSubscriptions": [
    {
      "StandardsArn": "arn:aws:securityhub:us-west-1::standards/pci-dss/
v/3.2.1",
      "StandardsInput": { },
      "StandardsStatus": "PENDING",
      "StandardsSubscriptionArn": "arn:aws:securityhub:us-
west-1:123456789012:subscription/pci-dss/v/3.2.1"
    }
  ]
}

```

有关更多信息，请参阅《Security Hub 用户指南》中的禁用或启用AWS 安全[标准](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchEnableStandards](#)中的。

## batch-get-automation-rules

以下代码示例演示了如何使用 batch-get-automation-rules。

### AWS CLI

获取自动化规则的详细信息

以下batch-get-automation-rules示例获取指定自动化规则的详细信息。您只需一个命令即可获得一条或多条自动化规则的详细信息。

```
aws securityhub batch-get-automation-rules \
  --automation-rules-arns '["arn:aws:securityhub:us-
east-1:123456789012:automation-rule/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"]'
```

输出：

```
{
  "Rules": [
    {
      "RuleArn": "arn:aws:securityhub:us-east-1:123456789012:automation-rule/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "RuleStatus": "ENABLED",
      "RuleOrder": 1,
      "RuleName": "Suppress informational findings",
      "Description": "Suppress GuardDuty findings with Informational
severity",
      "IsTerminal": false,
      "Criteria": {
        "ProductName": [
          {
            "Value": "GuardDuty",
            "Comparison": "EQUALS"
          }
        ],
        "SeverityLabel": [
          {
            "Value": "INFORMATIONAL",
            "Comparison": "EQUALS"
          }
        ]
      }
    }
  ],
}
```

```

        "WorkflowStatus": [
            {
                "Value": "NEW",
                "Comparison": "EQUALS"
            }
        ],
        "RecordState": [
            {
                "Value": "ACTIVE",
                "Comparison": "EQUALS"
            }
        ]
    },
    "Actions": [
        {
            "Type": "FINDING_FIELDS_UPDATE",
            "FindingFieldsUpdate": {
                "Note": {
                    "Text": "Automatically suppress GuardDuty findings with
Informational severity",
                    "UpdatedBy": "sechub-automation"
                },
                "Workflow": {
                    "Status": "SUPPRESSED"
                }
            }
        }
    ],
    "CreatedAt": "2023-05-31T17:56:14.837000+00:00",
    "UpdatedAt": "2023-05-31T17:59:38.466000+00:00",
    "CreatedBy": "arn:aws:iam::123456789012:role/Admin"
}
],
"UnprocessedAutomationRules": []
}

```

有关更多信息，请参阅《Sec AWS urity Hub 用户指南》中的[查看自动化规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetAutomationRules](#)中的。

## batch-get-configuration-policy-associations

以下代码示例演示了如何使用 batch-get-configuration-policy-associations。

## AWS CLI

获取一批目标的配置关联详细信息

以下 `batch-get-configuration-policy-associations` 示例检索指定目标的关联详细信息。您可以为目标提供账户 ID、组织单位 ID 或根 ID。

```
aws securityhub batch-get-configuration-policy-associations \
  --target '{"OrganizationalUnitId": "ou-6hi7-8j91k12m"}'
```

输出：

```
{
  "ConfigurationPolicyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "TargetId": "ou-6hi7-8j91k12m",
  "TargetType": "ORGANIZATIONAL_UNIT",
  "AssociationType": "APPLIED",
  "UpdatedAt": "2023-09-26T21:13:01.816000+00:00",
  "AssociationStatus": "SUCCESS",
  "AssociationStatusMessage": "Association applied successfully on this target."
}
```

有关更多信息，请参阅《[Security Hub 用户指南](#)》中的[查看 Sec AWS urity Hub 配置策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchGetConfigurationPolicyAssociations](#) 中的。

## batch-get-security-controls

以下代码示例演示了如何使用 `batch-get-security-controls`。

### AWS CLI

获取安全控制详情

以下 `batch-get-security-controls` 示例获取当前 AWS 账户和区域中安全控制 ACM.1 和 IAM.1 的详细信息。AWS

```
aws securityhub batch-get-security-controls \
  --security-control-ids ['ACM.1', 'IAM.1']'
```

输出：

```
{
  "SecurityControls": [
    {
      "SecurityControlId": "ACM.1",
      "SecurityControlArn": "arn:aws:securityhub:us-
east-2:123456789012:security-control/ACM.1",
      "Title": "Imported and ACM-issued certificates should be renewed after a
specified time period",
      "Description": "This control checks whether an AWS Certificate Manager
(ACM) certificate is renewed within the specified time period. It checks both
imported certificates and certificates provided by ACM. The control fails if the
certificate isn't renewed within the specified time period. Unless you provide a
custom parameter value for the renewal period, Security Hub uses a default value of
30 days.",
      "RemediationUrl": "https://docs.aws.amazon.com/console/securityhub/
ACM.1/remediation",
      "SeverityRating": "MEDIUM",
      "SecurityControlStatus": "ENABLED"
      "UpdateStatus": "READY",
      "Parameters": {
        "daysToExpiration": {
          "ValueType": CUSTOM,
          "Value": {
            "Integer": 15
          }
        }
      }
    },
    "LastUpdateReason": "Updated control parameter"
  ],
  {
    "SecurityControlId": "IAM.1",
    "SecurityControlArn": "arn:aws:securityhub:us-
east-2:123456789012:security-control/IAM.1",
    "Title": "IAM policies should not allow full \"*\" administrative
privileges",
    "Description": "This AWS control checks whether the default version of
AWS Identity and Access Management (IAM) policies (also known as customer managed
policies) do not have administrator access with a statement that has \"Effect\":
\"Allow\" with \"Action\": \"*\" over \"Resource\": \"*\". It only checks for
the Customer Managed Policies that you created, but not inline and AWS Managed
Policies.",
```

```

        "RemediationUrl": "https://docs.aws.amazon.com/console/securityhub/
IAM.1/remediation",
        "SeverityRating": "HIGH",
        "SecurityControlStatus": "ENABLED"
        "UpdateStatus": "READY",
        "Parameters": {}
    }
]
}

```

有关更多信息，请参阅《[Sec AWS urity Hub 用户指南](#)》中的[查看控件的详细信息](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetSecurityControls](#)中的。

## batch-get-standards-control-associations

以下代码示例演示了如何使用 batch-get-standards-control-associations。

### AWS CLI

获取控件的启用状态

以下batch-get-standards-control-associations示例标识了在指定标准中是否启用了指定的控件。

```

aws securityhub batch-get-standards-control-associations \
  --standards-control-association-ids '[{"SecurityControlId":
  "Config.1", "StandardsArn": "arn:aws:securityhub:us-east-1:123456789012:ruleset/cis-
aws-foundations-benchmark/v/1.2.0"}, {"SecurityControlId": "IAM.6", "StandardsArn":
  "arn:aws:securityhub:us-east-1:123456789012:standards/aws-foundational-security-
best-practices/v/1.0.0"}]'

```

输出：

```

{
  "StandardsControlAssociationDetails": [
    {
      "StandardsArn": "arn:aws:securityhub::ruleset/cis-aws-foundations-
benchmark/v/1.2.0",
      "SecurityControlId": "Config.1",
      "SecurityControlArn": "arn:aws:securityhub:us-
east-1:068873283051:security-control/Config.1",

```



```

    "AssociationStatus": "ENABLED",
    "RelatedRequirements": [
      "CIS AWS Foundations 2.5"
    ],
    "UpdatedAt": "2022-10-27T16:07:12.960000+00:00",
    "StandardsControlTitle": "Ensure AWS Config is enabled",
    "StandardsControlDescription": "AWS Config is a web service that
performs configuration management of supported AWS resources within your account
and delivers log files to you. The recorded information includes the configuration
item (AWS resource), relationships between configuration items (AWS resources), and
any configuration changes between resources. It is recommended to enable AWS Config
in all regions.",
    "StandardsControlArns": [
      "arn:aws:securityhub:us-east-1:068873283051:control/cis-aws-
foundations-benchmark/v/1.2.0/2.5"
    ]
  },
  {
    "StandardsArn": "arn:aws:securityhub:us-east-1::standards/aws-
foundational-security-best-practices/v/1.0.0",
    "SecurityControlId": "IAM.6",
    "SecurityControlArn": "arn:aws:securityhub:us-
east-1:068873283051:security-control/IAM.6",
    "AssociationStatus": "DISABLED",
    "RelatedRequirements": [],
    "UpdatedAt": "2022-11-22T21:30:35.080000+00:00",
    "UpdatedReason": "test",
    "StandardsControlTitle": "Hardware MFA should be enabled for the root
user",
    "StandardsControlDescription": "This AWS control checks whether your AWS
account is enabled to use a hardware multi-factor authentication (MFA) device to
sign in with root user credentials.",
    "StandardsControlArns": [
      "arn:aws:securityhub:us-east-1:068873283051:control/aws-
foundational-security-best-practices/v/1.0.0/IAM.6"
    ]
  }
]
}

```

有关更多信息，请参阅 [Sec AWS urity Hub 用户指南中的启用和禁用特定标准中的控件](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考BatchGetStandardsControlAssociations](#) 中的。

## batch-import-findings

以下代码示例演示了如何使用 batch-import-findings。

### AWS CLI

#### 更新调查结果

以下batch-import-findings示例更新了调查结果。

```
aws securityhub batch-import-findings \  
  --findings '  
    [{  
      "AwsAccountId": "123456789012",  
      "CreatedAt": "2020-05-27T17:05:54.832Z",  
      "Description": "Vulnerability in a CloudTrail trail",  
      "FindingProviderFields": {  
        "Severity": {  
          "Label": "LOW",  
          "Original": "10"  
        },  
        "Types": [  
          "Software and Configuration Checks/Vulnerabilities/CVE"  
        ]  
      },  
      "GeneratorId": "TestGeneratorId",  
      "Id": "Id1",  
      "ProductArn": "arn:aws:securityhub:us-  
west-1:123456789012:product/123456789012/default",  
      "Resources": [  
        {  
          "Id": "arn:aws:cloudtrail:us-west-1:123456789012:trail/  
TrailName",  
          "Partition": "aws",  
          "Region": "us-west-1",  
          "Type": "AwsCloudTrailTrail"  
        }  
      ],  
      "SchemaVersion": "2018-10-08",  
      "Title": "CloudTrail trail vulnerability",  
      "UpdatedAt": "2020-06-02T16:05:54.832Z"  
    }]
```

输出：

```
{
  "FailedCount": 0,
  "SuccessCount": 1,
  "FailedFindings": []
}
```

有关更多信息，请参阅《Sec AWS urity Hub 用户指南》中的[使用 BatchImportFindings 来创建和更新调查结果](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchImportFindings](#)中的。

## batch-update-automation-rules

以下代码示例演示了如何使用 batch-update-automation-rules。

### AWS CLI

#### 更新自动化规则

以下batch-update-automation-rules示例更新了指定的自动化规则。您可以使用单个命令更新一条或多条规则。只有 Security Hub 管理员帐户可以运行此命令。

```
aws securityhub batch-update-automation-rules \
  --update-automation-rules-request-items '[ \
    { \
      "Actions": [{ \
        "Type": "FINDING_FIELDS_UPDATE", \
        "FindingFieldsUpdate": { \
          "Note": { \
            "Text": "Known issue that is a risk", \
            "UpdatedBy": "sechub-automation" \
          }, \
          "Workflow": { \
            "Status": "NEW" \
          } \
        } \
      }], \
      "Criteria": { \
        "SeverityLabel": [{ \
          "Value": "LOW", \
          "Comparison": "EQUALS" \
        }] \
    } \
  ]'
```

```

    }, \
    "RuleArn": "arn:aws:securityhub:us-east-1:123456789012:automation-rule/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111", \
    "RuleOrder": 1, \
    "RuleStatus": "DISABLED" \
  } \
]'

```

输出：

```

{
  "ProcessedAutomationRules": [
    "arn:aws:securityhub:us-east-1:123456789012:automation-rule/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  ],
  "UnprocessedAutomationRules": []
}

```

有关更多信息，请参阅《Sec AWS urity Hub 用户指南》中的[编辑自动化规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchUpdateAutomationRules](#)中的。

## batch-update-findings

以下代码示例演示了如何使用 batch-update-findings。

### AWS CLI

#### 示例 1：更新调查结果

以下batch-update-findings示例更新了两个发现结果以添加注释、更改严重性标签并解决该问题。

```

aws securityhub batch-update-findings \
  --finding-identifiers '[{"Id": "arn:aws:securityhub:us-
west-1:123456789012:subscription/pci-dss/v/3.2.1/PCI.Lambda.2/finding/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111", "ProductArn": "arn:aws:securityhub:us-
west-1::product/aws/securityhub"}, {"Id": "arn:aws:securityhub:us-
west-1:123456789012:subscription/pci-dss/v/3.2.1/PCI.Lambda.2/finding/
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222", "ProductArn": "arn:aws:securityhub:us-
west-1::product/aws/securityhub"}]' \
  --note '{"Text": "Known issue that is not a risk.", "UpdatedBy": "user1"}' \

```

```
--severity '{"Label": "LOW"}' \
--workflow '{"Status": "RESOLVED"}'
```

输出：

```
{
  "ProcessedFindings": [
    {
      "Id": "arn:aws:securityhub:us-west-1:123456789012:subscription/pci-dss/v/3.2.1/PCI.Lambda.2/finding/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "ProductArn": "arn:aws:securityhub:us-west-1::product/aws/securityhub"
    },
    {
      "Id": "arn:aws:securityhub:us-west-1:123456789012:subscription/pci-dss/v/3.2.1/PCI.Lambda.2/finding/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "ProductArn": "arn:aws:securityhub:us-west-1::product/aws/securityhub"
    }
  ],
  "UnprocessedFindings": []
}
```

有关更多信息，请参阅《[Sec AWS ur BatchUpdateFindings ity Hub 用户指南](#)》中的[使用更新调查结果](#)。

示例 2：使用速记语法更新调查结果

以下batch-update-findings示例更新了两个发现结果，以添加注释、更改严重性标签并使用速记语法解决该问题。

```
aws securityhub batch-update-findings \
  --finding-identifiers Id="arn:aws:securityhub:us-west-1:123456789012:subscription/pci-dss/v/3.2.1/PCI.Lambda.2/finding/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",ProductArn="arn:aws:securityhub:us-west-1::product/aws/securityhub" Id="arn:aws:securityhub:us-west-1:123456789012:subscription/pci-dss/v/3.2.1/PCI.Lambda.2/finding/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",ProductArn="arn:aws:securityhub:us-west-1::product/aws/securityhub" \
  --note Text="Known issue that is not a risk.",UpdatedBy="user1" \
  --severity Label="LOW" \
  --workflow Status="RESOLVED"
```

输出：

```
{
  "ProcessedFindings": [
    {
      "Id": "arn:aws:securityhub:us-west-1:123456789012:subscription/pci-dss/v/3.2.1/PCI.Lambda.2/finding/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "ProductArn": "arn:aws:securityhub:us-west-1::product/aws/securityhub"
    },
    {
      "Id": "arn:aws:securityhub:us-west-1:123456789012:subscription/pci-dss/v/3.2.1/PCI.Lambda.2/finding/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "ProductArn": "arn:aws:securityhub:us-west-1::product/aws/securityhub"
    }
  ],
  "UnprocessedFindings": []
}
```

有关更多信息，请参阅《Sec AWS ur BatchUpdateFindings ity Hub 用户指南》中的[使用更新调查结果](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchUpdateFindings](#)中的。

## batch-update-standards-control-associations

以下代码示例演示了如何使用 batch-update-standards-control-associations。

### AWS CLI

更新已启用标准中控件的启用状态

以下batch-update-standards-control-associations示例禁用了指定标准中的CloudTrail.1。

```
aws securityhub batch-update-standards-control-associations \
  --standards-control-association-updates '[{"SecurityControlId": "CloudTrail.1",
  "StandardsArn": "arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0", "AssociationStatus": "DISABLED", "UpdatedReason": "Not applicable to environment"}, {"SecurityControlId": "CloudTrail.1", "StandardsArn": "arn:aws:securityhub:::standards/cis-aws-foundations-benchmark/v/1.4.0", "AssociationStatus": "DISABLED", "UpdatedReason": "Not applicable to environment"}]'
```

如果成功，此命令不会产生任何输出。

有关更多信息，请参阅《[Sec AWS urity Hub 用户指南](#)》中的[启用和禁用特定标准中的控件以及启用和禁用所有标准](#)中的控件。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchUpdateStandardsControlAssociations](#)中的。

## create-action-target

以下代码示例演示了如何使用 create-action-target。

### AWS CLI

#### 创建自定义操作

以下create-action-target示例创建了一个自定义操作。它提供操作的名称、描述和标识符。

```
aws securityhub create-action-target \  
  --name "Send to remediation" \  
  --description "Action to send the finding for remediation tracking" \  
  --id "Remediation"
```

输出：

```
{  
  "ActionTargetArn": "arn:aws:securityhub:us-west-1:123456789012:action/custom/  
Remediation"  
}
```

有关更多信息，请参阅《[Sec AWS urity Hub 用户指南](#)》中的[创建自定义操作并将其与CloudWatch 事件规则关联](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateActionTarget](#)中的。

## create-automation-rule

以下代码示例演示了如何使用 create-automation-rule。

### AWS CLI

#### 创建自动化规则

以下 `create-automation-rule` 示例在当前 AWS 账户和 AWS 区域中创建自动化规则。Security Hub 根据指定的条件筛选您的发现，并将操作应用于匹配的结果。只有 Security Hub 管理员帐户可以运行此命令。

```
aws securityhub create-automation-rule \
  --actions '[{ \
    "Type": "FINDING_FIELDS_UPDATE", \
    "FindingFieldsUpdate": { \
      "Severity": { \
        "Label": "HIGH" \
      }, \
      "Note": { \
        "Text": "Known issue that is a risk. Updated by automation rules", \
        "UpdatedBy": "sechub-automation" \
      } \
    } \
  }]' \
  --criteria '{ \
    "SeverityLabel": [{ \
      "Value": "INFORMATIONAL", \
      "Comparison": "EQUALS" \
    }] \
  }' \
  --description "A sample rule" \
  --no-is-terminal \
  --rule-name "sample rule" \
  --rule-order 1 \
  --rule-status "ENABLED"
```

输出：

```
{
  "RuleArn": "arn:aws:securityhub:us-east-1:123456789012:automation-rule/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

有关更多信息，请参阅《Sec AWS urity Hub 用户指南》中的[创建自动化规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateAutomationRule](#)中的。



## create-configuration-policy

以下代码示例演示了如何使用 create-configuration-policy。

### AWS CLI

#### 创建配置策略

以下create-configuration-policy示例使用指定设置创建配置策略。

```
aws securityhub create-configuration-policy \  
  --name "SampleConfigurationPolicy" \  
  --description "SampleDescription" \  
  --configuration-policy '{"SecurityHub": {"ServiceEnabled":  
  true, "EnabledStandardIdentifiers": ["arn:aws:securityhub:eu-  
central-1::standards/aws-foundational-security-best-practices/  
v/1.0.0", "arn:aws:securityhub::ruleset/cis-aws-foundations-benchmark/  
v/1.2.0"], "SecurityControlsConfiguration": {"DisabledSecurityControlIdentifiers":  
  ["CloudTrail.2"], "SecurityControlCustomParameters": [{"SecurityControlId":  
  "ACM.1", "Parameters": {"daysToExpiration": {"ValueType": "CUSTOM", "Value":  
  {"Integer": 15}}}}]}'} \  
  --tags '{"Environment": "Prod"}
```

输出：

```
{  
  "Arn": "arn:aws:securityhub:eu-central-1:123456789012:configuration-policy/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
  "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
  "Name": "SampleConfigurationPolicy",  
  "Description": "SampleDescription",  
  "UpdatedAt": "2023-11-28T20:28:04.494000+00:00",  
  "CreatedAt": "2023-11-28T20:28:04.494000+00:00",  
  "ConfigurationPolicy": {  
    "SecurityHub": {  
      "ServiceEnabled": true,  
      "EnabledStandardIdentifiers": [  
        "arn:aws:securityhub:eu-central-1::standards/aws-foundational-  
security-best-practices/v/1.0.0",  
        "arn:aws:securityhub::ruleset/cis-aws-foundations-benchmark/  
v/1.2.0"  
      ],  
      "SecurityControlsConfiguration": {
```

```

    "DisabledSecurityControlIdentifiers": [
      "CloudTrail.2"
    ],
    "SecurityControlCustomParameters": [
      {
        "SecurityControlId": "ACM.1",
        "Parameters": {
          "daysToExpiration": {
            "ValueType": "CUSTOM",
            "Value": {
              "Integer": 15
            }
          }
        }
      }
    ]
  }
}

```

有关更多信息，请参阅《[Security Hub 用户指南](#)》中的[创建和关联 S AWS security Hub 配置策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateConfigurationPolicy](#) 中的。

## create-finding-aggregator

以下代码示例演示了如何使用 create-finding-aggregator。

### AWS CLI

#### 启用查找结果聚合

以下 create-finding-aggregator 示例配置了查找聚合。它从美国东部（弗吉尼亚州）运营，后者将美国东部（弗吉尼亚州）指定为聚合区域。它表示仅链接指定区域，不自动关联新区域。它选择美国西部（加利福尼亚北部）和美国西部（俄勒冈）作为关联区域。

```

aws securityhub create-finding-aggregator \
  --region us-east-1 \
  --region-linking-mode SPECIFIED_REGIONS \
  --regions us-west-1,us-west-2

```

输出：

```
{
  "FindingAggregatorArn": "arn:aws:securityhub:us-east-1:222222222222:finding-
aggregator/123e4567-e89b-12d3-a456-426652340000",
  "FindingAggregationRegion": "us-east-1",
  "RegionLinkingMode": "SPECIFIED_REGIONS",
  "Regions": "us-west-1,us-west-2"
}
```

有关更多信息，请参阅《Sec AWS urity Hub 用户指南》中的[启用查找结果聚合](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFindingAggregator](#)中的。

## create-insight

以下代码示例演示了如何使用 create-insight。

### AWS CLI

#### 创建自定义见解

以下create-insight示例创建了一个名为“关键角色调查结果”的自定义见解，该洞察返回与AWS 角色相关的关键发现。

```
aws securityhub create-insight \
  --filters '{"ResourceType": [{"Comparison": "EQUALS", "Value": "AwsIamRole"}],
"SeverityLabel": [{"Comparison": "EQUALS", "Value": "CRITICAL"}]}' \
  --group-by-attribute "ResourceId" \
  --name "Critical role findings"
```

输出：

```
{
  "InsightArn": "arn:aws:securityhub:us-west-1:123456789012:insight/123456789012/
custom/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理自定义见解](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateInsight](#)中的。

## create-members

以下代码示例演示了如何使用 create-members。

### AWS CLI

将账户添加为成员账户

以下create-members示例将两个账户作为成员账户添加到请求的管理员账户。

```
aws securityhub create-members \  
  --account-details '[{"AccountId": "123456789111"}, {"AccountId":  
  "123456789222"}]'
```

输出：

```
{  
  "UnprocessedAccounts": []  
}
```

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateMembers](#)中的。

## decline-invitations

以下代码示例演示了如何使用 decline-invitations。

### AWS CLI

拒绝成为成员账户的邀请

以下decline-invitations示例拒绝了成为指定管理员账户成员账户的邀请。成员账户是请求的账户。

```
aws securityhub decline-invitations \  
  --account-ids "123456789012"
```

输出：

```
{  
  "UnprocessedAccounts": []  
}
```

```
}
```

有关更多信息，请参阅 [Sec AWS urity Hub 用户指南](#) 中的 [管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeclineInvitations](#) 中的。

## delete-action-target

以下代码示例演示了如何使用 delete-action-target。

### AWS CLI

删除自定义操作

以下 delete-action-target 示例删除由指定 ARN 标识的自定义操作。

```
aws securityhub delete-action-target \  
  --action-target-arn "arn:aws:securityhub:us-west-1:123456789012:action/custom/  
Remediation"
```

输出：

```
{  
  "ActionTargetArn": "arn:aws:securityhub:us-west-1:123456789012:action/custom/  
Remediation"  
}
```

有关更多信息，请参阅《[Sec AWS urity Hub 用户指南](#)》中的 [创建自定义操作并将其与 CloudWatch 事件规则关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteActionTarget](#) 中的。

## delete-configuration-policy

以下代码示例演示了如何使用 delete-configuration-policy。

### AWS CLI

要删除配置策略

以下 delete-configuration-policy 示例删除了指定的配置策略。

```
aws securityhub delete-configuration-policy \  

```

```
--identifier "arn:aws:securityhub:eu-central-1:123456789012:configuration-policy/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

此命令不生成任何输出。

有关更多信息，请参阅《[Security Hub 用户指南](#)》中的[删除和取消关联 Sec AWS urity Hub 配置策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteConfigurationPolicy](#)中的。

## delete-finding-aggregator

以下代码示例演示了如何使用 delete-finding-aggregator。

### AWS CLI

#### 停止查找聚合

以下delete-finding-aggregator示例停止查找聚合。它从美国东部（弗吉尼亚州）运营，这是聚合区域。

```
aws securityhub delete-finding-aggregator \  
  --region us-east-1 \  
  --finding-aggregator-arn arn:aws:securityhub:us-east-1:222222222222:finding-aggregator/123e4567-e89b-12d3-a456-426652340000
```

此命令不生成任何输出。

有关更多信息，请参阅《[Sec AWS urity Hub 用户指南](#)》中的[停止查找聚合](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFindingAggregator](#)中的。

## delete-insight

以下代码示例演示了如何使用 delete-insight。

### AWS CLI

#### 删除自定义见解

以下delete-insight示例删除具有指定 ARN 的自定义分析。

```
aws securityhub delete-insight \  
  --arn arn:aws:securityhub:us-east-1:123456789012:insight/12345678-90ab-cdef-EXAMPLE11111
```

```
--insight-arn "arn:aws:securityhub:us-west-1:123456789012:insight/123456789012/custom/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

输出：

```
{
  "InsightArn": "arn:aws:securityhub:eu-central-1:123456789012:insight/123456789012/custom/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理自定义见解](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInsight](#)中的。

## delete-invitations

以下代码示例演示了如何使用 delete-invitations。

### AWS CLI

删除成为成员账户的邀请

以下delete-invitations示例删除了指定管理员账户的成员账户邀请。成员账户是请求的账户。

```
aws securityhub delete-invitations \
  --account-ids "123456789012"
```

输出：

```
{
  "UnprocessedAccounts": []
}
```

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInvitations](#)中的。

## delete-members

以下代码示例演示了如何使用 delete-members。

## AWS CLI

### 删除成员账户

以下delete-members示例从请求的管理员帐户中删除指定的成员帐户。

```
aws securityhub delete-members \  
  --account-ids "123456789111" "123456789222"
```

输出：

```
{  
  "UnprocessedAccounts": []  
}
```

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMembers](#)中的。

## describe-action-targets

以下代码示例演示了如何使用 describe-action-targets。

### AWS CLI

检索有关自定义操作的详细信息

以下describe-action-targets示例检索有关由指定 ARN 标识的自定义操作的信息。

```
aws securityhub describe-action-targets \  
  --action-target-arns "arn:aws:securityhub:us-west-1:123456789012:action/custom/  
Remediation"
```

输出：

```
{  
  "ActionTargets": [  
    {  
      "ActionTargetArn": "arn:aws:securityhub:us-west-1:123456789012:action/  
custom/Remediation",  
      "Description": "Action to send the finding for remediation tracking",  
      "Name": "Send to remediation"  
    }  
  ]  
}
```



```
]
}
```

有关更多信息，请参阅《[Security Hub 用户指南](#)》中的[创建自定义操作并将其与 CloudWatch 事件规则关联](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeActionTargets](#)中的。

## describe-hub

以下代码示例演示了如何使用 describe-hub。

### AWS CLI

获取有关中心资源的信息

以下describe-hub示例返回指定中心资源的订阅日期。中心资源由其 ARN 标识。

```
aws securityhub describe-hub \
  --hub-arn "arn:aws:securityhub:us-west-1:123456789012:hub/default"
```

输出：

```
{
  "HubArn": "arn:aws:securityhub:us-west-1:123456789012:hub/default",
  "SubscribedAt": "2019-11-19T23:15:10.046Z"
}
```

有关更多信息，请参阅《AWS CloudFormation 用户指南》中的[AWS SecurityHub::: Hub](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeHub](#)中的。

## describe-organization-configuration

以下代码示例演示了如何使用 describe-organization-configuration。

### AWS CLI

查看如何为组织配置 Security Hub

以下describe-organization-configuration示例返回有关在 Security Hub 中配置组织的方式的信息。在此示例中，组织使用中央配置。只有 Security Hub 管理员帐户可以运行此命令。

```
aws securityhub describe-organization-configuration
```

输出：

```
{
  "AutoEnable": false,
  "MemberAccountLimitReached": false,
  "AutoEnableStandards": "NONE",
  "OrganizationConfiguration": {
    "ConfigurationType": "LOCAL",
    "Status": "ENABLED",
    "StatusMessage": "Central configuration has been enabled successfully"
  }
}
```

有关更多信息，请参阅《Sec AWS urity Hub 用户指南 AWS 》中的 [Organizations 账户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeOrganizationConfiguration](#)中的。

## describe-products

以下代码示例演示了如何使用 describe-products。

### AWS CLI

返回有关可用产品集成的信息

以下describe-products示例逐一返回可用的产品集成。

```
aws securityhub describe-products \
  --max-results 1
```

输出：

```
{
  "NextToken": "U2FsdGVkX18vvP10qb7RD1rWRWVFBJI46M0IAb+nZmRJmR15NoRi2gm13sdQEn30/
  pq/78dGs+bKpgA+7HMPH00qX33/zoRI+uIG/F9yLNhc0r0WzFUdy36JcXLQji3Rpnn/
  cD1SVkGA98qI3zPOSDg==",
  "Products": [
    {
      "ProductArn": "arn:aws:securityhub:us-west-1:123456789333:product/
      crowdstrike/crowdstrike-falcon",
```

```

    "ProductName": "CrowdStrike Falcon",
    "CompanyName": "CrowdStrike",
    "Description": "CrowdStrike Falcon's single lightweight sensor unifies
next-gen antivirus, endpoint detection and response, and 24/7 managed hunting, via
the cloud.",
    "Categories": [
        "Endpoint Detection and Response (EDR)",
        "AV Scanning and Sandboxing",
        "Threat Intelligence Feeds and Reports",
        "Endpoint Forensics",
        "Network Forensics"
    ],
    "IntegrationTypes": [
        "SEND_FINDINGS_TO_SECURITY_HUB"
    ],
    "MarketplaceUrl": "https://aws.amazon.com/marketplace/seller-profile?
id=a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "ActivationUrl": "https://falcon.crowdstrike.com/support/documentation",
    "ProductSubscriptionResourcePolicy": "{\"Version\":
\\\"2012-10-17\\\",\\\"Statement\\\":[\\\"Effect\\\":\\\"Allow\\\",\\\"Principal\\\":{\\\"AWS\\\":
\\\"123456789333\\\"},\\\"Action\\\":[\\\"securityhub:BatchImportFindings\\\"],\\\"Resource\\\":
\\\"arn:aws:securityhub:us-west-1:123456789012:product-subscription/crowdstrike/
crowdstrike-falcon\\\",\\\"Condition\\\":{\\\"StringEquals\\\":{\\\"securityhub:TargetAccount
\\\":\\\"123456789012\\\"}}},{\\\"Effect\\\":\\\"Allow\\\",\\\"Principal\\\":{\\\"AWS\\\":
\\\"123456789012\\\"},\\\"Action\\\":[\\\"securityhub:BatchImportFindings\\\"],\\\"Resource
\\\":\\\"arn:aws:securityhub:us-west-1:123456789333:product/crowdstrike/crowdstrike-
falcon\\\",\\\"Condition\\\":{\\\"StringEquals\\\":{\\\"securityhub:TargetAccount\\\":
\\\"123456789012\\\"}}]}]}\"
    }
  ]
}

```

有关更多信息，请参阅 [Sec AWS urity Hub 用户指南](#) 中的管理产品集成。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 DescribeProducts](#) 中的。

## describe-standards-controls

以下代码示例演示了如何使用 describe-standards-controls。

### AWS CLI

请求已启用的标准中的控件列表

以下describe-standards-controls示例请求请求者账户订阅 PCI DSS 标准时的控制列表。该请求一次返回两个控件。

```
aws securityhub describe-standards-controls \
  --standards-subscription-arn "arn:aws:securityhub:us-
west-1:123456789012:subscription/pci-dss/v/3.2.1" \
  --max-results 2
```

输出：

```
{
  "Controls": [
    {
      "StandardsControlArn": "arn:aws:securityhub:us-
west-1:123456789012:control/pci-dss/v/3.2.1/PCI.AutoScaling.1",
      "ControlStatus": "ENABLED",
      "ControlStatusUpdatedAt": "2020-05-15T18:49:04.473000+00:00",
      "ControlId": "PCI.AutoScaling.1",
      "Title": "Auto scaling groups associated with a load balancer should use
health checks",
      "Description": "This AWS control checks whether your Auto Scaling groups
that are associated with a load balancer are using Elastic Load Balancing health
checks.",
      "RemediationUrl": "https://docs.aws.amazon.com/console/securityhub/
PCI.AutoScaling.1/remediation",
      "SeverityRating": "LOW",
      "RelatedRequirements": [
        "PCI DSS 2.2"
      ]
    },
    {
      "StandardsControlArn": "arn:aws:securityhub:us-
west-1:123456789012:control/pci-dss/v/3.2.1/PCI.CW.1",
      "ControlStatus": "ENABLED",
      "ControlStatusUpdatedAt": "2020-05-15T18:49:04.498000+00:00",
      "ControlId": "PCI.CW.1",
      "Title": "A log metric filter and alarm should exist for usage of the
\"root\" user",
      "Description": "This control checks for the CloudWatch metric
filters using the following pattern { $.userIdentity.type = \"Root\" &&
$.userIdentity.invokedBy NOT EXISTS && $.eventType != \"AwsServiceEvent\" }
It checks that the log group name is configured for use with active multi-
region CloudTrail, that there is at least one Event Selector for a Trail with
```

```

IncludeManagementEvents set to true and ReadWriteType set to All, and that there is
at least one active subscriber to an SNS topic associated with the alarm.",
    "RemediationUrl": "https://docs.aws.amazon.com/console/securityhub/
PCI.CW.1/remediation",
    "SeverityRating": "MEDIUM",
    "RelatedRequirements": [
        "PCI DSS 7.2.1"
    ]
}
],
"NextToken": "U2FsdGvkX1+eNkPoZHV111ip5HUYQPWSWZGmftcmJiHL8JoKEsCDuaKayiPDyLK
+LiTkShveo0dvfxXck0BaGhohIXhsIedN+LSjQV/
17kfCfJcq4PziNC1N9xe9aq2pjlLVZnznTfSImrodT5bRNHe4fELCQq/z+5ka
+5Lzmc11axcwTd5lKgQyQqmUVoeriHZhyIiBgWKf7oNYdBVG80EortVWvSkoUTt
+B2ThcnC7143kI0UNx1kZ6sc64AsW"
}

```

有关更多信息，请参阅《[Sec AWS urity Hub 用户指南](#)》中的[查看控件详情](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeStandardsControls](#)中的。

## describe-standards

以下代码示例演示了如何使用 describe-standards。

### AWS CLI

返回可用标准列表

以下describe-standards示例返回可用标准的列表。

```
aws securityhub describe-standards
```

输出：

```

{
  "Standards": [
    {
      "StandardsArn": "arn:aws:securityhub:us-west-1::standards/aws-
foundational-security-best-practices/v/1.0.0",
      "Name": "AWS Foundational Security Best Practices v1.0.0",
      "Description": "The AWS Foundational Security Best Practices standard
is a set of automated security checks that detect when AWS accounts and deployed

```

```

resources do not align to security best practices. The standard is defined by AWS
security experts. This curated set of controls helps improve your security posture
in AWS, and cover AWS's most popular and foundational services.",
    "EnabledByDefault": true
  },
  {
    "StandardsArn": "arn:aws:securityhub:::ruleset/cis-aws-foundations-
benchmark/v/1.2.0",
    "Name": "CIS AWS Foundations Benchmark v1.2.0",
    "Description": "The Center for Internet Security (CIS) AWS Foundations
Benchmark v1.2.0 is a set of security configuration best practices for AWS. This
Security Hub standard automatically checks for your compliance readiness against a
subset of CIS requirements.",
    "EnabledByDefault": true
  },
  {
    "StandardsArn": "arn:aws:securityhub:us-west-1::standards/pci-dss/
v/3.2.1",
    "Name": "PCI DSS v3.2.1",
    "Description": "The Payment Card Industry Data Security Standard (PCI
DSS) v3.2.1 is an information security standard for entities that store, process,
and/or transmit cardholder data. This Security Hub standard automatically checks
for your compliance readiness against a subset of PCI DSS requirements.",
    "EnabledByDefault": false
  }
]
}

```

有关更多信息，请参阅 [Security Hub 用户指南中的 Sec AWS urity Hub 中的安全标准](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeStandards](#) 中的。

## disable-import-findings-for-product

以下代码示例演示了如何使用 `disable-import-findings-for-product`。

### AWS CLI

停止接收来自产品集成的调查结果

以下 `disable-import-findings-for-product` 示例禁用产品集成的指定订阅的结果流。

```
aws securityhub disable-import-findings-for-product \
```

```
--product-subscription-arn "arn:aws:securityhub:us-west-1:123456789012:product-subscription/crowdstrike/crowdstrike-falcon"
```

此命令不生成任何输出。

有关更多信息，请参阅 [Sec AWS urity Hub 用户指南](#) 中的管理产品集成。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisableImportFindingsForProduct](#) 中的。

## disable-organization-admin-account

以下代码示例演示了如何使用 `disable-organization-admin-account`。

### AWS CLI

移除 Security Hub 管理员帐户

以下 `disable-organization-admin-account` 示例撤消了指定账户作为 Organizations 的 Security Hub 管理员账户的分配 AWS 。

```
aws securityhub disable-organization-admin-account \  
--admin-account-id 777788889999
```

此命令不生成任何输出。

有关更多信息，请参阅《[Sec urity Hub 用户指南](#)》中的“指定 Sec AWS urity Hub 管理员帐户”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisableOrganizationAdminAccount](#) 中的。

## disable-security-hub

以下代码示例演示了如何使用 `disable-security-hub`。

### AWS CLI

禁用 S AWS ecurity Hub

以下 `disable-security-hub` 示例为请求的账户禁用 S AWS ecurity Hub。

```
aws securityhub disable-security-hub
```

此命令不生成任何输出。

有关更多信息，请参阅《Sec [AWS urity Hub 用户指南](#)》中的[禁用 S AWS ecurity Hub](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisableSecurityHub](#)中的。

## **disassociate-from-administrator-account**

以下代码示例演示了如何使用 disassociate-from-administrator-account。

### AWS CLI

取消与管理员帐户的关联

以下disassociate-from-administrator-account示例取消请求账户与其当前管理员账户的关联。

```
aws securityhub disassociate-from-administrator-account
```

此命令不生成任何输出。

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateFromAdministratorAccount](#)中的。

## **disassociate-from-master-account**

以下代码示例演示了如何使用 disassociate-from-master-account。

### AWS CLI

取消与管理员帐户的关联

以下disassociate-from-master-account示例取消请求账户与其当前管理员账户的关联。

```
aws securityhub disassociate-from-master-account
```

此命令不生成任何输出。

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateFromMasterAccount](#)中的。



## disassociate-members

以下代码示例演示了如何使用 disassociate-members。

### AWS CLI

取消成员账户的关联

以下disassociate-members示例取消指定成员账户与请求管理员账户的关联。

```
aws securityhub disassociate-members \  
  --account-ids "123456789111" "123456789222"
```

此命令不生成任何输出。

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateMembers](#)中的。

## enable-import-findings-for-product

以下代码示例演示了如何使用 enable-import-findings-for-product。

### AWS CLI

开始接收产品集成的调查结果

以下enable-import-findings-for-product示例启用了来自指定产品集成的结果流。

```
aws securityhub enable-import-findings-for-product \  
  --product-arn "arn:aws:securityhub:us-east-1:123456789333:product/crowdstrike/  
crowdstrike-falcon"
```

输出：

```
{  
  "ProductSubscriptionArn": "arn:aws:securityhub:us-east-1:123456789012:product-  
subscription/crowdstrike/crowdstrike-falcon"  
}
```

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理产品集成](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableImportFindingsForProduct](#)中的。

## enable-organization-admin-account

以下代码示例演示了如何使用 `enable-organization-admin-account`。

### AWS CLI

将组织帐户指定为 Security Hub 管理员帐户

以下`enable-organization-admin-account`示例将指定帐户指定为 Security Hub 管理员帐户。

```
aws securityhub enable-organization-admin-account \  
  --admin-account-id 777788889999
```

此命令不生成任何输出。

有关更多信息，请参阅《[Security Hub 用户指南](#)》中的“[指定 Security Hub 管理员帐户](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableOrganizationAdminAccount](#)中的。

## enable-security-hub

以下代码示例演示了如何使用 `enable-security-hub`。

### AWS CLI

启用 S AWS ecurity Hub

以下`enable-security-hub`示例为请求的帐户启用 S AWS ecurity Hub。它将 Security Hub 配置为启用默认标准。对于中心资源，它为标签`SecurityDepartment`分配值。

```
aws securityhub enable-security-hub \  
  --enable-default-standards \  
  --tags '{"Department": "Security"}'
```

此命令不生成任何输出。

有关更多信息，请参阅 [Security Hub 用户指南中的启用 S AWS ecurity Hub](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[EnableSecurityHub](#)中的。

## get-administrator-account

以下代码示例演示了如何使用 `get-administrator-account`。

### AWS CLI

检索有关管理员帐户的信息

以下 `get-administrator-account` 示例检索有关请求账户的管理员帐户的信息。

```
aws securityhub get-administrator-account
```

输出：

```
{
  "Master": {
    "AccountId": "123456789012",
    "InvitationId": "7ab938c5d52d7904ad09f9e7c20cc4eb",
    "InvitedAt": 2020-06-01T20:21:18.042000+00:00,
    "MemberStatus": "ASSOCIATED"
  }
}
```

有关更多信息，请参阅 [Sec AWS urity Hub 用户指南中的管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `GetAdministratorAccount`](#) 中的。

## get-configuration-policy-association

以下代码示例演示了如何使用 `get-configuration-policy-association`。

### AWS CLI

获取目标的配置关联详细信息

以下 `get-configuration-policy-association` 示例检索指定目标的关联详细信息。您可以为目标提供帐户 ID、组织单位 ID 或根 ID。

```
aws securityhub get-configuration-policy-association \
  --target '{"OrganizationalUnitId": "ou-6hi7-8j91k12m"}
```

输出：

```
{
  "ConfigurationPolicyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "TargetId": "ou-6hi7-8j91kl2m",
  "TargetType": "ORGANIZATIONAL_UNIT",
  "AssociationType": "APPLIED",
  "UpdatedAt": "2023-09-26T21:13:01.816000+00:00",
  "AssociationStatus": "SUCCESS",
  "AssociationStatusMessage": "Association applied successfully on this target."
}
```

有关更多信息，请参阅《[Security Hub 用户指南](#)》中的[查看 Security Hub 配置策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetConfigurationPolicyAssociation](#)中的。

## get-configuration-policy

以下代码示例演示了如何使用 get-configuration-policy。

### AWS CLI

查看配置策略详细信息

以下 get-configuration-policy 示例检索有关指定配置策略的详细信息。

```
aws securityhub get-configuration-policy \
  --identifier "arn:aws:securityhub:eu-central-1:123456789012:configuration-policy/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

输出：

```
{
  "Arn": "arn:aws:securityhub:eu-central-1:123456789012:configuration-policy/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "Id": "ce5ed1e7-9639-4e2f-9313-fa87fcef944b",
  "Name": "SampleConfigurationPolicy",
  "Description": "SampleDescription",
  "UpdatedAt": "2023-11-28T20:28:04.494000+00:00",
  "CreatedAt": "2023-11-28T20:28:04.494000+00:00",
  "ConfigurationPolicy": {
    "SecurityHub": {
      "ServiceEnabled": true,

```

```

    "EnabledStandardIdentifiers": [
      "arn:aws:securityhub:eu-central-1::standards/aws-foundational-
security-best-practices/v/1.0.0",
      "arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/
v/1.2.0"
    ],
    "SecurityControlsConfiguration": {
      "DisabledSecurityControlIdentifiers": [
        "CloudTrail.2"
      ],
      "SecurityControlCustomParameters": [
        {
          "SecurityControlId": "ACM.1",
          "Parameters": {
            "daysToExpiration": {
              "ValueType": "CUSTOM",
              "Value": {
                "Integer": 15
              }
            }
          }
        }
      ]
    }
  }
}

```

有关更多信息，请参阅《[Security Hub 用户指南](#)》中的[查看 Sec AWS urity Hub 配置策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetConfigurationPolicy](#)中的。

## get-enabled-standards

以下代码示例演示了如何使用 get-enabled-standards。

### AWS CLI

检索有关已启用标准的信息

以下get-enabled-standards示例检索有关 PCI DSS 标准的信息。

```
aws securityhub get-enabled-standards \
```

```
--standards-subscription-arn "arn:aws:securityhub:us-west-1:123456789012:subscription/pci-dss/v/3.2.1"
```

输出：

```
{
  "StandardsSubscriptions": [
    {
      "StandardsArn": "arn:aws:securityhub:us-west-1::standards/pci-dss/v/3.2.1",
      "StandardsInput": { },
      "StandardsStatus": "READY",
      "StandardsSubscriptionArn": "arn:aws:securityhub:us-west-1:123456789012:subscription/pci-dss/v/3.2.1"
    }
  ]
}
```

有关更多信息，请参阅 [Security Hub 用户指南中的 Sec AWS urity Hub 中的安全标准](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetEnabledStandards](#) 中的。

## get-finding-aggregator

以下代码示例演示了如何使用 get-finding-aggregator。

AWS CLI

检索当前的查找结果聚合配置

以下 get-finding-aggregator 示例检索当前的查找结果聚合配置。

```
aws securityhub get-finding-aggregator \
  --finding-aggregator-arn arn:aws:securityhub:us-east-1:222222222222:finding-aggregator/123e4567-e89b-12d3-a456-426652340000
```

输出：

```
{
  "FindingAggregatorArn": "arn:aws:securityhub:us-east-1:222222222222:finding-aggregator/123e4567-e89b-12d3-a456-426652340000",
  "FindingAggregationRegion": "us-east-1",
  "RegionLinkingMode": "SPECIFIED_REGIONS",
```

```
"Regions": "us-west-1,us-west-2"
}
```

有关更多信息，请参阅 [《Sec AWS urity Hub 用户指南》](#) 中的 [查看当前查找结果聚合配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetFindingAggregator](#) 中的。

## get-finding-history

以下代码示例演示了如何使用 `get-finding-history`。

### AWS CLI

要获取查找历史记录

以下 `get-finding-history` 示例获取指定查找结果的最近 90 天的历史记录。在此示例中，结果仅限于两条查找历史记录。

```
aws securityhub get-finding-history \
  --finding-identifier Id="arn:aws:securityhub:us-
east-1:123456789012:security-control/S3.17/finding/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",ProductArn="arn:aws:securityhub:us-east-1::product/aws/securityhub"
```

输出：

```
{
  "Records": [
    {
      "FindingIdentifier": {
        "Id": "arn:aws:securityhub:us-east-1:123456789012:security-control/
S3.17/finding/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "ProductArn": "arn:aws:securityhub:us-east-1::product/aws/
securityhub"
      },
      "UpdateTime": "2023-06-02T03:15:25.685000+00:00",
      "FindingCreated": false,
      "UpdateSource": {
        "Type": "BATCH_IMPORT_FINDINGS",
        "Identity": "arn:aws:securityhub:us-east-1::product/aws/securityhub"
      },
      "Updates": [
        {
          "UpdatedField": "Compliance.RelatedRequirements",
```

```

        "OldValue": "[\"NIST.800-53.r5 SC-12(2)\", \"NIST.800-53.r5
SC-12(3)\", \"NIST.800-53.r5 SC-12(6)\", \"NIST.800-53.r5 CM-3(6)\", \"NIST.800-53.r5
SC-13\", \"NIST.800-53.r5 SC-28\", \"NIST.800-53.r5 SC-28(1)\", \"NIST.800-53.r5
SC-7(10)\"]",
        "NewValue": "[\"NIST.800-53.r5 SC-12(2)\", \"NIST.800-53.r5
CM-3(6)\", \"NIST.800-53.r5 SC-13\", \"NIST.800-53.r5 SC-28\", \"NIST.800-53.r5
SC-28(1)\", \"NIST.800-53.r5 SC-7(10)\", \"NIST.800-53.r5 CA-9(1)\", \"NIST.800-53.r5
SI-7(6)\", \"NIST.800-53.r5 AU-9\"]"
    },
    {
        "UpdatedField": "LastObservedAt",
        "OldValue": "2023-06-01T09:15:38.587Z",
        "NewValue": "2023-06-02T03:15:22.946Z"
    },
    {
        "UpdatedField": "UpdatedAt",
        "OldValue": "2023-06-01T09:15:31.049Z",
        "NewValue": "2023-06-02T03:15:14.861Z"
    },
    {
        "UpdatedField": "ProcessedAt",
        "OldValue": "2023-06-01T09:15:41.058Z",
        "NewValue": "2023-06-02T03:15:25.685Z"
    }
]
},
{
    "FindingIdentifier": {
        "Id": "arn:aws:securityhub:us-east-1:123456789012:security-control/
S3.17/finding/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "ProductArn": "arn:aws:securityhub:us-east-1::product/aws/
securityhub"
    },
    "UpdateTime": "2023-05-23T02:06:51.518000+00:00",
    "FindingCreated": "true",
    "UpdateSource": {
        "Type": "BATCH_IMPORT_FINDINGS",
        "Identity": "arn:aws:securityhub:us-east-1::product/aws/securityhub"
    },
    "Updates": []
}
]
}

```



有关更多信息，请参阅《Sec AWS urity Hub 用户指南》中的[查找历史记录](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFindingHistory](#)中的。

## get-findings

以下代码示例演示了如何使用 get-findings。

### AWS CLI

示例 1：返回针对特定标准生成的调查结果

以下get-findings示例返回 PCI DSS 标准的结果。

```
aws securityhub get-findings \
  --filters '{"GeneratorId":[{"Value": "pci-dss", "Comparison": "PREFIX"}]}' \
  --max-items 1
```

输出：

```
{
  "Findings": [
    {
      "SchemaVersion": "2018-10-08",
      "Id": "arn:aws:securityhub:eu-central-1:123456789012:subscription/pci-dss/v/3.2.1/PCI.Lambda.2/finding/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "ProductArn": "arn:aws:securityhub:us-west-1::product/aws/securityhub",
      "GeneratorId": "pci-dss/v/3.2.1/PCI.Lambda.2",
      "AwsAccountId": "123456789012",
      "Types": [
        "Software and Configuration Checks/Industry and Regulatory Standards/PCI-DSS"
      ],
      "FindingProviderFields": {
        "Severity": {
          "Original": 0,
          "Label": "INFORMATIONAL"
        },
        "Types": [
          "Software and Configuration Checks/Industry and Regulatory Standards/PCI-DSS"
        ]
      },
    }
  ],
}
```

```

    "FirstObservedAt": "2020-06-02T14:02:49.159Z",
    "LastObservedAt": "2020-06-02T14:02:52.397Z",
    "CreatedAt": "2020-06-02T14:02:49.159Z",
    "UpdatedAt": "2020-06-02T14:02:52.397Z",
    "Severity": {
      "Original": 0,
      "Label": "INFORMATIONAL",
      "Normalized": 0
    },
    "Title": "PCI.Lambda.2 Lambda functions should be in a VPC",
    "Description": "This AWS control checks whether a Lambda function is in
a VPC.",
    "Remediation": {
      "Recommendation": {
        "Text": "For directions on how to fix this issue, please consult
the AWS Security Hub PCI DSS documentation.",
        "Url": "https://docs.aws.amazon.com/console/securityhub/
PCI.Lambda.2/remediation"
      }
    },
    "ProductFields": {
      "StandardsArn": "arn:aws:securityhub::standards/pci-dss/v/3.2.1",
      "StandardsSubscriptionArn": "arn:aws:securityhub:us-
west-1:123456789012:subscription/pci-dss/v/3.2.1",
      "ControlId": "PCI.Lambda.2",
      "RecommendationUrl": "https://docs.aws.amazon.com/console/
securityhub/PCI.Lambda.2/remediation",
      "RelatedAWSResources:0/name": "securityhub-lambda-inside-
vpc-0e904a3b",
      "RelatedAWSResources:0/type": "AWS::Config::ConfigRule",
      "StandardsControlArn": "arn:aws:securityhub:us-
west-1:123456789012:control/pci-dss/v/3.2.1/PCI.Lambda.2",
      "aws/securityhub/SeverityLabel": "INFORMATIONAL",
      "aws/securityhub/ProductName": "Security Hub",
      "aws/securityhub/CompanyName": "AWS",
      "aws/securityhub/FindingId": "arn:aws:securityhub:eu-
central-1::product/aws/securityhub/arn:aws:securityhub:eu-
central-1:123456789012:subscription/pci-dss/v/3.2.1/PCI.Lambda.2/finding/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    },
    "Resources": [
      {
        "Type": "AwsAccount",
        "Id": "AWS:::Account:123456789012",

```

```

        "Partition": "aws",
        "Region": "us-west-1"
      }
    ],
    "Compliance": {
      "Status": "PASSED",
      "RelatedRequirements": [
        "PCI DSS 1.2.1",
        "PCI DSS 1.3.1",
        "PCI DSS 1.3.2",
        "PCI DSS 1.3.4"
      ]
    },
    "WorkflowState": "NEW",
    "Workflow": {
      "Status": "NEW"
    },
    "RecordState": "ARCHIVED"
  }
],
"NextToken": "eyJ0ZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ=="
}

```

示例 2：返回工作流程状态为“已通知”的严重性调查结果

以下`get-findings`示例返回的发现结果的严重性标签值为“关键”，工作流程状态为“已通知”。结果按置信度值降序排序。

```

aws securityhub get-findings \
  --filters '{"SeverityLabel":[{"Value":
"CRITICAL","Comparison":"EQUALS"}],"WorkflowStatus":
[{"Value":"NOTIFIED","Comparison":"EQUALS"}]}' \
  --sort-criteria '{ "Field": "Confidence", "SortOrder": "desc"}' \
  --max-items 1

```

输出：

```

{
  "Findings": [
    {
      "SchemaVersion": "2018-10-08",
      "Id": "arn:aws:securityhub:us-west-1: 123456789012:subscription/cis-aws-
foundations-benchmark/v/1.2.0/1.13/finding/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",

```

```
    "ProductArn": "arn:aws:securityhub:us-west-2::product/aws/securityhub",
    "GeneratorId": "arn:aws:securityhub:::ruleset/cis-aws-foundations-
benchmark/v/1.2.0/rule/1.13",
    "AwsAccountId": "123456789012",
    "Types": [
      "Software and Configuration Checks/Industry and Regulatory
Standards/CIS AWS Foundations Benchmark"
    ],
    "FindingProviderFields" {
      "Severity": {
        "Original": 90,
        "Label": "CRITICAL"
      },
      "Types": [
        "Software and Configuration Checks/Industry and Regulatory
Standards/CIS AWS Foundations Benchmark"
      ]
    },
    "FirstObservedAt": "2020-05-21T20:16:34.752Z",
    "LastObservedAt": "2020-06-09T08:16:37.171Z",
    "CreatedAt": "2020-05-21T20:16:34.752Z",
    "UpdatedAt": "2020-06-09T08:16:36.430Z",
    "Severity": {
      "Original": 90,
      "Label": "CRITICAL",
      "Normalized": 90
    },
    "Title": "1.13 Ensure MFA is enabled for the \"root\" account",
    "Description": "The root account is the most privileged user in an AWS
account. MFA adds an extra layer of protection on top of a user name and password.
With MFA enabled, when a user signs in to an AWS website, they will be prompted for
their user name and password as well as for an authentication code from their AWS
MFA device.",
    "Remediation": {
      "Recommendation": {
        "Text": "For directions on how to fix this issue, please consult
the AWS Security Hub CIS documentation.",
        "Url": "https://docs.aws.amazon.com/console/securityhub/
standards-cis-1.13/remediation"
      }
    },
    "ProductFields": {
      "StandardsGuideArn": "arn:aws:securityhub:::ruleset/cis-aws-
foundations-benchmark/v/1.2.0",
```

```

        "StandardsGuideSubscriptionArn": "arn:aws:securityhub:us-
west-1:123456789012:subscription/cis-aws-foundations-benchmark/v/1.2.0",
        "RuleId": "1.13",
        "RecommendationUrl": "https://docs.aws.amazon.com/console/
securityhub/standards-cis-1.13/remediation",
        "RelatedAWSResources:0/name": "securityhub-root-account-mfa-
enabled-5pftha",
        "RelatedAWSResources:0/type": "AWS::Config::ConfigRule",
        "StandardsControlArn": "arn:aws:securityhub:us-
west-1:123456789012:control/cis-aws-foundations-benchmark/v/1.2.0/1.13",
        "aws/securityhub/SeverityLabel": "CRITICAL",
        "aws/securityhub/ProductName": "Security Hub",
        "aws/securityhub/CompanyName": "AWS",
        "aws/securityhub/FindingId": "arn:aws:securityhub:us-
west-1::product/aws/securityhub/arn:aws:securityhub:us-
west-1:123456789012:subscription/cis-aws-foundations-benchmark/v/1.2.0/1.13/finding/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    },
    "Resources": [
        {
            "Type": "AwsAccount",
            "Id": "AWS:::Account:123456789012",
            "Partition": "aws",
            "Region": "us-west-1"
        }
    ],
    "Compliance": {
        "Status": "FAILED"
    },
    "WorkflowState": "NEW",
    "Workflow": {
        "Status": "NOTIFIED"
    },
    "RecordState": "ACTIVE"
}
]
}

```

有关更多信息，请参阅《Sec AWS urity Hub 用户指南》中的[筛选和分组搜索结果](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFindings](#)中的。

## get-insight-results

以下代码示例演示了如何使用 `get-insight-results`。

### AWS CLI

检索结果以获取见解

以下 `get-insight-results` 示例返回具有指定 ARN 的洞察的洞察结果列表。

```
aws securityhub get-insight-results \
  --insight-arn "arn:aws:securityhub:us-west-1:123456789012:insight/123456789012/
  custom/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

输出：

```
{
  "InsightResults": {
    "GroupByAttribute": "ResourceId",
    "InsightArn": "arn:aws:securityhub:us-
west-1:123456789012:insight/123456789012/custom/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
    "ResultValues": [
      {
        "Count": 10,
        "GroupByAttributeValue": "AWS:::Account:123456789111"
      },
      {
        "Count": 3,
        "GroupByAttributeValue": "AWS:::Account:123456789222"
      }
    ]
  }
}
```

有关更多信息，请参阅《Sec AWS urity Hub 用户指南》中的[查看洞察结果和发现并对其采取行动](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetInsightResults](#) 中的。

## get-insights

以下代码示例演示了如何使用 `get-insights`。

## AWS CLI

### 检索有关洞察的详细信息

以下`get-insights`示例检索具有指定 ARN 的洞察的配置详细信息。

```
aws securityhub get-insights \  
  --insight-arns "arn:aws:securityhub:us-west-1:123456789012:insight/123456789012/  
  custom/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

输出：

```
{  
  "Insights": [  
    {  
      "Filters": {  
        "ResourceType": [  
          {  
            "Comparison": "EQUALS",  
            "Value": "AwsIamRole"  
          }  
        ],  
        "SeverityLabel": [  
          {  
            "Comparison": "EQUALS",  
            "Value": "CRITICAL"  
          }  
        ],  
      },  
      "GroupByAttribute": "ResourceId",  
      "InsightArn": "arn:aws:securityhub:us-  
west-1:123456789012:insight/123456789012/custom/a1b2c3d4-5678-90ab-cdef-  
EXAMPLE11111",  
      "Name": "Critical role findings"  
    }  
  ]  
}
```

有关更多信息，请参阅《[Sec AWS urity Hub 用户指南](#)》中的 [Sec AWS urity Hub 见解](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInsights](#)中的。

## get-invitations-count

以下代码示例演示了如何使用 `get-invitations-count`。

### AWS CLI

检索未被接受的邀请数量

以下`get-invitations-count`示例检索请求账户拒绝或未回复的邀请数量。

```
aws securityhub get-invitations-count
```

输出：

```
{
  "InvitationsCount": 3
}
```

有关更多信息，请参阅 [Sec AWS urity Hub 用户指南](#) 中的 [管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [GetInvitationsCount](#) 中的。

## get-master-account

以下代码示例演示了如何使用 `get-master-account`。

### AWS CLI

检索有关管理员帐户的信息

以下`get-master-account`示例检索有关请求账户的管理员帐户的信息。

```
aws securityhub get-master-account
```

输出：

```
{
  "Master": {
    "AccountId": "123456789012",
    "InvitationId": "7ab938c5d52d7904ad09f9e7c20cc4eb",
    "InvitedAt": "2020-06-01T20:21:18.042000+00:00",
  }
}
```



```
    "MemberStatus": "ASSOCIATED"
  }
}
```

有关更多信息，请参阅 [Sec AWS urity Hub 用户指南](#) 中的 [管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [GetMasterAccount](#) 中的。

## get-members

以下代码示例演示了如何使用 `get-members`。

### AWS CLI

检索有关所选成员账户的信息

以下 `get-members` 示例检索有关指定成员账户的信息。

```
aws securityhub get-members \
  --account-ids "444455556666" "777788889999"
```

输出：

```
{
  "Members": [
    {
      "AccountId": "123456789111",
      "AdministratorId": "123456789012",
      "InvitedAt": "2020-06-01T20:15:15.289000+00:00",
      "MasterId": "123456789012",
      "MemberStatus": "ASSOCIATED",
      "UpdatedAt": "2020-06-01T20:15:15.289000+00:00"
    },
    {
      "AccountId": "123456789222",
      "AdministratorId": "123456789012",
      "InvitedAt": "2020-06-01T20:15:15.289000+00:00",
      "MasterId": "123456789012",
      "MemberStatus": "ASSOCIATED",
      "UpdatedAt": "2020-06-01T20:15:15.289000+00:00"
    }
  ],
}
```

```
"UnprocessedAccounts": [ ]
}
```

有关更多信息，请参阅 [Sec AWS urity Hub 用户指南](#) 中的 [管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [GetMembers](#) 中的。

## get-security-control-definition

以下代码示例演示了如何使用 `get-security-control-definition`。

### AWS CLI

获取安全控制定义的详细信息

以下 `get-security-control-definition` 示例检索 Security Hub 安全控件的定义详细信息。详细信息包括控件标题、描述、区域可用性、参数和其他信息。

```
aws securityhub get-security-control-definition \
  --security-control-id ACM.1
```

输出：

```
{
  "SecurityControlDefinition": {
    "SecurityControlId": "ACM.1",
    "Title": "Imported and ACM-issued certificates should be renewed after a
specified time period",
    "Description": "This control checks whether an AWS Certificate Manager
(ACM) certificate is renewed within the specified time period. It checks both
imported certificates and certificates provided by ACM. The control fails if the
certificate isn't renewed within the specified time period. Unless you provide a
custom parameter value for the renewal period, Security Hub uses a default value of
30 days.",
    "RemediationUrl": "https://docs.aws.amazon.com/console/securityhub/ACM.1/
remediation",
    "SeverityRating": "MEDIUM",
    "CurrentRegionAvailability": "AVAILABLE",
    "ParameterDefinitions": {
      "daysToExpiration": {
        "Description": "Number of days within which the ACM certificate must
be renewed",
```

```
    "ConfigurationOptions": {
      "Integer": {
        "DefaultValue": 30,
        "Min": 14,
        "Max": 365
      }
    }
  }
}
```

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[自定义控件参数](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSecurityControlDefinition](#)中的。

## invite-members

以下代码示例演示了如何使用 invite-members。

### AWS CLI

向成员账户发送邀请

以下invite-members示例向指定的成员账户发送邀请。

```
aws securityhub invite-members \
  --account-ids "123456789111" "123456789222"
```

输出：

```
{
  "UnprocessedAccounts": []
}
```

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[InviteMembers](#)中的。

## list-automation-rules

以下代码示例演示了如何使用 list-automation-rules。

## AWS CLI

### 查看自动化规则列表

以下`list-automation-rules`示例列出了 AWS 账户的自动化规则。只有 Security Hub 管理员帐户可以运行此命令。

```
aws securityhub list-automation-rules \  
  --max-results 3 \  
  --next-token NULL
```

输出：

```
{  
  "AutomationRulesMetadata": [  
    {  
      "RuleArn": "arn:aws:securityhub:us-east-1:123456789012:automation-rule/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "RuleStatus": "ENABLED",  
      "RuleOrder": 1,  
      "RuleName": "Suppress informational findings",  
      "Description": "Suppress GuardDuty findings with Informational  
severity",  
      "IsTerminal": false,  
      "CreatedAt": "2023-05-31T17:56:14.837000+00:00",  
      "UpdatedAt": "2023-05-31T17:59:38.466000+00:00",  
      "CreatedBy": "arn:aws:iam::123456789012:role/Admin"  
    },  
    {  
      "RuleArn": "arn:aws:securityhub:us-east-1:123456789012:automation-rule/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
      "RuleStatus": "ENABLED",  
      "RuleOrder": 1,  
      "RuleName": "sample rule",  
      "Description": "A sample rule",  
      "IsTerminal": false,  
      "CreatedAt": "2023-07-15T23:37:20.223000+00:00",  
      "UpdatedAt": "2023-07-15T23:37:20.223000+00:00",  
      "CreatedBy": "arn:aws:iam::123456789012:role/Admin"  
    },  
    {  
      "RuleArn": "arn:aws:securityhub:us-east-1:123456789012:automation-rule/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
```

```

        "RuleStatus": "ENABLED",
        "RuleOrder": 1,
        "RuleName": "sample rule",
        "Description": "A sample rule",
        "IsTerminal": false,
        "CreatedAt": "2023-07-15T23:45:25.126000+00:00",
        "UpdatedAt": "2023-07-15T23:45:25.126000+00:00",
        "CreatedBy": "arn:aws:iam::123456789012:role/Admin"
    }
]
}

```

有关更多信息，请参阅《Sec AWS urity Hub 用户指南》中的[查看自动化规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAutomationRules](#)中的。

## list-configuration-policies

以下代码示例演示了如何使用 list-configuration-policies。

### AWS CLI

#### 列出配置策略摘要

以下list-configuration-policies示例列出了该组织的配置策略摘要。

```
aws securityhub list-configuration-policies \
  --max-items 3
```

输出：

```

{
  "ConfigurationPolicySummaries": [
    {
      "Arn": "arn:aws:securityhub:eu-central-1:123456789012:configuration-
policy/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "Name": "SampleConfigurationPolicy1",
      "Description": "SampleDescription1",
      "UpdatedAt": "2023-09-26T21:08:36.214000+00:00",
      "ServiceEnabled": true
    },
  ],
}

```

```
{
  "Arn": "arn:aws:securityhub:eu-central-1:123456789012:configuration-
policy/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "Name": "SampleConfigurationPolicy2",
  "Description": "SampleDescription2"
  "UpdatedAt": "2023-11-28T19:26:25.207000+00:00",
  "ServiceEnabled": true
},
{
  "Arn": "arn:aws:securityhub:eu-central-1:123456789012:configuration-
policy/a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "Name": "SampleConfigurationPolicy3",
  "Description": "SampleDescription3",
  "UpdatedAt": "2023-11-28T20:28:04.494000+00:00",
  "ServiceEnabled": true
}
}
```

有关更多信息，请参阅《[Security Hub 用户指南](#)》中的[查看 Sec AWS urity Hub 配置策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListConfigurationPolicies](#)中的。

## list-configuration-policy-associations

以下代码示例演示了如何使用 list-configuration-policy-associations。

### AWS CLI

#### 列出配置关联

以下list-configuration-policy-associations示例列出了该组织的配置关联摘要。响应包括与配置策略和自我管理行为的关联。

```
aws securityhub list-configuration-policy-associations \
  --association-type "APPLIED" \
  --max-items 4
```

#### 输出：

```
{
```

```
"ConfigurationPolicyAssociationSummaries": [
  {
    "ConfigurationPolicyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "TargetId": "r-1ab2",
    "TargetType": "ROOT",
    "AssociationType": "APPLIED",
    "UpdatedAt": "2023-11-28T19:26:49.417000+00:00",
    "AssociationStatus": "FAILED",
    "AssociationStatusMessage": "Policy association failed because 2
organizational units or accounts under this root failed."
  },
  {
    "ConfigurationPolicyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "TargetId": "ou-1ab2-c3de4f5g",
    "TargetType": "ORGANIZATIONAL_UNIT",
    "AssociationType": "APPLIED",
    "UpdatedAt": "2023-09-26T21:14:05.283000+00:00",
    "AssociationStatus": "FAILED",
    "AssociationStatusMessage": "One or more children under this target
failed association."
  },
  {
    "ConfigurationPolicyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "TargetId": "ou-6hi7-8j9kl2m",
    "TargetType": "ORGANIZATIONAL_UNIT",
    "AssociationType": "APPLIED",
    "UpdatedAt": "2023-09-26T21:13:01.816000+00:00",
    "AssociationStatus": "SUCCESS",
    "AssociationStatusMessage": "Association applied successfully on this
target."
  },
  {
    "ConfigurationPolicyId": "SELF_MANAGED_SECURITY_HUB",
    "TargetId": "111122223333",
    "TargetType": "ACCOUNT",
    "AssociationType": "APPLIED",
    "UpdatedAt": "2023-11-28T22:01:26.409000+00:00",
    "AssociationStatus": "SUCCESS"
  }
]
```

有关更多信息，请参阅《[Security Hub 用户指南](#)》中的查看 [Sec AWS urity Hub 配置策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListConfigurationPolicyAssociations](#)中的。

## list-enabled-products-for-import

以下代码示例演示了如何使用 `list-enabled-products-for-import`。

### AWS CLI

返回已启用的产品集成列表

以下 `list-enabled-products-for-import` 示例返回当前启用的产品集成的订阅 ARN 列表。

```
aws securityhub list-enabled-products-for-import
```

输出：

```
{
  "ProductSubscriptions": [ "arn:aws:securityhub:us-west-1:123456789012:product-
subscription/crowdstrike/crowdstrike-falcon", "arn:aws:securityhub:us-
west-1:123456789012:product-subscription/aws/securityhub" ]
}
```

有关更多信息，请参阅 [Sec AWS urity Hub 用户指南中的管理产品集成](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `ListEnabledProductsForImport`](#) 中的。

## list-finding-aggregators

以下代码示例演示了如何使用 `list-finding-aggregators`。

### AWS CLI

列出可用的小部件

以下 `list-finding-aggregators` 示例返回查找结果聚合配置的 ARN。

```
aws securityhub list-finding-aggregators
```

输出：

```
{
  "FindingAggregatorArn": "arn:aws:securityhub:us-east-1:222222222222:finding-
aggregator/123e4567-e89b-12d3-a456-426652340000"
}
```



有关更多信息，请参阅 [《Sec AWS urity Hub 用户指南》中的查看当前查找结果聚合配置](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListFindingAggregators](#)中的。

## list-invitations

以下代码示例演示了如何使用 list-invitations。

### AWS CLI

显示邀请列表

以下list-invitations示例检索发送到请求账户的邀请列表。

```
aws securityhub list-invitations
```

输出：

```
{
  "Invitations": [
    {
      "AccountId": "123456789012",
      "InvitationId": "7ab938c5d52d7904ad09f9e7c20cc4eb",
      "InvitedAt": 2020-06-01T20:21:18.042000+00:00,
      "MemberStatus": "ASSOCIATED"
    }
  ],
}
```

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListInvitations](#)中的。

## list-members

以下代码示例演示了如何使用 list-members。

### AWS CLI

检索成员账户列表

以下list-members示例返回请求的管理员账户的成员账户列表。

```
aws securityhub list-members
```

输出：

```
{
  "Members": [
    {
      "AccountId": "123456789111",
      "AdministratorId": "123456789012",
      "InvitedAt": 2020-06-01T20:15:15.289000+00:00,
      "MasterId": "123456789012",
      "MemberStatus": "ASSOCIATED",
      "UpdatedAt": 2020-06-01T20:15:15.289000+00:00
    },
    {
      "AccountId": "123456789222",
      "AdministratorId": "123456789012",
      "InvitedAt": 2020-06-01T20:15:15.289000+00:00,
      "MasterId": "123456789012",
      "MemberStatus": "ASSOCIATED",
      "UpdatedAt": 2020-06-01T20:15:15.289000+00:00
    }
  ],
}
```

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[管理管理员和成员帐户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListMembers](#)中的。

## list-organization-admin-accounts

以下代码示例演示了如何使用 list-organization-admin-accounts。

### AWS CLI

列出指定的 Security Hub 管理员帐户

以下list-organization-admin-accounts示例列出了组织的 Security Hub 管理员帐户。

```
aws securityhub list-organization-admin-accounts
```

输出：

```
{
  AdminAccounts": [
    { "AccountId": "777788889999" },
    { "Status": "ENABLED" }
  ]
}
```

有关更多信息，请参阅《[Security Hub 用户指南](#)》中的“指定 [Sec AWS urity Hub 管理员帐户](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListOrganizationAdminAccounts](#) 中的。

## list-security-control-definitions

以下代码示例演示了如何使用 list-security-control-definitions。

### AWS CLI

示例 1：列出所有可用的安全控件

以下 list-security-control-definitions 示例列出了所有 Security Hub 标准中可用的安全控制措施。此示例将结果限制为三个控件。

```
aws securityhub list-security-control-definitions \
  --max-items 3
```

输出：

```
{
  "SecurityControlDefinitions": [
    {
      "SecurityControlId": "ACM.1",
      "Title": "Imported and ACM-issued certificates should be renewed after a
specified time period",
      "Description": "This control checks whether an AWS Certificate Manager
(ACM) certificate is renewed within the specified time period. It checks both
imported certificates and certificates provided by ACM. The control fails if the
certificate isn't renewed within the specified time period. Unless you provide a
custom parameter value for the renewal period, Security Hub uses a default value of
30 days.",
      "RemediationUrl": "https://docs.aws.amazon.com/console/securityhub/
ACM.1/remediation",
      "SeverityRating": "MEDIUM",
```

```

    "CurrentRegionAvailability": "AVAILABLE",
    "CustomizableProperties": [
      "Parameters"
    ]
  },
  {
    "SecurityControlId": "ACM.2",
    "Title": "RSA certificates managed by ACM should use a key length of at
least 2,048 bits",
    "Description": "This control checks whether RSA certificates managed by
AWS Certificate Manager use a key length of at least 2,048 bits. The control fails
if the key length is smaller than 2,048 bits.",
    "RemediationUrl": "https://docs.aws.amazon.com/console/securityhub/
ACM.2/remediation",
    "SeverityRating": "HIGH",
    "CurrentRegionAvailability": "AVAILABLE",
    "CustomizableProperties": []
  },
  {
    "SecurityControlId": "APIGateway.1",
    "Title": "API Gateway REST and WebSocket API execution logging should be
enabled",
    "Description": "This control checks whether all stages of an Amazon
API Gateway REST or WebSocket API have logging enabled. The control fails if
the 'loggingLevel' isn't 'ERROR' or 'INFO' for all stages of the API. Unless you
provide custom parameter values to indicate that a specific log type should be
enabled, Security Hub produces a passed finding if the logging level is either
'ERROR' or 'INFO'.",
    "RemediationUrl": "https://docs.aws.amazon.com/console/securityhub/
APIGateway.1/remediation",
    "SeverityRating": "MEDIUM",
    "CurrentRegionAvailability": "AVAILABLE",
    "CustomizableProperties": [
      "Parameters"
    ]
  }
],
  "NextToken": "U2FsdGVkX1/UprCPzxVbkDeHikDXbDxfgJZ1w2RG1XWsFPTMTIQPVE0m/
FduIGxS70bRtAbaUt/8/RCQcg2PU0YXI20hH/Griho0Tgv+Tsm0qvQVFhkJepWmqh
+NyawjocVBeos6xzn/8qnbF9IuwGg=="
}

```

有关更多信息，请参阅《[Sec AWS urity Hub 用户指南](#)》中的[查看标准详情](#)。

## 示例 2：列出特定标准的可用安全控制措施

以下 `list-security-control-definitions` 示例列出了 CIS AWS 基金会基准测试 v1.4.0 的可用安全控制措施。此示例将结果限制为三个控件。

```
aws securityhub list-security-control-definitions \  
  --standards-arn "arn:aws:securityhub:us-east-1::standards/cis-aws-foundations-  
benchmark/v/1.4.0" \  
  --max-items 3
```

输出：

```
{  
  "SecurityControlDefinitions": [  
    {  
      "SecurityControlId": "CloudTrail.1",  
      "Title": "CloudTrail should be enabled and configured with at least one  
multi-Region trail that includes read and write management events",  
      "Description": "This AWS control checks that there is at least one  
multi-region AWS CloudTrail trail includes read and write management events.",  
      "RemediationUrl": "https://docs.aws.amazon.com/console/securityhub/  
CloudTrail.1/remediation",  
      "SeverityRating": "HIGH",  
      "CurrentRegionAvailability": "AVAILABLE",  
      "CustomizableProperties": []  
    },  
    {  
      "SecurityControlId": "CloudTrail.2",  
      "Title": "CloudTrail should have encryption at-rest enabled",  
      "Description": "This AWS control checks whether AWS CloudTrail is  
configured to use the server side encryption (SSE) AWS Key Management Service (AWS  
KMS) customer master key (CMK) encryption. The check will pass if the KmsKeyId is  
defined.",  
      "RemediationUrl": "https://docs.aws.amazon.com/console/securityhub/  
CloudTrail.2/remediation",  
      "SeverityRating": "MEDIUM",  
      "CurrentRegionAvailability": "AVAILABLE",  
      "CustomizableProperties": []  
    },  
    {  
      "SecurityControlId": "CloudTrail.4",  
      "Title": "CloudTrail log file validation should be enabled",
```

```

        "Description": "This AWS control checks whether CloudTrail log file
validation is enabled.",
        "RemediationUrl": "https://docs.aws.amazon.com/console/securityhub/
CloudTrail.4/remediation",
        "SeverityRating": "MEDIUM",
        "CurrentRegionAvailability": "AVAILABLE",
        "CustomizableProperties": []
    }
],
"NextToken": "eyJ0ZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAzfQ=="
}

```

有关更多信息，请参阅《Sec AWS urity Hub 用户指南》中的[查看标准详情](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSecurityControlDefinitions](#)中的。

## list-standards-control-associations

以下代码示例演示了如何使用 list-standards-control-associations。

### AWS CLI

获取每个已启用的标准中控件的启用状态

以下list-standards-control-associations示例列出了每个已启用的标准中的启用状态为CloudTrail.1。

```
aws securityhub list-standards-control-associations \
--security-control-id CloudTrail.1
```

输出：

```

{
  "StandardsControlAssociationSummaries": [
    {
      "StandardsArn": "arn:aws:securityhub:us-east-2::standards/nist-800-53/
v/5.0.0",
      "SecurityControlId": "CloudTrail.1",
      "SecurityControlArn": "arn:aws:securityhub:us-
east-2:123456789012:security-control/CloudTrail.1",
      "AssociationStatus": "ENABLED",
      "RelatedRequirements": [
        "NIST.800-53.r5 AC-2(4)",

```

```

        "NIST.800-53.r5 AC-4(26)",
        "NIST.800-53.r5 AC-6(9)",
        "NIST.800-53.r5 AU-10",
        "NIST.800-53.r5 AU-12",
        "NIST.800-53.r5 AU-2",
        "NIST.800-53.r5 AU-3",
        "NIST.800-53.r5 AU-6(3)",
        "NIST.800-53.r5 AU-6(4)",
        "NIST.800-53.r5 AU-14(1)",
        "NIST.800-53.r5 CA-7",
        "NIST.800-53.r5 SC-7(9)",
        "NIST.800-53.r5 SI-3(8)",
        "NIST.800-53.r5 SI-4(20)",
        "NIST.800-53.r5 SI-7(8)",
        "NIST.800-53.r5 SA-8(22)"
    ],
    "UpdatedAt": "2023-05-15T17:52:21.304000+00:00",
    "StandardsControlTitle": "CloudTrail should be enabled and configured
with at least one multi-Region trail that includes read and write management
events",
    "StandardsControlDescription": "This AWS control checks that there is
at least one multi-region AWS CloudTrail trail includes read and write management
events."
  },
  {
    "StandardsArn": "arn:aws:securityhub::ruleset/cis-aws-foundations-
benchmark/v/1.2.0",
    "SecurityControlId": "CloudTrail.1",
    "SecurityControlArn": "arn:aws:securityhub:us-
east-2:123456789012:security-control/CloudTrail.1",
    "AssociationStatus": "ENABLED",
    "RelatedRequirements": [
      "CIS AWS Foundations 2.1"
    ],
    "UpdatedAt": "2020-02-10T21:22:53.998000+00:00",
    "StandardsControlTitle": "Ensure CloudTrail is enabled in all regions",
    "StandardsControlDescription": "AWS CloudTrail is a web service that
records AWS API calls for your account and delivers log files to you. The recorded
information includes the identity of the API caller, the time of the API call,
the source IP address of the API caller, the request parameters, and the response
elements returned by the AWS service."
  },
  {

```

```

        "StandardsArn": "arn:aws:securityhub:us-east-2::standards/aws-
foundational-security-best-practices/v/1.0.0",
        "SecurityControlId": "CloudTrail.1",
        "SecurityControlArn": "arn:aws:securityhub:us-
east-2:123456789012:security-control/CloudTrail.1",
        "AssociationStatus": "DISABLED",
        "RelatedRequirements": [],
        "UpdatedAt": "2023-05-15T19:31:52.671000+00:00",
        "UpdatedReason": "Alternative compensating controls are in place",
        "StandardsControlTitle": "CloudTrail should be enabled and configured
with at least one multi-Region trail that includes read and write management
events",
        "StandardsControlDescription": "This AWS control checks that there is
at least one multi-region AWS CloudTrail trail includes read and write management
events."
    },
    {
        "StandardsArn": "arn:aws:securityhub:us-east-2::standards/cis-aws-
foundations-benchmark/v/1.4.0",
        "SecurityControlId": "CloudTrail.1",
        "SecurityControlArn": "arn:aws:securityhub:us-
east-2:123456789012:security-control/CloudTrail.1",
        "AssociationStatus": "ENABLED",
        "RelatedRequirements": [
            "CIS AWS Foundations Benchmark v1.4.0/3.1"
        ],
        "UpdatedAt": "2022-11-10T15:40:36.021000+00:00",
        "StandardsControlTitle": "Ensure CloudTrail is enabled in all regions",
        "StandardsControlDescription": "AWS CloudTrail is a web service that
records AWS API calls for your account and delivers log files to you. The recorded
information includes the identity of the API caller, the time of the API call,
the source IP address of the API caller, the request parameters, and the response
elements returned by the AWS service. CloudTrail provides a history of AWS API
calls for an account, including API calls made via the Management Console, SDKs,
command line tools, and higher-level AWS services (such as CloudFormation)."
    }
]
}

```

有关更多信息，请参阅 [Sec AWS urity Hub 用户指南中的启用和禁用特定标准中的控件](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考ListStandardsControlAssociations](#)中的。



## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

检索分配给资源的标签

以下 `list-tags-for-resource` 示例返回分配给指定中心资源的标签。

```
aws securityhub list-tags-for-resource \  
  --resource-arn "arn:aws:securityhub:us-west-1:123456789012:hub/default"
```

输出：

```
{  
  "Tags": {  
    "Department" : "Operations",  
    "Area" : "USMidwest"  
  }  
}
```

有关更多信息，请参阅《AWS CloudFormation 用户指南》中的 [AWS SecurityHub::: Hub](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## start-configuration-policy-association

以下代码示例演示了如何使用 `start-configuration-policy-association`。

### AWS CLI

示例 1：关联配置策略

以下 `start-configuration-policy-association` 示例将指定的配置策略与指定的组织单位相关联。配置可以与目标账户、组织单位或根用户相关联。

```
aws securityhub start-configuration-policy-association \  
  --configuration-policy-identifier "arn:aws:securityhub:eu-  
central-1:123456789012:configuration-policy/a1b2c3d4-5678-90ab-cdef-EXAMPLE33333" \  
  --target '{"OrganizationalUnitId": "ou-6hi7-8j91kl2m"}'
```

输出：

```
{
  "ConfigurationPolicyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
  "TargetId": "ou-6hi7-8j91kl2m",
  "TargetType": "ORGANIZATIONAL_UNIT",
  "AssociationType": "APPLIED",
  "UpdatedAt": "2023-11-29T17:40:52.468000+00:00",
  "AssociationStatus": "PENDING"
}
```

有关更多信息，请参阅《[Security Hub 用户指南](#)》中的[创建和关联 S AWS ecurity Hub 配置策略](#)。

### 示例 2：关联自我管理配置

以下 `start-configuration-policy-association` 示例将自我管理配置与指定账户相关联。

```
aws securityhub start-configuration-policy-association \
  --configuration-policy-identifier "SELF_MANAGED_SECURITY_HUB" \
  --target '{"OrganizationalUnitId": "123456789012"}
```

输出：

```
{
  "ConfigurationPolicyId": "SELF_MANAGED_SECURITY_HUB",
  "TargetId": "123456789012",
  "TargetType": "ACCOUNT",
  "AssociationType": "APPLIED",
  "UpdatedAt": "2023-11-29T17:40:52.468000+00:00",
  "AssociationStatus": "PENDING"
}
```

有关更多信息，请参阅《[Security Hub 用户指南](#)》中的[创建和关联 S AWS ecurity Hub 配置策略](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartConfigurationPolicyAssociation](#) 中的。

## start-configuration-policy-disassociation

以下代码示例演示了如何使用 `start-configuration-policy-disassociation`。

### AWS CLI

#### 示例 1：取消关联配置策略

以下 `start-configuration-policy-disassociation` 示例取消配置策略与指定组织单位的关联。可以取消配置与目标账户、组织单位或根账户的关联。

```
aws securityhub start-configuration-policy-disassociation \  
  --configuration-policy-identifier "arn:aws:securityhub:eu-  
central-1:123456789012:configuration-policy/a1b2c3d4-5678-90ab-cdef-EXAMPLE33333" \  
  --target '{"OrganizationalUnitId": "ou-6hi7-8j91kl2m"}'
```

此命令不生成任何输出。

有关更多信息，请参阅 [Sec AWS urity Hub 用户指南中的取消配置与账户和 OU 的关联](#)。

### 示例 2：取消关联自我管理配置

以下 `start-configuration-policy-disassociation` 示例取消自我管理配置与指定账户的关联。

```
aws securityhub start-configuration-policy-disassociation \  
  --configuration-policy-identifier "SELF_MANAGED_SECURITY_HUB" \  
  --target '{"AccountId": "123456789012"}'
```

此命令不生成任何输出。

有关更多信息，请参阅 [Sec AWS urity Hub 用户指南中的取消配置与账户和 OU 的关联](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `StartConfigurationPolicyDisassociation` 中的](#)。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

### AWS CLI

为资源分配标签

以下 `tag-resource` 示例将“部门”和“区域”标签的值分配给指定的中心资源。

```
aws securityhub tag-resource \  
  --resource-arn "arn:aws:securityhub:us-west-1:123456789012:hub/default" \  
  --tags '{"Department": "Operations", "Area": "USMidwest"}'
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CloudFormation 用户指南》中的[AWS SecurityHub::: Hub](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagResource](#)中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从资源中移除标签值

以下untag-resource示例从指定的中心资源中删除 Department 标签。

```
aws securityhub untag-resource \  
  --resource-arn "arn:aws:securityhub:us-west-1:123456789012:hub/default" \  
  --tag-keys "Department"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS CloudFormation 用户指南》中的[AWS SecurityHub::: Hub](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-action-target

以下代码示例演示了如何使用 update-action-target。

### AWS CLI

更新自定义操作

以下update-action-target示例更新了由指定 ARN 标识的自定义操作的名称。

```
aws securityhub update-action-target \  
  --action-target-arn "arn:aws:securityhub:us-west-1:123456789012:action/custom/  
Remediation" \  
  --name "Send to remediation"
```

此命令不生成任何输出。

有关更多信息，请参阅《[Sec AWS urity Hub 用户指南](#)》中的[创建自定义操作并将其与 CloudWatch 事件规则关联](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateActionTarget](#)中的。

## update-configuration-policy

以下代码示例演示了如何使用 update-configuration-policy。

### AWS CLI

#### 更新配置策略

以下update-configuration-policy示例更新现有配置策略以使用指定的设置。

```
aws securityhub update-configuration-policy \
  --identifier "arn:aws:securityhub:eu-central-1:508236694226:configuration-
policy/09f37766-57d8-4ede-9d33-5d8b0fecf70e" \
  --name "SampleConfigurationPolicyUpdated" \
  --description "SampleDescriptionUpdated" \
  --configuration-policy '{"SecurityHub": {"ServiceEnabled":
true, "EnabledStandardIdentifiers": ["arn:aws:securityhub:eu-
central-1::standards/aws-foundational-security-best-practices/
v/1.0.0", "arn:aws:securityhub::ruleset/cis-aws-foundations-benchmark/
v/1.2.0"], "SecurityControlsConfiguration": {"DisabledSecurityControlIdentifiers":
["CloudWatch.1"], "SecurityControlCustomParameters": [{"SecurityControlId":
"ACM.1", "Parameters": {"daysToExpiration": {"ValueType": "CUSTOM", "Value":
{"Integer": 21}}}]}}}' \
  --updated-reason "Disabling CloudWatch.1 and changing parameter value"
```

输出：

```
{
  "Arn": "arn:aws:securityhub:eu-central-1:123456789012:configuration-policy/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "Name": "SampleConfigurationPolicyUpdated",
  "Description": "SampleDescriptionUpdated",
  "UpdatedAt": "2023-11-28T20:28:04.494000+00:00",
  "CreatedAt": "2023-11-28T20:28:04.494000+00:00",
  "ConfigurationPolicy": {
    "SecurityHub": {
      "ServiceEnabled": true,
```

```

    "EnabledStandardIdentifiers": [
      "arn:aws:securityhub:eu-central-1::standards/aws-foundational-
security-best-practices/v/1.0.0",
      "arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/
v/1.2.0"
    ],
    "SecurityControlsConfiguration": {
      "DisabledSecurityControlIdentifiers": [
        "CloudWatch.1"
      ],
      "SecurityControlCustomParameters": [
        {
          "SecurityControlId": "ACM.1",
          "Parameters": {
            "daysToExpiration": {
              "ValueType": "CUSTOM",
              "Value": {
                "Integer": 21
              }
            }
          }
        }
      ]
    }
  }
}

```

有关更多信息，请参阅《[Security Hub 用户指南](#)》中的[更新 Sec AWS urity Hub 配置策略](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateConfigurationPolicy](#)中的。

## update-finding-aggregator

以下代码示例演示了如何使用 update-finding-aggregator。

### AWS CLI

#### 更新当前的查找结果聚合配置

以下update-finding-aggregator示例将查找结果聚合配置更改为从选定区域进行链接。它从美国东部（弗吉尼亚州）运营，这是聚合区域。它选择美国西部（加利福尼亚北部）和美国西部（俄勒冈）作为关联区域。

```
aws securityhub update-finding-aggregator \  
  --region us-east-1 \  
  --finding-aggregator-arn arn:aws:securityhub:us-east-1:222222222222:finding-  
aggregator/123e4567-e89b-12d3-a456-426652340000 \  
  --region-linking-mode SPECIFIED_REGIONS \  
  --regions us-west-1,us-west-2
```

此命令不生成任何输出。

有关更多信息，请参阅 [《Sec AWS urity Hub 用户指南》中的更新查找聚合配置](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateFindingAggregator](#) 中的。

## update-insight

以下代码示例演示了如何使用 update-insight。

### AWS CLI

示例 1：更改自定义数据分析的筛选条件

以下 update-insight 示例更改了自定义数据分析的筛选条件。更新的见解会查找与 AWS 角色相关的严重性较高的调查结果。

```
aws securityhub update-insight \  
  --insight-arn "arn:aws:securityhub:us-west-1:123456789012:insight/123456789012/  
custom/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" \  
  --filters '{"ResourceType": [{"Comparison": "EQUALS", "Value": "AwsIamRole"}],  
"SeverityLabel": [{"Comparison": "EQUALS", "Value": "HIGH"}]}' \  
  --name "High severity role findings"
```

示例 2：更改自定义数据分析的分组属性

以下 update-insight 示例使用指定 ARN 更改自定义数据分析的分组属性。新的分组属性是资源 ID。

```
aws securityhub update-insight \  
  --insight-arn "arn:aws:securityhub:us-west-1:123456789012:insight/123456789012/  
custom/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" \  
  --group-by-attribute "ResourceId" \  
  --name "Critical role findings"
```

输出：

```
{
  "Insights": [
    {
      "InsightArn": "arn:aws:securityhub:us-
west-1:123456789012:insight/123456789012/custom/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
      "Name": "Critical role findings",
      "Filters": {
        "SeverityLabel": [
          {
            "Value": "CRITICAL",
            "Comparison": "EQUALS"
          }
        ],
        "ResourceType": [
          {
            "Value": "AwsIamRole",
            "Comparison": "EQUALS"
          }
        ]
      },
      "GroupByAttribute": "ResourceId"
    }
  ]
}
```

有关更多信息，请参阅 [Sec AWS urity Hub 用户指南](#) 中的 [管理自定义见解](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [UpdateInsight](#) 中的。

## update-organization-configuration

以下代码示例演示了如何使用 `update-organization-configuration`。

### AWS CLI

更新为组织配置 Security Hub 的方式

以下 `update-organization-configuration` 示例指定 Security Hub 应使用集中配置来配置组织。运行此命令后，委派的 Security Hub 管理员可以创建和管理配置策略来配置组织。委派的管理



员也可以使用此命令从中央配置切换到本地配置。如果配置类型为本地配置，则授权管理员可以选择是否在新组织帐户中自动启用 Security Hub 和默认安全标准。

```
aws securityhub update-organization-configuration \  
  --no-auto-enable \  
  --organization-configuration '{"ConfigurationType": "CENTRAL"}'
```

此命令不生成任何输出。

有关更多信息，请参阅《Sec AWS urity Hub 用户指南 AWS 》中的 [Organizations 账户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateOrganizationConfiguration](#)中的。

## update-security-control

以下代码示例演示了如何使用 update-security-control。

### AWS CLI

#### 更新安全控制属性

以下update-security-control示例为 Security Hub 安全控制参数指定了自定义值。

```
aws securityhub update-security-control \  
  --security-control-id ACM.1 \  
  --parameters '{"daysToExpiration": {"ValueType": "CUSTOM", "Value": {"Integer":  
15}}}' \  
  --last-update-reason "Internal compliance requirement"
```

此命令不生成任何输出。

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的 [自定义控件参数](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateSecurityControl](#)中的。

## update-security-hub-configuration

以下代码示例演示了如何使用 update-security-hub-configuration。

### AWS CLI

#### 更新 Security Hub 配置

以下update-security-hub-configuration示例将 Security Hub 配置为自动为启用的标准启用新控件。

```
aws securityhub update-security-hub-configuration \  
  --auto-enable-controls
```

此命令不生成任何输出。

有关更多信息，请参阅 Sec AWS urity Hub 用户指南中的[自动启用新控件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateSecurityHubConfiguration](#)中的。

## update-standards-control

以下代码示例演示了如何使用 update-standards-control。

### AWS CLI

#### 示例 1：禁用控件

以下update-standards-control示例禁用 PCI。 AutoScaling.1 控制。

```
aws securityhub update-standards-control \  
  --standards-control-arn "arn:aws:securityhub:us-west-1:123456789012:control/pci-  
dss/v/3.2.1/PCI.AutoScaling.1" \  
  --control-status "DISABLED" \  
  --disabled-reason "Not applicable for my service"
```

此命令不生成任何输出。

#### 示例 2：启用控件

以下update-standards-control示例启用 PCI。 AutoScaling.1 控制。

```
aws securityhub update-standards-control \  
  --standards-control-arn "arn:aws:securityhub:us-west-1:123456789012:control/pci-  
dss/v/3.2.1/PCI.AutoScaling.1" \  
  --control-status "ENABLED"
```

此命令不生成任何输出。

有关更多信息，请参阅《Sec AWS urity Hub 用户指南》中的[禁用和启用单个控件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateStandardsControl](#)中的。

## AWS Serverless Application Repository 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Serverless Application Repository。 AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### put-application-policy

以下代码示例演示了如何使用 put-application-policy。

AWS CLI

示例 1：公开共享应用程序

以下内容公开put-application-policy共享应用程序，因此任何人都可以在 AWS Serverless Application Repository 中找到并部署您的应用程序。

```
aws serverlessrepo put-application-policy \  
  --application-id arn:aws:serverlessrepo:us-east-1:123456789012:applications/my-  
test-application \  
  --statements Principals='*',Actions=Deploy
```

输出：

```
{  
  "Statements": [  
    {
```

```

        "Actions": [
            "Deploy"
        ],
        "Principals": [
            ""
        ],
        "StatementId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
    }
]
}

```

## 示例 2：私下共享应用程序

以下内容私下put-application-policy共享应用程序，因此只有特定 AWS 帐户才能在 AWS Serverless Application Repository 中查找和部署您的应用程序。

```

aws serverlessrepo put-application-policy \
  --application-id arn:aws:serverlessrepo:us-east-1:123456789012:applications/my-
  test-application \
  --statements Principals=111111111111,222222222222,Actions=Deploy

```

输出：

```

{
  "Statements": [
    {
      "Actions": [
        "Deploy"
      ],
      "Principals": [
        "111111111111",
        "222222222222"
      ],
      "StatementId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
    }
  ]
}

```

有关更多信息，请参阅 [《AWS Serverless Application Repository 开发者指南》](#) 中的[通过控制台共享应用程序](#)

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutApplicationPolicy](#) 中的。

## 使用 Service Catalog 示例 AWS CLI

以下代码示例向您展示了如何使用 with Service Catalog 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **accept-portfolio-share**

以下代码示例演示了如何使用 accept-portfolio-share。

AWS CLI

接受投资组合份额

以下accept-portfolio-share示例接受其他用户提出的共享指定投资组合的提议。

```
aws servicecatalog accept-portfolio-share \  
  --portfolio-id port-2s6wuabcdefghijk
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AcceptPortfolioShare](#)中的。

### **associate-principal-with-portfolio**

以下代码示例演示了如何使用 associate-principal-with-portfolio。

AWS CLI

将委托人与投资组合关联起来

以下 `associate-principal-with-portfolio` 示例将用户与指定的产品组合相关联。

```
aws servicecatalog associate-principal-with-portfolio \  
  --portfolio-id port-2s6abcdefwdh4 \  
  --principal-arn arn:aws:iam::123456789012:user/usertest \  
  --principal-type IAM
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociatePrincipalWithPortfolio](#) 中的。

## **associate-product-with-portfolio**

以下代码示例演示了如何使用 `associate-product-with-portfolio`。

### AWS CLI

将产品与产品组合关联

以下 `associate-product-with-portfolio` 示例将给定产品与指定的产品组合相关联。

```
aws servicecatalog associate-product-with-portfolio \  
  --product-id prod-3p5abcdef3oyk \  
  --portfolio-id port-2s6abcdef5wdh4
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateProductWithPortfolio](#) 中的。

## **associate-tag-option-with-resource**

以下代码示例演示了如何使用 `associate-tag-option-with-resource`。

### AWS CLI

将 TagOption 与资源关联

以下 `associate-tag-option-with-resource` 示例将指定的 TagOption 与指定的资源相关联。

```
aws servicecatalog associate-tag-option-with-resource \  
  --resource-id port-2s6abcdq5wdh4 \  
  --tag-option-id tag-option-1
```

```
--tag-option-id tag-p3abc2pkpz5qc
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateTagOptionWithResource](#)中的。

## copy-product

以下代码示例演示了如何使用 copy-product。

### AWS CLI

#### 复制产品

以下copy-product示例使用 JSON 文件传递参数，创建指定产品的副本。

```
aws servicecatalog copy-product --cli-input-json file://copy-product-input.json
```

copy-product-input.json 的内容：

```
{
  "SourceProductArn": "arn:aws:catalog:us-west-2:123456789012:product/prod-
tcabcd3syn2xy",
  "TargetProductName": "copy-of-myproduct",
  "CopyOptions": [
    "CopyTags"
  ]
}
```

输出：

```
{
  "CopyProductToken": "copyproduct-abc5defgjkdji"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CopyProduct](#)中的。

## create-portfolio-share

以下代码示例演示了如何使用 create-portfolio-share。

## AWS CLI

### 与账户共享投资组合

以下`create-portfolio-share`示例与指定账户共享指定的投资组合。

```
aws servicecatalog create-portfolio-share \  
  --portfolio-id port-2s6abcdef5wdh4 \  
  --account-id 794123456789
```

此命令不产生任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePortfolioShare](#)中的。

## **create-portfolio**

以下代码示例演示了如何使用 `create-portfolio`。

## AWS CLI

### 创建投资组合

以下`create-portfolio`示例创建了一个投资组合。

```
aws servicecatalog create-portfolio \  
  --provider-name my-provider \  
  --display-name my-portfolio
```

输出：

```
{  
  "PortfolioDetail": {  
    "ProviderName": "my-provider",  
    "DisplayName": "my-portfolio",  
    "CreatedTime": 1571337221.555,  
    "ARN": "arn:aws:catalog:us-east-2:123456789012:portfolio/  
port-2s6xmplq5wdh4",  
    "Id": "port-2s6xmplq5wdh4"  
  }  
}
```



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePortfolio](#)中的。

## create-product

以下代码示例演示了如何使用 create-product。

### AWS CLI

#### 创建产品

以下create-product示例使用 JSON 文件传递参数来创建产品。

```
aws servicecatalog create-product \  
  --cli-input-json file://create-product-input.json
```

create-product-input.json 的内容：

```
{  
  "AcceptLanguage": "en",  
  "Name": "test-product",  
  "Owner": "test-owner",  
  "Description": "test-description",  
  "Distributor": "test-distributor",  
  "SupportDescription": "test-support",  
  "SupportEmail": "test@amazon.com",  
  "SupportUrl": "https://aws.amazon.com",  
  "ProductType": "CLOUD_FORMATION_TEMPLATE",  
  "Tags": [  
    {  
      "Key": "region",  
      "Value": "us-east-1"  
    }  
  ],  
  "ProvisioningArtifactParameters": {  
    "Name": "test-version-name",  
    "Description": "test-version-description",  
    "Info": {  
      "LoadTemplateFromURL": "https://s3-us-west-1.amazonaws.com/  
cloudformation-templates-us-west-1/my-cfn-template.template"  
    },  
    "Type": "CLOUD_FORMATION_TEMPLATE"  
  }  
}
```

```
}
```

输出：

```
{
  "Tags": [
    {
      "Key": "region",
      "Value": "us-east-1"
    }
  ],
  "ProductViewDetail": {
    "CreatedTime": 1576025036.0,
    "ProductARN": "arn:aws:catalog:us-west-2:1234568542028:product/
prod-3p5abcdef3oyk",
    "Status": "CREATED",
    "ProductViewSummary": {
      "Type": "CLOUD_FORMATION_TEMPLATE",
      "Distributor": "test-distributor",
      "SupportUrl": "https://aws.amazon.com",
      "SupportEmail": "test@amazon.com",
      "Id": "prodview-abcd42wvx45um",
      "SupportDescription": "test-support",
      "ShortDescription": "test-description",
      "Owner": "test-owner",
      "Name": "test-product2",
      "HasDefaultPath": false,
      "ProductId": "prod-3p5abcdef3oyk"
    }
  },
  "ProvisioningArtifactDetail": {
    "CreatedTime": 1576025036.0,
    "Active": true,
    "Id": "pa-pq3p5lil12a34",
    "Description": "test-version-description",
    "Name": "test-version-name",
    "Type": "CLOUD_FORMATION_TEMPLATE"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateProduct](#)中的。

## create-provisioning-artifact

以下代码示例演示了如何使用 create-provisioning-artifact。

### AWS CLI

#### 创建置备对象

以下 create-provisioning-artifact 示例使用一个 JSON 文件来传递参数来创建配置对象。

```
aws servicecatalog create-provisioning-artifact \  
  --cli-input-json file://create-provisioning-artifact-input.json
```

create-provisioning-artifact-input.json 的内容：

```
{  
  "ProductId": "prod-nfi2abcdefghi",  
  "Parameters": {  
    "Name": "test-provisioning-artifact",  
    "Description": "test description",  
    "Info": {  
      "LoadTemplateFromURL": "https://s3-us-west-1.amazonaws.com/  
cloudformation-templates-us-west-1/my-cfn-template.template"  
    },  
    "Type": "CLOUD_FORMATION_TEMPLATE"  
  }  
}
```

输出：

```
{  
  "Info": {  
    "TemplateUrl": "https://s3-us-west-1.amazonaws.com/cloudformation-templates-  
us-west-1/my-cfn-template.template"  
  },  
  "Status": "CREATING",  
  "ProvisioningArtifactDetail": {  
    "Id": "pa-bb4abcdefwnaio",  
    "Name": "test-provisioning-artifact",  
    "Description": "test description",  
    "Active": true,  
    "Type": "CLOUD_FORMATION_TEMPLATE",  
    "CreatedTime": 1576022545.0  
  }  
}
```

```
}  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateProvisioningArtifact](#)中的。

## create-tag-option

以下代码示例演示了如何使用 create-tag-option。

### AWS CLI

要创建 TagOption

以下create-tag-option示例创建了一个 TagOption。

```
aws servicecatalog create-tag-option  
  --key 1234  
  --value name
```

输出：

```
{  
  "TagOptionDetail": {  
    "Id": "tag-iabcdn4fzjjms",  
    "Value": "name",  
    "Active": true,  
    "Key": "1234"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTagOption](#)中的。

## delete-portfolio-share

以下代码示例演示了如何使用 delete-portfolio-share。

### AWS CLI

停止与账户共享投资组合

以下delete-portfolio-share示例停止与指定账户共享投资组合。

```
aws servicecatalog delete-portfolio-share \  
  --portfolio-id port-2s6abcdq5wdh4 \  
  --account-id 123456789012
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePortfolioShare](#)中的。

## delete-portfolio

以下代码示例演示了如何使用 delete-portfolio。

### AWS CLI

#### 删除投资组合

以下delete-portfolio示例删除了指定的投资组合。

```
aws servicecatalog delete-portfolio \  
  --id port-abcdlx4gox4do
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePortfolio](#)中的。

## delete-product

以下代码示例演示了如何使用 delete-product。

### AWS CLI

#### 删除产品

以下delete-product示例删除了指定的产品。

```
aws servicecatalog delete-product \  
  --id prod-abcdcek6yhbxi
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteProduct](#)中的。

## delete-provisioning-artifact

以下代码示例演示了如何使用 delete-provisioning-artifact。

### AWS CLI

删除置备对象

以下delete-provisioning-artifact示例删除了指定的置备对象。

```
aws servicecatalog delete-provisioning-artifact \  
  --product-id prod-abc2uebuplcpw \  
  --provisioning-artifact-id pa-pqabcddii7ouc
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteProvisioningArtifact](#)中的。

## delete-tag-option

以下代码示例演示了如何使用 delete-tag-option。

### AWS CLI

要删除 TagOption

以下delete-tag-option示例删除指定的 TagOption。

```
aws servicecatalog delete-tag-option \  
  --id tag-iabcdn4fzjjms
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTagOption](#)中的。

## describe-copy-product-status

以下代码示例演示了如何使用 describe-copy-product-status。

### AWS CLI

描述复制产品操作的状态

以下describe-copy-product-status示例显示了指定异步复制产品操作的当前状态。

```
aws servicecatalog describe-copy-product-status \  
  --copy-product-token copyproduct-znn5tf5abcd3w
```

输出：

```
{  
  "CopyProductStatus": "SUCCEEDED",  
  "TargetProductId": "prod-os6hog7abcdt2"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCopyProductStatus](#)中的。

## describe-portfolio

以下代码示例演示了如何使用 describe-portfolio。

### AWS CLI

描述投资组合

以下describe-portfolio示例显示了指定投资组合的详细信息。

```
aws servicecatalog describe-portfolio \  
  --id port-2s6abcdq5wdh4
```

输出：

```
{  
  "TagOptions": [],  
  "PortfolioDetail": {  
    "ARN": "arn:aws:catalog:us-west-2:687558541234:portfolio/  
port-2s6abcdq5wdh4",  
    "Id": "port-2s6wuzyzq5wdh4",  
    "CreatedTime": 1571337221.555,  
    "DisplayName": "my-portfolio",  
    "ProviderName": "my-provider"  
  },  
  "Tags": []  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePortfolio](#)中的。

## describe-product-as-admin

以下代码示例演示了如何使用 describe-product-as-admin。

### AWS CLI

以管理员身份描述产品

以下describe-product-as-admin示例使用管理员权限显示指定产品的详细信息。

```
aws servicecatalog describe-product-as-admin \  
  --id prod-abcdcek6yhbx1
```

输出：

```
{  
  "TagOptions": [],  
  "ProductViewDetail": {  
    "ProductARN": "arn:aws:catalog:us-west-2:687558542028:product/prod-  
abcdcek6yhbx1",  
    "ProductViewSummary": {  
      "SupportEmail": "test@amazon.com",  
      "Type": "CLOUD_FORMATION_TEMPLATE",  
      "Distributor": "test-distributor",  
      "ShortDescription": "test-description",  
      "Owner": "test-owner",  
      "Id": "prodview-wi3l2j4abc6vc",  
      "SupportDescription": "test-support",  
      "ProductId": "prod-abcdcek6yhbx1",  
      "HasDefaultPath": false,  
      "Name": "test-product3",  
      "SupportUrl": "https://aws.amazon.com"  
    },  
    "CreatedTime": 1577136715.0,  
    "Status": "CREATED"  
  },  
  "ProvisioningArtifactSummaries": [  
    {  
      "CreatedTime": 1577136715.0,  
      "Description": "test-version-description",  
      "ProvisioningArtifactMetadata": {
```



```

        "SourceProvisioningArtifactId": "pa-abcdxkkiv5fcm"
    },
    "Name": "test-version-name-3",
    "Id": "pa-abcdxkkiv5fcm"
  }
],
"Tags": [
  {
    "Value": "iad",
    "Key": "region"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeProductAsAdmin](#)中的。

## describe-provisioned-product

以下代码示例演示了如何使用 describe-provisioned-product。

### AWS CLI

#### 描述预配置的产品

以下describe-provisioned-product示例显示了指定预配置产品的详细信息。

```
aws servicecatalog describe-provisioned-product \
  --id pp-dpom27bm4abcd
```

#### 输出：

```
{
  "ProvisionedProductDetail": {
    "Status": "ERROR",
    "CreatedTime": 1577222793.358,
    "Arn": "arn:aws:servicecatalog:us-west-2:123456789012:stack/mytestppname3/pp-dpom27bm4abcd",
    "Id": "pp-dpom27bm4abcd",
    "StatusMessage": "AmazonCloudFormationException Parameters: [KeyName] must have values (Service: AmazonCloudFormation; Status Code: 400; Error Code: ValidationError; Request ID: 5528602a-a9ef-427c-825c-f82c31b814f5)",
    "IdempotencyToken": "527c5358-2a1a-4b9e-b1b9-7293b0ddff42",
  }
}
```

```
    "LastRecordId": "rec-tfuawdjovzxge",
    "Type": "CFN_STACK",
    "Name": "mytestppname3"
  },
  "CloudWatchDashboards": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeProvisionedProduct](#)中的。

## describe-provisioning-artifact

以下代码示例演示了如何使用 `describe-provisioning-artifact`。

### AWS CLI

#### 描述置备工件

以下`describe-provisioning-artifact`示例显示了指定置备对象的详细信息。

```
aws servicecatalog describe-provisioning-artifact \
  --provisioning-artifact-id pa-pcz347abcdcfm \
  --product-id prod-abcdfz3syn2rg
```

#### 输出：

```
{
  "Info": {
    "TemplateUrl": "https://awsdocs.s3.amazonaws.com/servicecatalog/
myexampledevelopment-environment.template"
  },
  "ProvisioningArtifactDetail": {
    "Id": "pa-pcz347abcdcfm",
    "Active": true,
    "Type": "CLOUD_FORMATION_TEMPLATE",
    "Description": "updated description",
    "CreatedTime": 1562097906.0,
    "Name": "updated name"
  },
  "Status": "AVAILABLE"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeProvisioningArtifact](#)中的。

## describe-tag-option

以下代码示例演示了如何使用 describe-tag-option。

### AWS CLI

描述一个 TagOption

以下describe-tag-option示例显示了指定项的详细信息 TagOption。

```
aws servicecatalog describe-tag-option \  
  --id tag-p3tej2abcd5qc
```

输出：

```
{  
  "TagOptionDetail": {  
    "Active": true,  
    "Id": "tag-p3tej2abcd5qc",  
    "Value": "value-3",  
    "Key": "1234"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTagOption](#)中的。

## disassociate-principal-from-portfolio

以下代码示例演示了如何使用 disassociate-principal-from-portfolio。

### AWS CLI

解除本金与投资组合的关联

以下disassociate-principal-from-portfolio示例取消了指定本金与投资组合的关联。

```
aws servicecatalog disassociate-principal-from-portfolio \  
  --portfolio-id port-2s6abcdq5wdh4 \  
  --principal-arn arn:aws:iam::123456789012:group/myendusers
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociatePrincipalFromPortfolio](#)中的。

## disassociate-product-from-portfolio

以下代码示例演示了如何使用 `disassociate-product-from-portfolio`。

### AWS CLI

取消产品与产品组合的关联

以下`disassociate-product-from-portfolio`示例取消指定产品与产品组合的关联。

```
aws servicecatalog disassociate-product-from-portfolio \  
  --product-id prod-3p5abcdmu3oyk \  
  --portfolio-id port-2s6abcdq5wdh4
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateProductFromPortfolio](#)中的。

## disassociate-tag-option-from-resource

以下代码示例演示了如何使用 `disassociate-tag-option-from-resource`。

### AWS CLI

取消与资源的 TagOption 关联

以下`disassociate-tag-option-from-resource`示例取消指定TagOption与资源的关联。

```
aws servicecatalog disassociate-tag-option-from-resource \  
  --resource-id port-2s6abcdq5wdh4 \  
  --tag-option-id tag-p3abc2pkpz5qc
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateTagOptionFromResource](#)中的。

## list-accepted-portfolio-shares

以下代码示例演示了如何使用 `list-accepted-portfolio-shares`。

## AWS CLI

### 列出已接受的投资组合股份

以下 `list-accepted-portfolio-shares` 示例列出了该账户接受共享的所有产品组合，仅包括默认的 Service Catalog 产品组合。

```
aws servicecatalog list-accepted-portfolio-shares \  
  --portfolio-share-type "AWS_SERVICECATALOG"
```

输出：

```
{  
  "PortfolioDetails": [  
    {  
      "ARN": "arn:aws:catalog:us-west-2:123456789012:portfolio/port-  
d2abcd5dpkuma",  
      "Description": "AWS Service Catalog Reference blueprints for often-used  
AWS services such as EC2, S3, RDS, VPC and EMR.",  
      "CreatedTime": 1574456190.687,  
      "ProviderName": "AWS Service Catalog",  
      "DisplayName": "Reference Architectures",  
      "Id": "port-d2abcd5dpkuma"  
    },  
    {  
      "ARN": "arn:aws:catalog:us-west-2:123456789012:portfolio/port-  
abcdefaua7zpu",  
      "Description": "AWS well-architected blueprints for high reliability  
applications.",  
      "CreatedTime": 1574461496.092,  
      "ProviderName": "AWS Service Catalog",  
      "DisplayName": "High Reliability Architectures",  
      "Id": "port-abcdefaua7zpu"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAcceptedPortfolioShares](#) 中的。

### **list-portfolio-access**

以下代码示例演示了如何使用 `list-portfolio-access`。

## AWS CLI

列出有权访问投资组合的账户

以下`list-portfolio-access`示例列出了有权访问指定投资组合的 AWS 账户。

```
aws servicecatalog list-portfolio-access \  
  --portfolio-id port-2s6abcdq5wdh4
```

输出：

```
{  
  "AccountIds": [  
    "123456789012"  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPortfolioAccess](#)中的。

## `list-portfolios-for-product`

以下代码示例演示了如何使用 `list-portfolios-for-product`。

### AWS CLI

列出与产品关联的产品组合

以下`list-portfolios-for-product`示例列出了与指定产品关联的产品组合。

```
aws servicecatalog list-portfolios-for-product \  
  --product-id prod-abcdefz3syn2rg
```

输出：

```
{  
  "PortfolioDetails": [  
    {  
      "CreatedTime": 1571337221.555,  
      "Id": "port-2s6abcdq5wdh4",  
      "ARN": "arn:aws:catalog:us-west-2:123456789012:portfolio/  
port-2s6abcdq5wdh4",
```

```
        "DisplayName": "my-portfolio",
        "ProviderName": "my-provider"
    },
    {
        "CreatedTime": 1559665256.348,
        "Id": "port-5abcd3e5st4ei",
        "ARN": "arn:aws:catalog:us-west-2:123456789012:portfolio/
port-5abcd3e5st4ei",
        "DisplayName": "test",
        "ProviderName": "provider-name"
    }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPortfoliosForProduct](#)中的。

## list-portfolios

以下代码示例演示了如何使用 list-portfolios。

### AWS CLI

列出投资组合

以下list-portfolios示例列出了当前区域的 Service Catalog 产品组合。

```
aws servicecatalog list-portfolios
```

输出：

```
{
  "PortfolioDetails": [
    {
      "CreatedTime": 1559665256.348,
      "ARN": "arn:aws:catalog:us-east-2:123456789012:portfolio/
port-5pzcxmlst4ei",
      "DisplayName": "my-portfolio",
      "Id": "port-5pzcxmlst4ei",
      "ProviderName": "my-user"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPortfolios](#)中的。

## list-principals-for-portfolio

以下代码示例演示了如何使用 `list-principals-for-portfolio`。

### AWS CLI

列出投资组合的所有委托人

以下`list-principals-for-portfolio`示例列出了指定投资组合的所有委托人。

```
aws servicecatalog list-principals-for-portfolio \
  --portfolio-id port-2s6abcdq5wdh4
```

输出：

```
{
  "Principals": [
    {
      "PrincipalARN": "arn:aws:iam::123456789012:user/usertest",
      "PrincipalType": "IAM"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPrincipalsForPortfolio](#)中的。

## list-provisioning-artifacts

以下代码示例演示了如何使用 `list-provisioning-artifacts`。

### AWS CLI

列出产品的所有配置对象

以下`list-provisioning-artifacts`示例列出了指定产品的所有配置对象。

```
aws servicecatalog list-provisioning-artifacts \
  --product-id prod-nfi2abcdefgcpw
```



输出：

```
{
  "ProvisioningArtifactDetails": [
    {
      "Id": "pa-abcdef54ipm6z",
      "Description": "test-version-description",
      "Type": "CLOUD_FORMATION_TEMPLATE",
      "CreatedTime": 1576021147.0,
      "Active": true,
      "Name": "test-version-name"
    },
    {
      "Id": "pa-bb4zyxwwnaio",
      "Description": "test description",
      "Type": "CLOUD_FORMATION_TEMPLATE",
      "CreatedTime": 1576022545.0,
      "Active": true,
      "Name": "test-provisioning-artifact-2"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListProvisioningArtifacts](#)中的。

## list-resources-for-tag-option

以下代码示例演示了如何使用 list-resources-for-tag-option。

AWS CLI

列出与关联的资源 TagOption

以下list-resources-for-tag-option示例列出了与指定关联的资源TagOption。

```
aws servicecatalog list-resources-for-tag-option \
  --tag-option-id tag-p3tej2abcd5qc
```

输出：

```
{
```

```
    "ResourceDetails": [
      {
        "ARN": "arn:aws:catalog:us-west-2:123456789012:product/prod-
abcdfz3syn2rg",
        "Name": "my product",
        "Description": "description",
        "CreatedTime": 1562097906.0,
        "Id": "prod-abcdfz3syn2rg"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResourcesForTagOption](#)中的。

## list-tag-options

以下代码示例演示了如何使用 list-tag-options。

### AWS CLI

以下list-tag-options示例列出了的所有值TagOptions。

```
aws servicecatalog list-tag-options
```

输出：

```
{
  "TagOptionDetails": [
    {
      "Value": "newvalue",
      "Active": true,
      "Id": "tag-iabcdn4fzjjms",
      "Key": "1234"
    },
    {
      "Value": "value1",
      "Active": true,
      "Id": "tag-e3abcdvmwvrzy",
      "Key": "key"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagOptions](#)中的。

## provision-product

以下代码示例演示了如何使用 provision-product。

### AWS CLI

#### 配置产品

以下provision-product示例使用指定的置备工件置备指定的产品。

```
aws servicecatalog provision-product \  
  --product-id prod-abcdefz3syn2rg \  
  --provisioning-artifact-id pa-abc347pcscfm \  
  --provisioned-product-name "mytestppname3"
```

输出：

```
{  
  "RecordDetail": {  
    "RecordId": "rec-tfuawdabcdege",  
    "CreatedTime": 1577222793.362,  
    "ProvisionedProductId": "pp-abcd27bm4mldq",  
    "PathId": "lpv2-abcdg3jp6t5k6",  
    "RecordErrors": [],  
    "ProductId": "prod-abcdefz3syn2rg",  
    "UpdatedTime": 1577222793.362,  
    "RecordType": "PROVISION_PRODUCT",  
    "ProvisionedProductName": "mytestppname3",  
    "ProvisioningArtifactId": "pa-pcz347abcdcfm",  
    "RecordTags": [],  
    "Status": "CREATED",  
    "ProvisionedProductType": "CFN_STACK"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ProvisionProduct](#)中的。

## reject-portfolio-share

以下代码示例演示了如何使用 reject-portfolio-share。

## AWS CLI

### 拒绝投资组合份额

以下reject-portfolio-share示例拒绝给定投资组合的投资组合份额。

```
aws servicecatalog reject-portfolio-share \
  --portfolio-id port-2s6wuabcdefghijk
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RejectPortfolioShare](#)中的。

## scan-provisioned-products

以下代码示例演示了如何使用 scan-provisioned-products。

## AWS CLI

### 列出所有可用的预配置产品

以下scan-provisioned-products示例列出了可用的预配置产品。

```
aws servicecatalog scan-provisioned-products
```

输出：

```
{
  "ProvisionedProducts": [
    {
      "Status": "ERROR",
      "Arn": "arn:aws:servicecatalog:us-west-2:123456789012:stack/
mytestppname3/pp-abcd27bm4mldq",
      "StatusMessage": "AmazonCloudFormationException Parameters: [KeyName]
must have values (Service: AmazonCloudFormation; Status Code: 400; Error Code:
Validation error; Request ID: 5528602a-a9ef-427c-825c-f82c31b814f5)",
      "Id": "pp-abcd27bm4mldq",
      "Type": "CFN_STACK",
      "IdempotencyToken": "527c5358-2a1a-4b9e-b1b9-7293b0ddff42",
      "CreatedTime": 1577222793.358,
      "Name": "mytestppname3",
      "LastRecordId": "rec-tfuawdabcdxge"
    }
  ]
}
```

```
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ScanProvisionedProducts](#)中的。

## search-products-as-admin

以下代码示例演示了如何使用 search-products-as-admin。

### AWS CLI

#### 使用管理员权限搜索产品

以下search-products-as-admin示例使用产品组合 ID 作为筛选条件搜索具有管理员权限的产品。

```
aws servicecatalog search-products-as-admin \
  --portfolio-id port-5abcd3e5st4ei
```

输出：

```
{
  "ProductViewDetails": [
    {
      "ProductViewSummary": {
        "Name": "my product",
        "Owner": "owner name",
        "Type": "CLOUD_FORMATION_TEMPLATE",
        "ProductId": "prod-abcdefz3syn2rg",
        "HasDefaultPath": false,
        "Id": "prodview-abcdmyuzv2dlu",
        "ShortDescription": "description"
      },
      "ProductARN": "arn:aws:catalog:us-west-2:123456789012:product/prod-abcdefz3syn2rg",
      "CreatedTime": 1562097906.0,
      "Status": "CREATED"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SearchProductsAsAdmin](#)中的。

## search-provisioned-products

以下代码示例演示了如何使用 search-provisioned-products。

### AWS CLI

#### 搜索预配置的产品

以下 search-provisioned-products 示例使用 JSON 文件传递参数，搜索与指定产品 ID 匹配的预配置产品。

```
aws servicecatalog search-provisioned-products \  
  --cli-input-json file://search-provisioned-products-input.json
```

search-provisioned-products-input.json 的内容：

```
{  
  "Filters": {  
    "SearchQuery": [  
      "prod-tcjevz3syn2rg"  
    ]  
  }  
}
```

输出：

```
{  
  "ProvisionedProducts": [  
    {  
      "ProvisioningArtifactId": "pa-pcz347abcdcfm",  
      "Name": "mytestppname3",  
      "CreatedTime": 1577222793.358,  
      "Id": "pp-abcd27bm4mldq",  
      "Status": "ERROR",  
      "UserArn": "arn:aws:iam::123456789012:user/cliuser",  
      "StatusMessage": "AmazonCloudFormationException Parameters: [KeyName]  
must have values (Service: AmazonCloudFormation; Status Code: 400; Error Code:  
ValidationError; Request ID: 5528602a-a9ef-427c-825c-f82c31b814f5)",  
      "Arn": "arn:aws:servicecatalog:us-west-2:123456789012:stack/  
mytestppname3/pp-abcd27bm4mldq",  
      "Tags": [  

```

```

        {
            "Value": "arn:aws:catalog:us-west-2:123456789012:product/prod-
abcdfz3syn2rg",
            "Key": "aws:servicecatalog:productArn"
        },
        {
            "Value": "arn:aws:iam::123456789012:user/cliuser",
            "Key": "aws:servicecatalog:provisioningPrincipalArn"
        },
        {
            "Value": "value-3",
            "Key": "1234"
        },
        {
            "Value": "pa-pcz347abcdcfm",
            "Key": "aws:servicecatalog:provisioningArtifactIdentifier"
        },
        {
            "Value": "arn:aws:catalog:us-west-2:123456789012:portfolio/
port-2s6abcdq5wdh4",
            "Key": "aws:servicecatalog:portfolioArn"
        },
        {
            "Value": "arn:aws:servicecatalog:us-west-2:123456789012:stack/
mytestppname3/pp-abcd27bm4mldq",
            "Key": "aws:servicecatalog:provisionedProductArn"
        }
    ],
    "IdempotencyToken": "527c5358-2a1a-4b9e-b1b9-7293b0ddff42",
    "UserArnSession": "arn:aws:iam::123456789012:user/cliuser",
    "Type": "CFN_STACK",
    "LastRecordId": "rec-tfuawdabcdxge",
    "ProductId": "prod-abcdfz3syn2rg"
}
],
"TotalResultsCount": 1
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SearchProvisionedProducts](#)中的。

## update-portfolio

以下代码示例演示了如何使用 update-portfolio。

## AWS CLI

### 更新投资组合

以下update-portfolio示例更新了指定投资组合的名称。

```
aws servicecatalog update-portfolio \  
  --id port-5abcd3e5st4ei \  
  --display-name "New portfolio name"
```

输出：

```
{  
  "PortfolioDetail": {  
    "DisplayName": "New portfolio name",  
    "ProviderName": "provider",  
    "ARN": "arn:aws:catalog:us-west-2:123456789012:portfolio/  
port-5abcd3e5st4ei",  
    "Id": "port-5abcd3e5st4ei",  
    "CreatedTime": 1559665256.348  
  },  
  "Tags": []  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePortfolio](#)中的。

### update-product

以下代码示例演示了如何使用 update-product。

## AWS CLI

### 更新产品

以下update-product示例更新了指定产品的名称和所有者。

```
aws servicecatalog update-product \  
  --id prod-os6abc7drqlt2 \  
  --name "New product name" \  
  --owner "Updated product owner"
```



输出：

```
{
  "Tags": [
    {
      "Value": "iad",
      "Key": "region"
    }
  ],
  "ProductViewDetail": {
    "ProductViewSummary": {
      "Owner": "Updated product owner",
      "ProductId": "prod-os6abc7drqlt2",
      "Distributor": "test-distributor",
      "SupportUrl": "https://aws.amazon.com",
      "Name": "New product name",
      "ShortDescription": "test-description",
      "HasDefaultPath": false,
      "Id": "prodview-6abcdgrfhvidy",
      "SupportDescription": "test-support",
      "SupportEmail": "test@amazon.com",
      "Type": "CLOUD_FORMATION_TEMPLATE"
    },
    "Status": "CREATED",
    "ProductARN": "arn:aws:catalog:us-west-2:123456789012:product/prod-os6abc7drqlt2",
    "CreatedTime": 1577136255.0
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateProduct](#)中的。

## update-provisioning-artifact

以下代码示例演示了如何使用 update-provisioning-artifact。

### AWS CLI

#### 更新置备构件

以下update-provisioning-artifact示例使用 JSON 文件传递参数，更新了指定置备工件的名称和描述。

```
aws servicecatalog update-provisioning-artifact \  
  --cli-input-json file://update-provisioning-artifact-input.json
```

update-provisioning-artifact-input.json 的内容：

```
{  
  "ProductId": "prod-abcdefz3syn2rg",  
  "ProvisioningArtifactId": "pa-pcz347abcdcfm",  
  "Name": "updated name",  
  "Description": "updated description"  
}
```

输出：

```
{  
  "Info": {  
    "TemplateUrl": "https://awsdocs.s3.amazonaws.com/servicecatalog/  
myexampledevelopment-environment.template"  
  },  
  "Status": "AVAILABLE",  
  "ProvisioningArtifactDetail": {  
    "Active": true,  
    "Description": "updated description",  
    "Id": "pa-pcz347abcdcfm",  
    "Name": "updated name",  
    "Type": "CLOUD_FORMATION_TEMPLATE",  
    "CreatedTime": 1562097906.0  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateProvisioningArtifact](#)中的。

## update-tag-option

以下代码示例演示了如何使用 update-tag-option。

### AWS CLI

#### 要更新 TagOption

以下update-tag-option示例使用指定的 JSON 文件更新 a TagOption 的值。

```
aws servicecatalog update-tag-option --cli-input-json file://update-tag-option-input.json
```

update-tag-option-input.json 的内容：

```
{
  "Id": "tag-iabcdn4fzjjms",
  "Value": "newvalue",
  "Active": true
}
```

输出：

```
{
  "TagOptionDetail": {
    "Value": "newvalue",
    "Key": "1234",
    "Active": true,
    "Id": "tag-iabcdn4fzjjms"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateTagOption](#)中的。

## 使用 Service Quotas 示例 AWS CLI

以下代码示例向您展示了如何使用 with Service Quotas 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

## 操作

### get-aws-default-service-quota

以下代码示例演示了如何使用 `get-aws-default-service-quota`。

#### AWS CLI

##### 描述默认服务配额

以下 `get-aws-default-service-quota` 示例显示了指定配额的详细信息。

```
aws service-quotas get-aws-default-service-quota \  
  --service-code ec2 \  
  --quota-code L-1216C47A
```

##### 输出：

```
{  
  "Quota": {  
    "ServiceCode": "ec2",  
    "ServiceName": "Amazon Elastic Compute Cloud (Amazon EC2)",  
    "QuotaArn": "arn:aws:servicequotas:us-east-2::ec2/L-1216C47A",  
    "QuotaCode": "L-1216C47A",  
    "QuotaName": "Running On-Demand Standard (A, C, D, H, I, M, R, T, Z)  
instances",  
    "Value": 5.0,  
    "Unit": "None",  
    "Adjustable": true,  
    "GlobalQuota": false,  
    "UsageMetric": {  
      "MetricNamespace": "AWS/Usage",  
      "MetricName": "ResourceCount",  
      "MetricDimensions": {  
        "Class": "Standard/OnDemand",  
        "Resource": "vCPU",  
        "Service": "EC2",  
        "Type": "Resource"  
      },  
      "MetricStatisticRecommendation": "Maximum"  
    }  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetAwsDefaultServiceQuota](#)中的。

## get-requested-service-quota-change

以下代码示例演示了如何使用 `get-requested-service-quota-change`。

### AWS CLI

描述增加服务配额的请求

以下`get-requested-service-quota-change`示例描述了指定的增加配额的请求。

```
aws service-quotas get-requested-service-quota-change \  
  --request-id d187537d15254312a9609aa51bbf7624u7W49tP0
```

输出：

```
{  
  "RequestedQuota": {  
    "Id": "d187537d15254312a9609aa51bbf7624u7W49tP0",  
    "CaseId": "6780195351",  
    "ServiceCode": "ec2",  
    "ServiceName": "Amazon Elastic Compute Cloud (Amazon EC2)",  
    "QuotaCode": "L-20F13EBD",  
    "QuotaName": "Running Dedicated c5n Hosts",  
    "DesiredValue": 2.0,  
    "Status": "CASE_OPENED",  
    "Created": 1580446904.067,  
    "LastUpdated": 1580446953.265,  
    "Requester": "{\"accountId\":\"123456789012\",\"callerArn\":  
  \"arn:aws:iam::123456789012:root\"}",  
    "QuotaArn": "arn:aws:servicequotas:us-east-2:123456789012:ec2/L-20F13EBD",  
    "GlobalQuota": false,  
    "Unit": "None"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRequestedServiceQuotaChange](#)中的。

## get-service-quota

以下代码示例演示了如何使用 `get-service-quota`。

## AWS CLI

### 描述服务配额

以下`get-service-quota`示例显示有关指定配额的详细信息。

```
aws service-quotas get-service-quota \  
  --service-code ec2 \  
  --quota-code L-1216C47A
```

输出：

```
{  
  "Quota": {  
    "ServiceCode": "ec2",  
    "ServiceName": "Amazon Elastic Compute Cloud (Amazon EC2)",  
    "QuotaArn": "arn:aws:servicequotas:us-east-2:123456789012:ec2/L-1216C47A",  
    "QuotaCode": "L-1216C47A",  
    "QuotaName": "Running On-Demand Standard (A, C, D, H, I, M, R, T, Z)  
instances",  
    "Value": 1920.0,  
    "Unit": "None",  
    "Adjustable": true,  
    "GlobalQuota": false,  
    "UsageMetric": {  
      "MetricNamespace": "AWS/Usage",  
      "MetricName": "ResourceCount",  
      "MetricDimensions": {  
        "Class": "Standard/OnDemand",  
        "Resource": "vCPU",  
        "Service": "EC2",  
        "Type": "Resource"  
      },  
      "MetricStatisticRecommendation": "Maximum"  
    },  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetServiceQuota](#)中的。

## list-aws-default-service-quotas

以下代码示例演示了如何使用 `list-aws-default-service-quotas`。

### AWS CLI

列出服务的默认配额

以下 `list-aws-default-service-quotas` 示例列出了指定服务配额的默认值。

```
aws service-quotas list-aws-default-service-quotas \
  --service-code xray
```

输出：

```
{
  "Quotas": [
    {
      "ServiceCode": "xray",
      "ServiceName": "AWS X-Ray",
      "QuotaArn": "arn:aws:servicequotas:us-west-2::xray/L-C6B6F05D",
      "QuotaCode": "L-C6B6F05D",
      "QuotaName": "Indexed annotations per trace",
      "Value": 50.0,
      "Unit": "None",
      "Adjustable": false,
      "GlobalQuota": false
    },
    {
      "ServiceCode": "xray",
      "ServiceName": "AWS X-Ray",
      "QuotaArn": "arn:aws:servicequotas:us-west-2::xray/L-D781C0FD",
      "QuotaCode": "L-D781C0FD",
      "QuotaName": "Segment document size",
      "Value": 64.0,
      "Unit": "Kilobytes",
      "Adjustable": false,
      "GlobalQuota": false
    },
    {
      "ServiceCode": "xray",
      "ServiceName": "AWS X-Ray",
      "QuotaArn": "arn:aws:servicequotas:us-west-2::xray/L-998BFF16",
```

```

        "QuotaCode": "L-998BFF16",
        "QuotaName": "Trace and service graph retention in days",
        "Value": 30.0,
        "Unit": "None",
        "Adjustable": false,
        "GlobalQuota": false
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAwsDefaultServiceQuotas](#)中的。

## list-requested-service-quota-change-history-by-quota

以下代码示例演示了如何使用 `list-requested-service-quota-change-history-by-quota`。

### AWS CLI

列出您的配额增加请求

以下`list-requested-service-quota-change-history-by-quota`示例列出了针对指定配额的配额增加请求。

```

aws service-quotas list-requested-service-quota-change-history-by-quota \
  --service-code ec2 \
  --quota-code L-20F13EBD

```

输出：

```

{
  "RequestedQuotas": [
    {
      "Id": "d187537d15254312a9609aa51bbf7624u7W49tP0",
      "CaseId": "6780195351",
      "ServiceCode": "ec2",
      "ServiceName": "Amazon Elastic Compute Cloud (Amazon EC2)",
      "QuotaCode": "L-20F13EBD",
      "QuotaName": "Running Dedicated c5n Hosts",
      "DesiredValue": 2.0,
      "Status": "CASE_OPENED",
    }
  ]
}

```



```

        "Created": 1580446904.067,
        "LastUpdated": 1580446953.265,
        "Requester": "{\"accountId\":\"123456789012\",\"callerArn\":
\\\"arn:aws:iam::123456789012:root\\\"}\",
        "QuotaArn": "arn:aws:servicequotas:us-east-2:123456789012:ec2/
L-20F13EBD",
        "GlobalQuota": false,
        "Unit": "None"
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参  
考[ListRequestedServiceQuotaChangeHistoryByQuota](#)中的。

## list-requested-service-quota-change-history

以下代码示例演示了如何使用 `list-requested-service-quota-change-history`。

### AWS CLI

列出您的配额增加请求

以下`list-requested-service-quota-change-history`示例列出了指定服务的配额增加请求。

```
aws service-quotas list-requested-service-quota-change-history \
--service-code ec2
```

输出：

```

{
  "RequestedQuotas": [
    {
      "Id": "d187537d15254312a9609aa51bbf7624u7W49tP0",
      "CaseId": "6780195351",
      "ServiceCode": "ec2",
      "ServiceName": "Amazon Elastic Compute Cloud (Amazon EC2)",
      "QuotaCode": "L-20F13EBD",
      "QuotaName": "Running Dedicated c5n Hosts",
      "DesiredValue": 2.0,

```

```

        "Status": "CASE_OPENED",
        "Created": 1580446904.067,
        "LastUpdated": 1580446953.265,
        "Requester": "{\"accountId\":\"123456789012\",\"callerArn\":
\\\"arn:aws:iam::123456789012:root\\\"}\",
        "QuotaArn": "arn:aws:servicequotas:us-east-2:123456789012:ec2/
L-20F13EBD",
        "GlobalQuota": false,
        "Unit": "None"
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRequestedServiceQuotaChangeHistory](#)中的。

## list-service-quotas

以下代码示例演示了如何使用 list-service-quotas。

### AWS CLI

列出服务的配额

以下list-service-quotas示例显示了有关配额的详细信息 AWS CloudFormation。

```
aws service-quotas list-service-quotas \
--service-code cloudformation
```

输出：

```

{
  "Quotas": [
    {
      "ServiceCode": "cloudformation",
      "ServiceName": "AWS CloudFormation",
      "QuotaArn": "arn:aws:servicequotas:us-
east-2:123456789012:cloudformation/L-87D14FB7",
      "QuotaCode": "L-87D14FB7",
      "QuotaName": "Output count in CloudFormation template",
      "Value": 60.0,

```

```
        "Unit": "None",
        "Adjustable": false,
        "GlobalQuota": false
    },
    {
        "ServiceCode": "cloudformation",
        "ServiceName": "AWS CloudFormation",
        "QuotaArn": "arn:aws:servicequotas:us-
east-2:123456789012:cloudformation/L-0485CB21",
        "QuotaCode": "L-0485CB21",
        "QuotaName": "Stack count",
        "Value": 200.0,
        "Unit": "None",
        "Adjustable": true,
        "GlobalQuota": false
    }
]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListServiceQuotas](#)中的。

## list-services

以下代码示例演示了如何使用 list-services。

### AWS CLI

列出可用服务

以下命令列出了 Service Quotas 中可用的服务。

```
aws service-quotas list-services
```

输出：

```
{
  "Services": [
    {
      "ServiceCode": "AWSCloudMap",
      "ServiceName": "AWS Cloud Map"
    },
    {
```

```
    "ServiceCode": "access-analyzer",
    "ServiceName": "Access Analyzer"
  },
  {
    "ServiceCode": "acm",
    "ServiceName": "AWS Certificate Manager (ACM)"
  },
  ...truncated...
  {
    "ServiceCode": "xray",
    "ServiceName": "AWS X-Ray"
  }
]
```

您可以添加`--query`参数来筛选显示您感兴趣的信息。以下示例仅显示服务代码。

```
aws service-quotas list-services \
  --query Services[*].ServiceCode
```

输出：

```
[
  "AWSCloudMap",
  "access-analyzer",
  "acm",
  "acm-pca",
  "amplify",
  "apigateway",
  "application-autoscaling",
  ...truncated...
  "xray"
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListServices](#)中的。

## request-service-quota-increase

以下代码示例演示了如何使用 `request-service-quota-increase`。

## AWS CLI

### 申请增加服务配额

以下request-service-quota-increase示例请求增加指定的服务配额。

```
aws service-quotas request-service-quota-increase \  
  --service-code ec2 \  
  --quota-code L-20F13EBD \  
  --desired-value 2
```

输出：

```
{  
  "RequestedQuota": {  
    "Id": "d187537d15254312a9609aa51bbf7624u7W49tP0",  
    "ServiceCode": "ec2",  
    "ServiceName": "Amazon Elastic Compute Cloud (Amazon EC2)",  
    "QuotaCode": "L-20F13EBD",  
    "QuotaName": "Running Dedicated c5n Hosts",  
    "DesiredValue": 2.0,  
    "Status": "PENDING",  
    "Created": 1580446904.067,  
    "Requester": "{\"accountId\":\"123456789012\",\"callerArn\":  
  \":\"arn:aws:iam::123456789012:root\"}\",  
    "QuotaArn": "arn:aws:servicequotas:us-east-2:123456789012:ec2/L-20F13EBD",  
    "GlobalQuota": false,  
    "Unit": "None"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RequestServiceQuotaIncrease](#)中的。

## 使用 Amazon SES 示例 AWS CLI

以下代码示例向您展示了如何在 Amazon SES 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## **delete-identity**

以下代码示例演示了如何使用 delete-identity。

AWS CLI

删除身份

以下示例使用 delete-identity 命令从通过 Amazon SES 验证的身份列表中删除身份：

```
aws ses delete-identity --identity user@example.com
```

有关已验证身份的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证电子邮件地址和域”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteIdentity](#) 中的。

## **get-identity-dkim-attributes**

以下代码示例演示了如何使用 get-identity-dkim-attributes。

AWS CLI

获取身份列表的 Amazon SES Easy DKIM 属性

以下示例使用 get-identity-dkim-attributes 命令检索身份列表的 Amazon SES Easy DKIM 属性：

```
aws ses get-identity-dkim-attributes --identities "example.com" "user@example.com"
```

输出：

```
{
  "DkimAttributes": {
    "example.com": {
      "DkimTokens": [
        "EXAMPLEjcs5xoyqytjsotsijas7236gr",
        "EXAMPLEjr76cvoc6mysspnioorxsn6ep",
        "EXAMPLEkbnkqkhlm2lyz77ppkulerm4k"
      ],
      "DkimEnabled": true,
      "DkimVerificationStatus": "Success"
    },
    "user@example.com": {
      "DkimEnabled": false,
      "DkimVerificationStatus": "NotStarted"
    }
  }
}
```

如果调用此命令时，列表包含从未提交验证的身份，则该身份将不会出现在输出中。

有关 Easy DKIM 的更多信息，请参阅《亚马逊简单电子邮件服务开发者指南》中的 Amazon SES 中的 Easy DKIM。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetIdentityDkimAttributes](#) 中的。

## get-identity-notification-attributes

以下代码示例演示了如何使用 `get-identity-notification-attributes`。

### AWS CLI

获取身份列表的 Amazon SES 通知属性

以下示例使用 `get-identity-notification-attributes` 命令检索身份列表的 Amazon SES 通知属性：

```
aws ses get-identity-notification-attributes --identities "user1@example.com"
"user2@example.com"
```

输出：

```
{
```

```
"NotificationAttributes": {
  "user1@example.com": {
    "ForwardingEnabled": false,
    "ComplaintTopic": "arn:aws:sns:us-east-1:EXAMPLE65304:MyTopic",
    "BounceTopic": "arn:aws:sns:us-east-1:EXAMPLE65304:MyTopic",
    "DeliveryTopic": "arn:aws:sns:us-east-1:EXAMPLE65304:MyTopic"
  },
  "user2@example.com": {
    "ForwardingEnabled": true
  }
}
```

此命令返回电子邮件反馈转发的状态，如果适用，还会返回向其发送退信、投诉和送达通知的 Amazon SNS 主题的亚马逊资源名称 (ARN)。

如果调用此命令时，列表包含从未提交验证的身份，则该身份将不会出现在输出中。

有关通知的更多信息，请参阅《亚马逊简单电子邮件服务开发者指南》中的在 Amazon SES 上使用通知。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetIdentityNotificationAttributes](#) 中的。

## get-identity-verification-attributes

以下代码示例演示了如何使用 `get-identity-verification-attributes`。

### AWS CLI

获取身份列表的 Amazon SES 验证状态

以下示例使用 `get-identity-verification-attributes` 命令检索身份列表的 Amazon SES 验证状态：

```
aws ses get-identity-verification-attributes --identities "user1@example.com"
"user2@example.com"
```

输出：

```
{
  "VerificationAttributes": {
```



```
    "user1@example.com": {
      "VerificationStatus": "Success"
    },
    "user2@example.com": {
      "VerificationStatus": "Pending"
    }
  }
}
```

如果调用此命令时，列表包含从未提交验证的身份，则该身份将不会出现在输出中。

有关已验证身份的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证电子邮件地址和域”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetIdentityVerificationAttributes](#) 中的。

## get-send-quota

以下代码示例演示了如何使用 `get-send-quota`。

### AWS CLI

获取 Amazon SES 发送限制

以下示例使用 `get-send-quota` 命令返回 Amazon SES 发送限制：

```
aws ses get-send-quota
```

输出：

```
{
  "Max24HourSend": 200.0,
  "SentLast24Hours": 1.0,
  "MaxSendRate": 1.0
}
```

Max24 HourSend 是您的发送配额，这是您在 24 小时内可以发送的最大电子邮件数量。发送配额反映一个滚动的时段。每当您尝试发送电子邮件时，Amazon SES 都会检查您在过去 24 小时内发送的电子邮件数量。只要您发送电子邮件总数小于您的配额，发送请求就会被接受，并发送您的电子邮件。

SentLast24 小时是您在过去 24 小时内发送的电子邮件数量。

MaxSendRate 是您每秒可以发送的最大电子邮件数。

请注意，发送限制基于收件人而不是消息。例如，一封包含 10 个收件人的电子邮件占用 10 份发送配额。

有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“管理您的 Amazon SES 发送限制”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetSendQuota](#) 中的。

## get-send-statistics

以下代码示例演示了如何使用 get-send-statistics。

### AWS CLI

让你的 Amazon SES 发送统计数据

以下示例使用 get-send-statistics 命令返回您的 Amazon SES 发送统计数据

```
aws ses get-send-statistics
```

输出：

```
{
  "SendDataPoints": [
    {
      "Complaints": 0,
      "Timestamp": "2013-06-12T19:32:00Z",
      "DeliveryAttempts": 2,
      "Bounces": 0,
      "Rejects": 0
    },
    {
      "Complaints": 0,
      "Timestamp": "2013-06-12T00:47:00Z",
      "DeliveryAttempts": 1,
      "Bounces": 0,
      "Rejects": 0
    }
  ]
}
```

```
]
}
```

结果是一个数据点列表，代表最近两周的发送活动。列表中的每个数据点都包含 15 分钟间隔的统计数据。

在此示例中，只有两个数据点，因为用户在过去两周内发送的唯一电子邮件是在两个 15 分钟间隔内发送的。

有关更多信息，请参阅《亚马逊简单电子邮件服务开发者指南》中的监控您的 Amazon SES 使用情况统计信息。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetSendStatistics](#) 中的。

## list-identities

以下代码示例演示了如何使用 `list-identities`。

### AWS CLI

列出特定 AWS 账户的所有身份（电子邮件地址和域名）

以下示例使用 `list-identities` 命令列出已提交到 Amazon SES 进行验证的所有身份：

```
aws ses list-identities
```

输出：

```
{
  "Identities": [
    "user@example.com",
    "example.com"
  ]
}
```

返回的列表包含所有身份，无论验证状态如何（已验证、待验证、失败等）。

在此示例中，由于未指定 `identity-type` 参数，因此返回了电子邮件地址和域。

有关验证的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证电子邮件地址和域”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListIdentities](#)中的。

## send-email

以下代码示例演示了如何使用 send-email。

### AWS CLI

使用 Amazon SES 发送格式化的电子邮件

以下示例使用 send-email 命令发送格式化的电子邮件：

```
aws ses send-email --from sender@example.com --destination file://destination.json
--message file://message.json
```

输出：

```
{
  "MessageId": "EXAMPLEf3a5efcd1-51adec81-d2a4-4e3f-9fe2-5d85c1b23783-000000"
}
```

目标和消息是 JSON 数据结构，保存在当前目录下的 .json 文件中。这些文件如下所示：

destination.json:

```
{
  "ToAddresses": ["recipient1@example.com", "recipient2@example.com"],
  "CcAddresses": ["recipient3@example.com"],
  "BccAddresses": []
}
```

message.json:

```
{
  "Subject": {
    "Data": "Test email sent using the AWS CLI",
    "Charset": "UTF-8"
  },
  "Body": {
    "Text": {
      "Data": "This is the message body in text format.",

```

```
    "Charset": "UTF-8"
  },
  "Html": {
    "Data": "This message body contains HTML formatting. It can, for example,
contain links like this one: <a class=\"ulink\" href=\"http://docs.aws.amazon.com/
ses/latest/DeveloperGuide\" target=\"_blank\">Amazon SES Developer Guide</a>.",
    "Charset": "UTF-8"
  }
}
}
```

请将发件人和收件人的电子邮件地址替换为您要使用的电子邮件地址。请注意，发件人的电子邮件地址必须已通过 Amazon SES 验证。在您获得 Amazon SES 的生产访问权限之前，您还必须验证每个收件人的电子邮件地址，除非收件人是 Amazon SES 邮箱模拟器。有关验证的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证电子邮件地址和域”。

输出中的消息 ID 表示 send-email 调用成功。

如果您没有收到电子邮件，请检查垃圾邮件。

有关发送格式化电子邮件的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“使用 Amazon SES API 发送格式化的电子邮件”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SendEmail](#) 中的。

## send-raw-email

以下代码示例演示了如何使用 send-raw-email。

### AWS CLI

使用 Amazon SES 发送原始电子邮件

以下示例使用 send-raw-email 命令发送带有 TXT 附件的电子邮件：

```
aws ses send-raw-email --raw-message file://message.json
```

输出：

```
{
```

```
"MessageId": "EXAMPLEf3f73d99b-c63fb06f-d263-41f8-a0fb-d0dc67d56c07-000000"  
}
```

原始消息是一个 JSON 数据结构，保存在当前目录下名为 `message.json` 的文件中。其中包含以下内容：

```
{  
  "Data": "From: sender@example.com\nTo: recipient@example.com\nSubject: Test email  
sent using the AWS CLI (contains an attachment)\nMIME-Version: 1.0\nContent-type:  
Multipart/Mixed; boundary=\"NextPart\"\n\n--NextPart\nContent-Type: text/plain  
\n\nThis is the message body.\n\n--NextPart\nContent-Type: text/plain;\nContent-  
Disposition: attachment; filename=\"attachment.txt\"\n\nThis is the text in the  
attachment.\n\n--NextPart--"  
}
```

如您所见，“Data”是一个长字符串，包含 MIME 格式的全部原始电子邮件内容，其中包括一个名为 `attachment.txt` 的附件。

请将 `sender@example.com` 和 `recipient@example.com` 替换为您要使用的地址。请注意，发件人的电子邮件地址必须已通过 Amazon SES 验证。在您获得 Amazon SES 的生产访问权限之前，您还必须验证收件人的电子邮件地址，除非收件人是 Amazon SES 邮箱模拟器。有关验证的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证电子邮件地址和域”。

输出中的消息 ID 表示对的调用 `send-raw-email` 成功。

如果您没有收到电子邮件，请检查垃圾邮件。

有关发送原始电子邮件的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“使用 Amazon SES API 发送原始电子邮件”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SendRawEmail](#) 中的。

## set-identity-dkim-enabled

以下代码示例演示了如何使用 `set-identity-dkim-enabled`。

### AWS CLI

为经过 Amazon SES 验证的身份启用或禁用 Easy DKIM

以下示例使用 `set-identity-dkim-enabled` 命令为已验证的电子邮件地址禁用 DKIM：

```
aws ses set-identity-dkim-enabled --identity user@example.com --no-dkim-enabled
```

有关 Easy DKIM 的更多信息，请参阅《亚马逊简单电子邮件服务开发者指南》中的 Amazon SES 中的 Easy DKIM。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SetIdentityDkimEnabled](#) 中的。

## **set-identity-feedback-forwarding-enabled**

以下代码示例演示了如何使用 `set-identity-feedback-forwarding-enabled`。

### AWS CLI

为经过 Amazon SES 验证的身份启用或禁用退回邮件和投诉电子邮件反馈转发

以下示例使用 `set-identity-feedback-forwarding-enabled` 命令使经过验证的电子邮件地址能够通过电子邮件接收退信和投诉通知：

```
aws ses set-identity-feedback-forwarding-enabled --identity user@example.com --forwarding-enabled
```

您需要通过 Amazon SNS 或电子邮件反馈转发接收退回通知和投诉通知，因此，只有在退回通知和投诉通知均选择 Amazon SNS 主题时，才能禁用电子邮件反馈转发。

有关通知的更多信息，请参阅《亚马逊简单电子邮件服务开发者指南》中的在 Amazon SES 上使用通知。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SetIdentityFeedbackForwardingEnabled](#) 中的。

## **set-identity-notification-topic**

以下代码示例演示了如何使用 `set-identity-notification-topic`。

### AWS CLI

设置 Amazon SES 将发布针对已验证身份的退信、投诉和/或送达通知的 Amazon SNS 主题

以下示例使用 `set-identity-notification-topic` 命令指定经过验证的电子邮件地址将收到退回通知的 Amazon SNS 主题：

```
aws ses set-identity-notification-topic --identity user@example.com --notification-type Bounce --sns-topic arn:aws:sns:us-east-1:EXAMPLE65304:MyTopic
```

有关通知的更多信息，请参阅《亚马逊简单电子邮件服务开发者指南》中的在 Amazon SES 上使用通知。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetIdentityNotificationTopic](#)中的。

## verify-domain-dkim

以下代码示例演示了如何使用 verify-domain-dkim。

### AWS CLI

生成经过验证的域名的 DKIM 令牌，以便使用 Amazon SES 进行 DKIM 签名

以下示例使用verify-domain-dkim命令为已通过 Amazon SES 验证的域生成 DKIM 令牌：

```
aws ses verify-domain-dkim --domain example.com
```

输出：

```
{
  "DkimTokens": [
    "EXAMPLEEq76owjnks31nluwg65scbemvw",
    "EXAMPLEi3dnsj67hstzaj673klariwx2",
    "EXAMPLEwfbtcukvimehexktdtaz6naj"
  ]
}
```

要设置 DKIM，您必须使用返回的 DKIM 令牌使用指向 Amazon SES 托管的 DKIM 公钥的别名记录更新域名的 DNS 设置。有关更多信息，请参阅《亚马逊简单电子邮件服务开发者指南》中的 Amazon SES 中的 Easy DKIM。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[VerifyDomainDkim](#)中的。

## verify-domain-identity

以下代码示例演示了如何使用 verify-domain-identity。



## AWS CLI

使用 Amazon SES 验证域

以下示例使用 `verify-domain-identity` 命令验证域：

```
aws ses verify-domain-identity --domain example.com
```

输出：

```
{
  "VerificationToken": "eoEmxw+YaYhb3h3iVJHuXMJXqeu1q1/wmvjuEXAMPLE"
}
```

要完成域验证，您必须在域的 DNS 设置中添加一条包含返回的验证令牌的 TXT 记录。有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证域”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[VerifyDomainIdentity](#)中的。

## `verify-email-identity`

以下代码示例演示了如何使用 `verify-email-identity`。

## AWS CLI

使用 Amazon SES 验证电子邮件地址

以下示例使用 `verify-email-identity` 命令验证电子邮件地址：

```
aws ses verify-email-identity --email-address user@example.com
```

在使用 Amazon SES 发送电子邮件之前，您必须验证从中发送电子邮件的地址或域，以证明您拥有该地址或域。如果您尚未获得生产访问权限，则还需要验证所有收件电子邮件地址，Amazon SES 邮箱模拟器提供的电子邮件地址除外。

接 `verify-email-identity` 到电话后，该电子邮件地址将收到一封验证电子邮件。用户必须单击此电子邮件中的链接才能完成验证过程。

有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证电子邮件地址”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[VerifyEmailIdentity](#)中的。

## 使用 Shield 示例 AWS CLI

以下代码示例向您展示了如何使用 with Shield 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **associate-drt-log-bucket**

以下代码示例演示了如何使用 associate-drt-log-bucket。

AWS CLI

授权 DRT 访问亚马逊 S3 存储桶

以下associate-drt-log-bucket示例在 DRT 和指定的 S3 存储桶之间创建关联。这允许 DRT 代表账户访问存储桶。：

```
aws shield associate-drt-log-bucket \  
  --log-bucket flow-logs-for-website-lb
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS Shield 高级开发者指南》中的授权 DDoS 响应团队](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateDrtLogBucket](#)中的。

### **associate-drt-role**

以下代码示例演示了如何使用 associate-drt-role。

## AWS CLI

授权 DRT 代表您缓解潜在的攻击

以下 `associate-drt-role` 示例在 DRT 和指定角色之间创建关联。DRT 可以使用该角色来访问和管理账户。

```
aws shield associate-drt-role \  
  --role-arn arn:aws:iam::123456789012:role/service-role/DrtRole
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS Shield 高级开发者指南》中的授权 DDoS 响应团队](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateDrtRole](#) 中的。

## create-protection

以下代码示例演示了如何使用 `create-protection`。

## AWS CLI

为单个 AWS 资源启用 AWS Shield 高级保护

以下 `create-protection` 示例为指定 AWS CloudFront 发行版启用 Shield 高级保护。

```
aws shield create-protection \  
  --name "Protection for CloudFront distribution" \  
  --resource-arn arn:aws:cloudfront::123456789012:distribution/E198WC25FX0WY8
```

输出：

```
{  
  "ProtectionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"  
}
```

有关更多信息，请参阅 [《AWS Shield 高级开发者指南》中的“指定要保护的资源”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateProtection](#) 中的。

## create-subscription

以下代码示例演示了如何使用 `create-subscription`。

## AWS CLI

为账户启用 AWS Shield 高级防护

以下create-subscription示例为账户启用 Shield 高级保护。

```
aws shield create-subscription
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS Shield 高级开发者指南](#)》中的[AWS Shield 高级版入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSubscription](#)中的。

## delete-protection

以下代码示例演示了如何使用 delete-protection。

### AWS CLI

从 AWS 资源中移除 AWS Shield 高级防护

以下delete-protection示例移除了指定的 AWS Shield 高级保护。

```
aws shield delete-protection \  
  --protection-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS Shield 高级开发者指南](#)》中的[从 AWS 资源中移除AWS Shield Advance d](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteProtection](#)中的。

## describe-attack

以下代码示例演示了如何使用 describe-attack。

### AWS CLI

检索攻击的详细描述

以下describe-attack示例显示有关具有指定攻击 ID 的 DDoS 攻击的详细信息。您可以通过运行list-attacks命令来获取攻击 ID。

```
aws shield describe-attack --attack-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222
```

输出：

```
{
  "Attack": {
    "AttackId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "ResourceArn": "arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/testElb",
    "SubResources": [
      {
        "Type": "IP",
        "Id": "192.0.2.2",
        "AttackVectors": [
          {
            "VectorType": "SYN_FLOOD",
            "VectorCounters": [
              {
                "Name": "SYN_FLOOD_BPS",
                "Max": 982184.0,
                "Average": 982184.0,
                "Sum": 11786208.0,
                "N": 12,
                "Unit": "BPS"
              }
            ]
          }
        ]
      },
      {
        "Type": "IP",
        "Id": "192.0.2.3",
        "AttackVectors": [
          {
            "VectorType": "SYN_FLOOD",
            "VectorCounters": [
              {
                "Name": "SYN_FLOOD_BPS",
                "Max": 982184.0,
```

```
        "Average": 982184.0,
        "Sum": 9821840.0,
        "N": 10,
        "Unit": "BPS"
      }
    ]
  },
  "Counters": []
},
{
  "Type": "IP",
  "Id": "192.0.2.4",
  "AttackVectors": [
    {
      "VectorType": "SYN_FLOOD",
      "VectorCounters": [
        {
          "Name": "SYN_FLOOD_BPS",
          "Max": 982184.0,
          "Average": 982184.0,
          "Sum": 7857472.0,
          "N": 8,
          "Unit": "BPS"
        }
      ]
    }
  ],
  "Counters": []
},
{
  "Type": "IP",
  "Id": "192.0.2.5",
  "AttackVectors": [
    {
      "VectorType": "SYN_FLOOD",
      "VectorCounters": [
        {
          "Name": "SYN_FLOOD_BPS",
          "Max": 982184.0,
          "Average": 982184.0,
          "Sum": 1964368.0,
          "N": 2,
          "Unit": "BPS"
        }
      ]
    }
  ],
  "Counters": []
}
```

```
    }
  ]
}
],
"Counters": []
},
{
  "Type": "IP",
  "Id": "2001:DB8::bcde:4321:8765:0:0",
  "AttackVectors": [
    {
      "VectorType": "SYN_FLOOD",
      "VectorCounters": [
        {
          "Name": "SYN_FLOOD_BPS",
          "Max": 982184.0,
          "Average": 982184.0,
          "Sum": 1964368.0,
          "N": 2,
          "Unit": "BPS"
        }
      ]
    }
  ]
},
"Counters": []
},
{
  "Type": "IP",
  "Id": "192.0.2.6",
  "AttackVectors": [
    {
      "VectorType": "SYN_FLOOD",
      "VectorCounters": [
        {
          "Name": "SYN_FLOOD_BPS",
          "Max": 982184.0,
          "Average": 982184.0,
          "Sum": 1964368.0,
          "N": 2,
          "Unit": "BPS"
        }
      ]
    }
  ]
},
],
```

```
        "Counters": []
      }
    ],
    "StartTime": 1576024927.457,
    "EndTime": 1576025647.457,
    "AttackCounters": [],
    "AttackProperties": [
      {
        "AttackLayer": "NETWORK",
        "AttackPropertyIdentifier": "SOURCE_IP_ADDRESS",
        "TopContributors": [
          {
            "Name": "198.51.100.5",
            "Value": 2024475682
          },
          {
            "Name": "198.51.100.8",
            "Value": 1311380863
          },
          {
            "Name": "203.0.113.4",
            "Value": 900599855
          },
          {
            "Name": "198.51.100.4",
            "Value": 769417366
          },
          {
            "Name": "203.1.113.13",
            "Value": 757992847
          }
        ],
        "Unit": "BYTES",
        "Total": 92773354841
      },
      {
        "AttackLayer": "NETWORK",
        "AttackPropertyIdentifier": "SOURCE_COUNTRY",
        "TopContributors": [
          {
            "Name": "United States",
            "Value": 80938161764
          },
          {
```



```
        "Name": "Brazil",
        "Value": 9929864330
    },
    {
        "Name": "Netherlands",
        "Value": 1635009446
    },
    {
        "Name": "Mexico",
        "Value": 144832971
    },
    {
        "Name": "Japan",
        "Value": 45369000
    }
],
"Unit": "BYTES",
"Total": 92773354841
},
{
    "AttackLayer": "NETWORK",
    "AttackPropertyIdentifier": "SOURCE_ASN",
    "TopContributors": [
        {
            "Name": "12345",
            "Value": 74953625841
        },
        {
            "Name": "12346",
            "Value": 4440087595
        },
        {
            "Name": "12347",
            "Value": 1635009446
        },
        {
            "Name": "12348",
            "Value": 1221230000
        },
        {
            "Name": "12349",
            "Value": 1199425294
        }
    ]
},
```

```
        "Unit": "BYTES",
        "Total": 92755479921
      }
    ],
    "Mitigations": []
  }
}
```

有关更多信息，请参阅《AWS Shield 高级开发者指南》中的[查看 DDoS 事件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAttack](#)中的。

## describe-drt-access

以下代码示例演示了如何使用 describe-drt-access。

### AWS CLI

要检索 DRT 必须代表您缓解攻击的授权的描述

以下describe-drt-access示例检索 DRT 拥有的角色和 S3 存储桶授权，这使其能够代表您响应潜在的攻击。

```
aws shield describe-drt-access
```

输出：

```
{
  "RoleArn": "arn:aws:iam::123456789012:role/service-role/DrtRole",
  "LogBucketList": [
    "flow-logs-for-website-lb"
  ]
}
```

有关更多信息，请参阅《AWS Shield 高级开发者指南》中的[授权 DDoS 响应团队](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDrtAccess](#)中的。

## describe-emergency-contact-settings

以下代码示例演示了如何使用 describe-emergency-contact-settings。

## AWS CLI

检索您在 DRT 中存档的紧急电子邮件地址

以下describe-emergency-contact-settings示例检索账户的 DRT 中存档的电子邮件地址。这些是 DRT 在应对可疑攻击时应联系的地址。

```
aws shield describe-emergency-contact-settings
```

输出：

```
{
  "EmergencyContactList": [
    {
      "EmailAddress": "ops@example.com"
    },
    {
      "EmailAddress": "ddos-notifications@example.com"
    }
  ]
}
```

有关更多信息，请参阅《AWS Shield <<https://docs.aws.amazon.com/waf/latest/developerguide/ddos-overview.html>> 高级开发者指南》中的AWS Shield 工作原理。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEmergencyContactSettings](#)中的。

## describe-protection

以下代码示例演示了如何使用 describe-protection。

### AWS CLI

检索 AWS Shield 高级防护的详细信息

以下describe-protection示例显示有关具有指定 ID 的 Shield 高级防护的详细信息。您可以通过运行list-protections命令来获取保护 ID。

```
aws shield describe-protection \
  --protection-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
  "Protection": {
    "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Name": "1.2.3.4",
    "ResourceArn": "arn:aws:ec2:us-west-2:123456789012:eip-allocation/
eipalloc-0ac1537af40742a6d"
  }
}
```

有关更多信息，请参阅《AWS Shield 高级开发者指南》中的“[指定要保护的资源](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeProtection](#) 中的。

## describe-subscription

以下代码示例演示了如何使用 describe-subscription。

### AWS CLI

检索账户的 AWS Shield 高级保护详情

以下 describe-subscription 示例显示了有关为该账户提供的 Shield 高级保护的详细信息。：

```
aws shield describe-subscription
```

输出：

```
{
  "Subscription": {
    "StartTime": 1534368978.0,
    "EndTime": 1597613778.0,
    "TimeCommitmentInSeconds": 63244800,
    "AutoRenew": "ENABLED",
    "Limits": [
      {
        "Type": "GLOBAL_ACCELERATOR",
        "Max": 1000
      },
      {
        "Type": "ROUTE53_HOSTED_ZONE",
        "Max": 1000
      }
    ]
  }
}
```

```
{
  {
    "Type": "CF_DISTRIBUTION",
    "Max": 1000
  },
  {
    "Type": "ELB_LOAD_BALANCER",
    "Max": 1000
  },
  {
    "Type": "EC2_ELASTIC_IP_ALLOCATION",
    "Max": 1000
  }
]
}
```

有关更多信息，[请参阅《AWS Shield 高级开发者指南》中的AWS Shield 工作原理。](#)

- 有关 API 的详细信息，[请参阅AWS CLI 命令参考DescribeSubscription](#)中的。

## disassociate-drt-log-bucket

以下代码示例演示了如何使用 disassociate-drt-log-bucket。

### AWS CLI

移除 DRT 代表您访问 Amazon S3 存储桶的授权

以下disassociate-drt-log-bucket示例删除了 DRT 与指定 S3 存储桶之间的关联。此命令完成后，DRT 将无法再代表该账户访问存储桶。

```
aws shield disassociate-drt-log-bucket \
  --log-bucket flow-logs-for-website-lb
```

此命令不生成任何输出。

有关更多信息，[请参阅《AWS Shield 高级开发者指南》中的授权 DDoS 响应团队。](#)

- 有关 API 的详细信息，[请参阅AWS CLI 命令参考DisassociateDrtLogBucket](#)中的。

## disassociate-drt-role

以下代码示例演示了如何使用 disassociate-drt-role。

## AWS CLI

移除 DRT 的授权以代表您缓解潜在攻击

以下`disassociate-drt-role`示例删除了 DRT 与账户之间的关联。通话结束后，DRT 将无法再访问或管理您的帐户。

```
aws shield disassociate-drt-role
```

此命令不生成任何输出。

有关更多信息，请参阅 [《AWS Shield 高级开发者指南》中的授权 DDoS 响应团队](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DisassociateDrtRole](#) 中的。

## get-subscription-state

以下代码示例演示了如何使用 `get-subscription-state`。

## AWS CLI

检索账户 AWS Shield Advanced 订阅的当前状态

以下`get-subscription-state`示例检索账户的 Shield 高级保护的状态。

```
aws shield get-subscription-state
```

输出：

```
{
  "SubscriptionState": "ACTIVE"
}
```

有关更多信息，请参阅 [《AWS Shield 高级开发者指南》中的 AWS Shield 工作原理](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetSubscriptionState](#) 中的。

## list-attacks

以下代码示例演示了如何使用 `list-attacks`。

## AWS CLI

### 从 AWS Shield Advanced 中检索攻击摘要

以下 `list-attacks` 示例检索指定 AWS CloudFront 分布在指定时间段内的攻击摘要。响应包括攻击 ID，您可以将其提供给 `describe-attack` 命令以获取有关攻击的详细信息。

```
aws shield list-attacks \
  --resource-arns arn:aws:cloudfront::12345678910:distribution/E1PXM22ZVFAOR \
  --start-time FromInclusive=1529280000,ToExclusive=1529300000
```

输出：

```
{
  "AttackSummaries": [
    {
      "AttackId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "ResourceArn": "arn:aws:cloudfront::123456789012:distribution/E1PXM22ZVFAOR",
      "StartTime": 1529280000.0,
      "EndTime": 1529449200.0,
      "AttackVectors": [
        {
          "VectorType": "SYN_FLOOD"
        }
      ]
    }
  ]
}
```

有关更多信息，请参阅《AWS Shield 高级开发者指南》中的[查看 DDoS 事件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListAttacks](#) 中的。

## list-protections

以下代码示例演示了如何使用 `list-protections`。

## AWS CLI

### 从 AWS Shield Advanced 中检索防护摘要

以下list-protections示例检索为该账户启用的保护的摘要。

```
aws shield list-protections
```

输出：

```
{
  "Protections": [
    {
      "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "Name": "Protection for CloudFront distribution",
      "ResourceArn": "arn:aws:cloudfront::123456789012:distribution/
E198WC25FX0WY8"
    }
  ]
}
```

有关更多信息，请参阅《AWS Shield 高级开发者指南》中的“[指定要保护的资源](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListProtections](#)中的。

## update-emergency-contact-settings

以下代码示例演示了如何使用 update-emergency-contact-settings。

### AWS CLI

定义 DRT 中存档的紧急电子邮件地址

以下update-emergency-contact-settings示例定义了 DRT 在响应可疑攻击时应联系的两个电子邮件地址。

```
aws shield update-emergency-contact-settings \
  --emergency-contact-list EmailAddress=ops@example.com EmailAddress=ddos-
notifications@example.com
```

此命令不生成任何输出。

有关更多信息，请参阅《[AWS Shield 高级开发者指南](#)》中的[AWS Shield 工作原理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateEmergencyContactSettings](#)中的。



## update-subscription

以下代码示例演示了如何使用 update-subscription。

### AWS CLI

修改账户的 AWS Shield Advanced 订阅

以下update-subscription示例为该账户启用了 AWS Shield Advanced 订阅的自动续订。

```
aws shield update-subscription \  
  --auto-renew ENABLED
```

此命令不生成任何输出。

有关更多信息，[请参阅《AWS Shield 高级开发者指南》中的AWS Shield 工作原理。](#)

- 有关 API 的详细信息，[请参阅AWS CLI 命令参考UpdateSubscription](#)中的。

## 签名者使用示例 AWS CLI

以下代码示例向您展示了如何使用 with Signer 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## cancel-signing-profile

以下代码示例演示了如何使用 cancel-signing-profile。

## AWS CLI

### 删除签名档案

以下cancel-signing-profile示例从 S AWS igner 中删除现有的签名配置文件。

```
aws signer cancel-signing-profile \  
  --profile-name MyProfile1
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelSigningProfile](#)中的。

## describe-signing-job

以下代码示例演示了如何使用 describe-signing-job。

## AWS CLI

显示有关签名作业的详细信息

以下describe-signing-job示例显示有关指定签名任务的详细信息。

```
aws signer describe-signing-job \  
  --job-id 2065c468-73e2-4385-a6c9-0123456789abc
```

输出：

```
{  
  "status": "Succeeded",  
  "completedAt": 1568412037,  
  "platformId": "AmazonFreeRTOS-Default",  
  "signingMaterial": {  
    "certificateArn": "arn:aws:acm:us-  
west-2:123456789012:certificate/6a55389b-306b-4e8c-a95c-0123456789abc"  
  },  
  "statusReason": "Signing Succeeded",  
  "jobId": "2065c468-73e2-4385-a6c9-0123456789abc",  
  "source": {  
    "s3": {  
      "version": "PNyFaUTgsQh5ZdMCcoCe6pT1g0pgB_M4",  
      "bucketName": "signer-source",  
      "key": "MyCode.rb"  
    }  
  }  
}
```

```

    }
  },
  "profileName": "MyProfile2",
  "signedObject": {
    "s3": {
      "bucketName": "signer-destination",
      "key": "signed-2065c468-73e2-4385-a6c9-0123456789abc"
    }
  },
  "requestedBy": "arn:aws:iam::123456789012:user/maria",
  "createdAt": 1568412036
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSigningJob](#)中的。

## get-signing-platform

以下代码示例演示了如何使用 get-signing-platform。

### AWS CLI

显示有关签名平台的详细信息

以下get-signing-platform示例显示有关指定签名平台的详细信息。

```
aws signer get-signing-platform \
  --platform-id AmazonFreeRTOS-TI-CC3220SF
```

输出：

```

{
  "category": "AWS",
  "displayName": "Amazon FreeRTOS SHA1-RSA CC3220SF-Format",
  "target": "SHA1-RSA-TISHA1",
  "platformId": "AmazonFreeRTOS-TI-CC3220SF",
  "signingConfiguration": {
    "encryptionAlgorithmOptions": {
      "defaultValue": "RSA",
      "allowedValues": [
        "RSA"
      ]
    },
    "hashAlgorithmOptions": {

```

```

        "defaultValue": "SHA1",
        "allowedValues": [
            "SHA1"
        ]
    },
    "maxSizeInMB": 16,
    "partner": "AmazonFreeRTOS",
    "signingImageFormat": {
        "defaultFormat": "JSONEmbedded",
        "supportedFormats": [
            "JSONEmbedded"
        ]
    }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSigningPlatform](#)中的。

## get-signing-profile

以下代码示例演示了如何使用 get-signing-profile。

### AWS CLI

显示有关签名资料的详细信息

以下get-signing-profile示例显示有关指定签名配置文件的详细信息。

```
aws signer get-signing-profile \
  --profile-name MyProfile3
```

输出：

```

{
  "platformId": "AmazonFreeRTOS-TI-CC3220SF",
  "profileName": "MyProfile3",
  "status": "Active",
  "signingMaterial": {
    "certificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/6a55389b-306b-4e8c-a95c-0123456789abc"
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSigningProfile](#)中的。

## list-signing-jobs

以下代码示例演示了如何使用 `list-signing-jobs`。

### AWS CLI

列出所有签名任务

以下`list-signing-jobs`示例显示了有关该账户所有签名任务的详细信息。

```
aws signer list-signing-jobs
```

在此示例中，返回了两个作业，一个成功，一个失败。

```
{
  "jobs": [
    {
      "status": "Succeeded",
      "signingMaterial": {
        "certificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/6a55389b-306b-4e8c-a95c-0123456789abc"
      },
      "jobId": "2065c468-73e2-4385-a6c9-0123456789abc",
      "source": {
        "s3": {
          "version": "PNyFaUTgsQh5ZdMCcoCe6pT1g0pgB_M4",
          "bucketName": "signer-source",
          "key": "MyCode.rb"
        }
      },
      "signedObject": {
        "s3": {
          "bucketName": "signer-destination",
          "key": "signed-2065c468-73e2-4385-a6c9-0123456789abc"
        }
      },
      "createdAt": 1568412036
    },
    {
      "status": "Failed",
      "source": {
```

```

        "s3": {
            "version": "PNyFaUTgsQh5ZdMCcoCe6pT1g0pgB_M4",
            "bucketName": "signer-source",
            "key": "MyOtherCode.rb"
        }
    },
    "signingMaterial": {
        "certificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/6a55389b-306b-4e8c-a95c-0123456789abc"
    },
    "createdAt": 1568402690,
    "jobId": "74d9825e-22fc-4a0d-b962-0123456789abc"
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSigningJobs](#)中的。

## list-signing-platforms

以下代码示例演示了如何使用 list-signing-platforms。

### AWS CLI

列出所有签名平台

以下list-signing-platforms示例显示有关所有可用签名平台的详细信息。

```
aws signer list-signing-platforms
```

输出：

```

{
  "platforms": [
    {
      "category": "AWS",
      "displayName": "AWS IoT Device Management SHA256-ECDSA ",
      "target": "SHA256-ECDSA",
      "platformId": "AWSIoTDeviceManagement-SHA256-ECDSA",
      "signingConfiguration": {
        "encryptionAlgorithmOptions": {
          "defaultValue": "ECDSA",
          "allowedValues": [

```

```

        "ECDSA"
      ]
    },
    "hashAlgorithmOptions": {
      "defaultValue": "SHA256",
      "allowedValues": [
        "SHA256"
      ]
    }
  },
  "maxSizeInMB": 2048,
  "partner": "AWSIoTDeviceManagement",
  "signingImageFormat": {
    "defaultFormat": "JSONDetached",
    "supportedFormats": [
      "JSONDetached"
    ]
  }
},
{
  "category": "AWS",
  "displayName": "Amazon FreeRTOS SHA1-RSA CC3220SF-Format",
  "target": "SHA1-RSA-TISHA1",
  "platformId": "AmazonFreeRTOS-TI-CC3220SF",
  "signingConfiguration": {
    "encryptionAlgorithmOptions": {
      "defaultValue": "RSA",
      "allowedValues": [
        "RSA"
      ]
    },
    "hashAlgorithmOptions": {
      "defaultValue": "SHA1",
      "allowedValues": [
        "SHA1"
      ]
    }
  },
  "maxSizeInMB": 16,
  "partner": "AmazonFreeRTOS",
  "signingImageFormat": {
    "defaultFormat": "JSONEmbedded",
    "supportedFormats": [
      "JSONEmbedded"
    ]
  }
}

```

```

    ]
  }
},
{
  "category": "AWS",
  "displayName": "Amazon FreeRTOS SHA256-ECDSA",
  "target": "SHA256-ECDSA",
  "platformId": "AmazonFreeRTOS-Default",
  "signingConfiguration": {
    "encryptionAlgorithmOptions": {
      "defaultValue": "ECDSA",
      "allowedValues": [
        "ECDSA"
      ]
    },
    "hashAlgorithmOptions": {
      "defaultValue": "SHA256",
      "allowedValues": [
        "SHA256"
      ]
    }
  },
  "maxSizeInMB": 16,
  "partner": "AmazonFreeRTOS",
  "signingImageFormat": {
    "defaultFormat": "JSONEmbedded",
    "supportedFormats": [
      "JSONEmbedded"
    ]
  }
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSigningPlatforms](#)中的。

## list-signing-profiles

以下代码示例演示了如何使用 list-signing-profiles。

### AWS CLI

#### 列出所有签名档案



以下list-signing-profiles示例显示了有关该账户所有签名资料的详细信息。

```
aws signer list-signing-profiles
```

输出：

```
{
  "profiles": [
    {
      "platformId": "AmazonFreeRTOS-TI-CC3220SF",
      "profileName": "MyProfile4",
      "status": "Active",
      "signingMaterial": {
        "certificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/6a55389b-306b-4e8c-a95c-0123456789abc"
      }
    },
    {
      "platformId": "AWSIoTDeviceManagement-SHA256-ECDSA",
      "profileName": "MyProfile5",
      "status": "Active",
      "signingMaterial": {
        "certificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/6a55389b-306b-4e8c-a95c-0123456789abc"
      }
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSigningProfiles](#)中的。

## put-signing-profile

以下代码示例演示了如何使用 put-signing-profile。

### AWS CLI

#### 创建签名档案

以下put-signing-profile示例使用指定的证书和平台创建签名配置文件。

```
aws signer put-signing-profile \
```

```
--profile-name MyProfile6 \  
--signing-material certificateArn=arn:aws:acm:us-  
west-2:123456789012:certificate/6a55389b-306b-4e8c-a95c-0123456789abc \  
--platform AmazonFreeRTOS-TI-CC3220SF
```

输出：

```
{  
  "arn": "arn:aws:signer:us-west-2:123456789012:/signing-profiles/MyProfile6"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutSigningProfile](#)中的。

## start-signing-job

以下代码示例演示了如何使用 start-signing-job。

### AWS CLI

#### 开始签名作业

以下start-signing-job示例对在指定源找到的代码启动签名作业。它使用指定的配置文件进行签名，并将签名的代码放置在指定的目的地。

```
aws signer start-signing-job \  
  --source 's3={bucketName=signer-  
source,key=MyCode.rb,version=PNyFaUTgsQh5ZdMCcoCe6pT1g0pgB_M4}' \  
  --destination 's3={bucketName=signer-destination,prefix=signed-}' \  
  --profile-name MyProfile7
```

输出是签名任务的 ID。

```
{  
  "jobId": "2065c468-73e2-4385-a6c9-0123456789abc"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartSigningJob](#)中的。

## 使用 Snowball 的示例 AWS CLI

以下代码示例向您展示了如何使用 with Snowball 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### get-snowball-usage

以下代码示例演示了如何使用 get-snowball-usage。

AWS CLI

获取有关您账户的 Snowball 服务限制的信息

以下get-snowball-usage示例显示有关您的账户的 Snowball 服务限制的信息，以及您的账户正在使用的 Snowball 数量。

```
aws snowball get-snowball-usage
```

输出：

```
{
  "SnowballLimit": 1,
  "SnowballsInUse": 0
}
```

有关更多信息，请参阅 [AWS Snowball 开发者指南中的 Snowball AWS 边缘限制](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSnowballUsage](#)中的。

## list-jobs

以下代码示例演示了如何使用 list-jobs。

### AWS CLI

列出你账户中当前的 Snowball 职位

以下list-jobs示例显示了一个JobListEntry对象数组。在此示例中，列出了单个作业。

```
aws snowball list-jobs
```

输出：

```
{
  "JobListEntries": [
    {
      "CreationDate": 2016-09-27T14:50Z,
      "Description": "Important Photos 2016-08-11",
      "IsMaster": TRUE,
      "JobId": "ABCd1e324fe-022f-488e-a98b-3b0566063db1",
      "JobState": "Complete",
      "JobType": "IMPORT",
      "SnowballType": "EDGE"
    }
  ]
}
```

有关更多信息，请参阅 Snowball [开发者 AWS 指南中的 Snowball Edge 设备任务](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListJobs](#)中的。

## 使用 Amazon SNS 示例 AWS CLI

以下代码示例向您展示了如何在 Amazon SNS 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)
- [场景](#)

## 操作

### **add-permission**

以下代码示例演示了如何使用 `add-permission`。

#### AWS CLI

为主题添加权限

以下 `add-permission` 示例添加了账户在 AWS 账户 987654321098 下使用指定主题的 `Publish` 操作的权限 123456789012。

```
aws sns add-permission \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \  
  --label Publish-Permission \  
  --aws-account-id 987654321098 \  
  --action-name Publish
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AddPermission](#) 中的。

### **check-if-phone-number-is-opted-out**

以下代码示例演示了如何使用 `check-if-phone-number-is-opted-out`。

#### AWS CLI

检查电话号码的 SMS 消息退出

以下 `check-if-phone-number-is-opted-out` 示例检查指定的电话号码是否已选择不接收来自当前 AWS 账户的 SMS 消息。

```
aws sns check-if-phone-number-is-opted-out \  
  --phone-number +1555550100
```

输出：

```
{  
  "isOptedOut": false  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CheckIfPhoneNumberIsOptedOut](#)中的。

## confirm-subscription

以下代码示例演示了如何使用 `confirm-subscription`。

### AWS CLI

#### 确认订阅

以下 `confirm-subscription` 命令将完成订阅名为 `my-topic` 的 SNS 主题时启动的确认过程。`--token` 参数来自发送到订阅调用中指定的通知端点的确认消息。

```
aws sns confirm-subscription \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \  
  --token  
2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7ce5d0cce1
```

输出：

```
{  
  "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-  
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ConfirmSubscription](#)中的。

## create-platform-application

以下代码示例演示了如何使用 `create-platform-application`。

## AWS CLI

### 创建平台应用程序

以下 `create-platform-application` 示例使用指定的平台凭据创建了一个 Google Firebase 平台应用程序。

```
aws sns create-platform-application \  
  --name MyApplication \  
  --platform GCM \  
  --attributes PlatformCredential=EXAMPLEabcd12345jklm67890stuv12345bcdef
```

输出：

```
{  
  "PlatformApplicationArn": "arn:aws:sns:us-west-2:123456789012:app/GCM/  
MyApplication"  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreatePlatformApplication](#) 中的。

## **create-topic**

以下代码示例演示了如何使用 `create-topic`。

## AWS CLI

### 创建 SNS 主题

以下 `create-topic` 示例将创建名为 `my-topic` 的 SNS 主题。

```
aws sns create-topic \  
  --name my-topic
```

输出：

```
{  
  "ResponseMetadata": {  
    "RequestId": "1469e8d7-1642-564e-b85d-a19b4b341f83"  
  },
```

```
"TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

有关更多信息，请参阅 [《AWS 命令行界面用户指南》](#) 中的在 Amazon SQS 和 Amazon SNS 中使用命令AWS 行界面。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTopic](#)中的。

## delete-endpoint

以下代码示例演示了如何使用 delete-endpoint。

### AWS CLI

删除平台应用程序终端节点

以下delete-endpoint示例删除了指定的平台应用程序终端节点。

```
aws sns delete-endpoint \  
  --endpoint-arn arn:aws:sns:us-west-2:123456789012:endpoint/GCM/  
  MyApplication/12345678-abcd-9012-efgh-345678901234
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteEndpoint](#)中的。

## delete-platform-application

以下代码示例演示了如何使用 delete-platform-application。

### AWS CLI

删除平台应用程序

以下delete-platform-application示例删除了指定的平台应用程序。

```
aws sns delete-platform-application \  
  --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/ADM/  
  MyApplication
```

此命令不生成任何输出。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePlatformApplication](#)中的。

## delete-topic

以下代码示例演示了如何使用 delete-topic。

### AWS CLI

#### 删除 SNS 主题

以下 delete-topic 示例将删除指定的 SNS 主题。

```
aws sns delete-topic \  
  --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTopic](#)中的。

## get-endpoint-attributes

以下代码示例演示了如何使用 get-endpoint-attributes。

### AWS CLI

#### 列出平台应用程序终端节点属性

以下get-endpoint-attributes示例列出了指定平台应用程序终端节点的属性。

```
aws sns get-endpoint-attributes \  
  --endpoint-arn arn:aws:sns:us-west-2:123456789012:endpoint/GCM/  
  MyApplication/12345678-abcd-9012-efgh-345678901234
```

输出：

```
{  
  "Attributes": {  
    "Enabled": "true",  
    "Token": "EXAMPLE12345..."  
  }  
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetEndpointAttributes](#)中的。

## get-platform-application-attributes

以下代码示例演示了如何使用 `get-platform-application-attributes`。

### AWS CLI

列出平台应用程序的属性

以下`get-platform-application-attributes`示例列出了指定平台应用程序的属性。

```
aws sns get-platform-application-attributes \  
  --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/MPNS/  
  MyApplication
```

输出：

```
{  
  "Attributes": {  
    "Enabled": "true",  
    "SuccessFeedbackSampleRate": "100"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPlatformApplicationAttributes](#)中的。

## get-sms-attributes

以下代码示例演示了如何使用 `get-sms-attributes`。

### AWS CLI

列出默认 SMS 消息属性

以下 `get-sms-attributes` 示例将列出发送 SMS 消息的默认属性。

```
aws sns get-sms-attributes
```

输出：

```
{
  "attributes": {
    "DefaultSenderId": "MyName"
  }
}
```

- 有关 API 详细信息，请参阅《AWS CLI Command Reference》中的 [GetSMSAttributes](#)。

## get-subscription-attributes

以下代码示例演示了如何使用 `get-subscription-attributes`。

### AWS CLI

检索主题的订阅属性

下面 `get-subscription-attributes` 显示了指定订阅的属性。你可以从 `list-subscriptions` 命令的输出中获取 `subscription-arn`。

```
aws sns get-subscription-attributes \
  --subscription-arn "arn:aws:sns:us-west-2:123456789012:my-
  topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
```

输出：

```
{
  "Attributes": {
    "Endpoint": "my-email@example.com",
    "Protocol": "email",
    "RawMessageDelivery": "false",
    "ConfirmationWasAuthenticated": "false",
    "Owner": "123456789012",
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
    topic:8a21d249-4329-4871-acc6-7be709c6ea7f",
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
  }
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetSubscriptionAttributes](#) 中的。

## get-topic-attributes

以下代码示例演示了如何使用 `get-topic-attributes`。

### AWS CLI

检索主题的属性

以下 `get-topic-attributes` 示例将显示指定主题的属性。

```
aws sns get-topic-attributes \
  --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

输出：

```
{
  "Attributes": {
    "SubscriptionsConfirmed": "1",
    "DisplayName": "my-topic",
    "SubscriptionsDeleted": "0",
    "EffectiveDeliveryPolicy": "{\"http\":{\"defaultHealthyRetryPolicy\":
{\\"minDelayTarget\":20,\\\"maxDelayTarget\":20,\\\"numRetries\":3,\\\"numMaxDelayRetries
\\\":0,\\\"numNoDelayRetries\\\":0,\\\"numMinDelayRetries\\\":0,\\\"backoffFunction\\\":\\\"linear
\\\"},\\\"disableSubscriptionOverrides\\\":false}}",
    "Owner": "123456789012",
    "Policy": "{\"Version\":\"2008-10-17\", \"Id\":\"__default_policy_ID\",
\\\"Statement\\\":[{\\"Sid\\\":\\\"__default_statement_ID\\\",\\\"Effect\\\":\\\"Allow\\\",\\\"Principal
\\\":{\\"AWS\\\":\\\"*\\\"},\\\"Action\\\":[\\\"SNS:Subscribe\\\",\\\"SNS:ListSubscriptionsByTopic
\\\",\\\"SNS:DeleteTopic\\\",\\\"SNS:GetTopicAttributes\\\",\\\"SNS:Publish\\\",
\\\"SNS:RemovePermission\\\",\\\"SNS:AddPermission\\\",\\\"SNS:SetTopicAttributes\\\"],
\\\"Resource\\\":\\\"arn:aws:sns:us-west-2:123456789012:my-topic\\\",\\\"Condition\\\":
{\\"StringEquals\\\":{\\"AWS:SourceOwner\\\":\\\"0123456789012\\\"}}]}]\",
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
    "SubscriptionsPending": "0"
  }
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetTopicAttributes](#) 中的。

## list-endpoints-by-platform-application

以下代码示例演示了如何使用 `list-endpoints-by-platform-application`。

## AWS CLI

### 列出平台应用程序的终端节点

以下 `list-endpoints-by-platform-application` 示例列出了指定平台应用程序的终端节点和终端节点属性。

```
aws sns list-endpoints-by-platform-application \
  --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
  MyApplication
```

输出：

```
{
  "Endpoints": [
    {
      "Attributes": {
        "Token": "EXAMPLE12345...",
        "Enabled": "true"
      },
      "EndpointArn": "arn:aws:sns:us-west-2:123456789012:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListEndpointsByPlatformApplication](#) 中的。

## list-phone-numbers-opted-out

以下代码示例演示了如何使用 `list-phone-numbers-opted-out`。

### AWS CLI

#### 列出 SMS 消息退出

以下 `list-phone-numbers-opted-out` 示例将列出退出 SMS 消息接收的电话号码。

```
aws sns list-phone-numbers-opted-out
```

输出：

```
{
  "phoneNumbers": [
    "+15555550100"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPhoneNumbersOptedOut](#)中的。

## list-platform-applications

以下代码示例演示了如何使用 `list-platform-applications`。

### AWS CLI

列出平台应用程序

以下`list-platform-applications`示例列出了 ADM 和 MPNS 的平台应用程序。

```
aws sns list-platform-applications
```

输出：

```
{
  "PlatformApplications": [
    {
      "PlatformApplicationArn": "arn:aws:sns:us-west-2:123456789012:app/ADM/MyApplication",
      "Attributes": {
        "SuccessFeedbackSampleRate": "100",
        "Enabled": "true"
      }
    },
    {
      "PlatformApplicationArn": "arn:aws:sns:us-west-2:123456789012:app/MPNS/MyOtherApplication",
      "Attributes": {
        "SuccessFeedbackSampleRate": "100",
        "Enabled": "true"
      }
    }
  ]
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListPlatformApplications](#)中的。

## list-subscriptions-by-topic

以下代码示例演示了如何使用 `list-subscriptions-by-topic`。

### AWS CLI

列出与主题关联的订阅

以下内容`list-subscriptions-by-topic`检索与指定主题关联的 SNS 订阅列表。

```
aws sns list-subscriptions-by-topic \  
  --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

输出：

```
{  
  "Subscriptions": [  
    {  
      "Owner": "123456789012",  
      "Endpoint": "my-email@example.com",  
      "Protocol": "email",  
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",  
      "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-  
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSubscriptionsByTopic](#)中的。

## list-subscriptions

以下代码示例演示了如何使用 `list-subscriptions`。

### AWS CLI

列出 SNS 订阅

以下 `list-subscriptions` 示例显示了您 AWS 账户中的 SNS 订阅列表。

```
aws sns list-subscriptions
```

输出：

```
{
  "Subscriptions": [
    {
      "Owner": "123456789012",
      "Endpoint": "my-email@example.com",
      "Protocol": "email",
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
      "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListSubscriptions](#) 中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

AWS CLI

列出主题的标签

以下 `list-tags-for-resource` 示例列出了指定 Amazon SNS 主题的标签。

```
aws sns list-tags-for-resource \
  --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic
```

输出：

```
{
  "Tags": [
    {
      "Key": "Team",
      "Value": "Alpha"
    }
  ]
}
```



```
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## list-topics

以下代码示例演示了如何使用 `list-topics`。

### AWS CLI

列出 SNS 主题

以下`list-topics`示例列出了您 AWS 账户中的所有 SNS 主题。

```
aws sns list-topics
```

输出：

```
{  
  "Topics": [  
    {  
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"  
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTopics](#)中的。

## opt-in-phone-number

以下代码示例演示了如何使用 `opt-in-phone-number`。

### AWS CLI

选择接收 SMS 消息

以下`opt-in-phone-number`示例选择指定的电话号码接收 SMS 消息。

```
aws sns opt-in-phone-number \  

```

```
--phone-number +15555550100
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[OptInPhoneNumber](#)中的。

## publish

以下代码示例演示了如何使用 publish。

### AWS CLI

#### 示例 1：向主题发布消息

以下 publish 示例将指定消息发布到指定 SNS 主题。该消息来自一个文本文件，您可以在该文件中包含换行符。

```
aws sns publish \  
  --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic" \  
  --message file://message.txt
```

message.txt 的内容：

```
Hello World  
Second Line
```

输出：

```
{  
  "MessageId": "123a45b6-7890-12c3-45d6-111122223333"  
}
```

#### 示例 2：向电话号码发布 SMS 消息

以下 publish 示例将消息 Hello world! 发布到电话号码 +1-555-555-0100。

```
aws sns publish \  
  --message "Hello world!" \  
  --phone-number +1-555-555-0100
```

输出：

```
{
  "MessageId": "123a45b6-7890-12c3-45d6-333322221111"
}
```

- 有关 API 详细信息，请参阅《AWS CLI Command Reference》中的 [Publish](#)。

## put-data-protection-policy

以下代码示例演示了如何使用 put-data-protection-policy。

### AWS CLI

#### 设置数据保护政策

示例 1：拒绝发布者使用发布消息 CreditCardNumber

以下put-data-protection-policy示例拒绝发布者使用发布消息 CreditCardNumber。

```
aws sns put-data-protection-policy \
  --resource-arn arn:aws:sns:us-east-1:123456789012:mytopic \
  --data-protection-policy "{\"Name\":\"data_protection_policy\",\"Description\":"
  "\":\"Example data protection policy\",\"Version\":\"2021-06-01\",\"Statement\":"
  "\":[{\":\"DataDirection\":\"Inbound\",\":\"Principal\":[\"*\"]\",\":\"DataIdentifier\":"
  "\":[\"arn:aws:dataprotection::aws:data-identifier/CreditCardNumber\"],\":\"Operation\":"
  "\":{\":\"Deny\":{}}}]\"}
```

此命令不生成任何输出。

示例 2：从文件加载参数

以下内容从文件put-data-protection-policy加载参数。

```
aws sns put-data-protection-policy \
  --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
  --data-protection-policy file://policy.json
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutDataProtectionPolicy](#)中的。

## remove-permission

以下代码示例演示了如何使用 `remove-permission`。

### AWS CLI

从主题中移除权限

以下 `remove-permission` 示例 `Publish-Permission` 从指定主题中移除权限。

```
aws sns remove-permission \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \  
  --label Publish-Permission
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RemovePermission](#) 中的。

## set-endpoint-attributes

以下代码示例演示了如何使用 `set-endpoint-attributes`。

### AWS CLI

设置端点属性

以下 `set-endpoint-attributes` 示例禁用了指定的平台应用程序终端节点。

```
aws sns set-endpoint-attributes \  
  --endpoint-arn arn:aws:sns:us-west-2:123456789012:endpoint/GCM/  
MyApplication/12345678-abcd-9012-efgh-345678901234 \  
  --attributes Enabled=false
```

输出：

```
{  
  "Attributes": {  
    "Enabled": "false",  
    "Token": "EXAMPLE12345..."  
  }  
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SetEndpointAttributes](#) 中的。

## set-platform-application-attributes

以下代码示例演示了如何使用 `set-platform-application-attributes`。

### AWS CLI

设置平台应用程序属性

以下 `set-platform-application-attributes` 示例将指定平台应用程序的 `EventDeliveryFailure` 属性设置为指定 Amazon SNS 主题的 ARN。

```
aws sns set-platform-application-attributes \  
  --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/  
MyApplication \  
  --attributes EventDeliveryFailure=arn:aws:sns:us-  
west-2:123456789012:AnotherTopic
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SetPlatformApplicationAttributes](#) 中的。

## set-sms-attributes

以下代码示例演示了如何使用 `set-sms-attributes`。

### AWS CLI

设置 SMS 消息属性

以下 `set-sms-attributes` 示例将 SMS 消息的默认发件人 ID 设置为 `MyName`。

```
aws sns set-sms-attributes \  
  --attributes DefaultSenderId=MyName
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI Command Reference》中的 [SetSMSAttributes](#)。

## set-subscription-attributes

以下代码示例演示了如何使用 `set-subscription-attributes`。

## AWS CLI

### 设置订阅属性

以下 `set-subscription-attributes` 示例将 `RawMessageDelivery` 属性设置为 SQS 订阅。

```
aws sns set-subscription-attributes \  
  --subscription-arn arn:aws:sns:us-  
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \  
  --attribute-name RawMessageDelivery \  
  --attribute-value true
```

此命令不生成任何输出。

以下 `set-subscription-attributes` 示例将 `FilterPolicy` 属性设置为 SQS 订阅。

```
aws sns set-subscription-attributes \  
  --subscription-arn arn:aws:sns:us-  
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \  
  --attribute-name FilterPolicy \  
  --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

此命令不生成任何输出。

以下 `set-subscription-attributes` 示例从 SQS 订阅中移除 `FilterPolicy` 属性。

```
aws sns set-subscription-attributes \  
  --subscription-arn arn:aws:sns:us-  
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \  
  --attribute-name FilterPolicy \  
  --attribute-value "{}"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SetSubscriptionAttributes](#) 中的。

### set-topic-attributes

以下代码示例演示了如何使用 `set-topic-attributes`。

## AWS CLI

### 为主题设置属性

以下 `set-topic-attributes` 示例为指定主题设置 `DisplayName` 属性。

```
aws sns set-topic-attributes \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \  
  --attribute-name DisplayName \  
  --attribute-value MyTopicDisplayName
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [SetTopicAttributes](#) 中的。

## subscribe

以下代码示例演示了如何使用 `subscribe`。

## AWS CLI

### 订阅主题

以下 `subscribe` 命令将电子邮件地址订阅到指定主题。

```
aws sns subscribe \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \  
  --protocol email \  
  --notification-endpoint my-email@example.com
```

输出：

```
{  
  "SubscriptionArn": "pending confirmation"  
}
```

- 有关 API 详细信息，请参阅《AWS CLI Command Reference》中的 [Subscribe](#)。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

## AWS CLI

### 为主题添加标签

以下 `tag-resource` 示例将元数据标签添加到指定 Amazon SNS 主题。

```
aws sns tag-resource \  
  --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \  
  --tags Key=Team,Value=Alpha
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## unsubscribe

以下代码示例演示了如何使用 `unsubscribe`。

## AWS CLI

### 从主题取消订阅

以下 `unsubscribe` 示例将从主题删除指定的订阅。

```
aws sns unsubscribe \  
  --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-  
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI Command Reference》中的 [Unsubscribe](#)。

## untag-resource

以下代码示例演示了如何使用 `untag-resource`。

## AWS CLI

### 从主题中移除标签

以下 `untag-resource` 示例从指定的 Amazon SNS 主题中删除所有带有指定密钥的标签。

```
aws sns untag-resource \  
  --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic
```



```
--resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \  
--tag-keys Team
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## 场景

### 为推送通知创建平台终端节点

以下代码示例展示如何为 Amazon SNS 推送通知创建平台终端节点。

## AWS CLI

### 创建平台应用程序端点

以下 `create-platform-endpoint` 示例使用指定令牌为指定平台应用程序创建端点。

```
aws sns create-platform-endpoint \  
  --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/  
  MyApplication \  
  --token EXAMPLE12345...
```

输出：

```
{  
  "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/  
  MyApplication/12345678-abcd-9012-efgh-345678901234"  
}
```

## 使用亚马逊 SQS 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon SQS 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### **add-permission**

以下代码示例演示了如何使用 `add-permission`。

#### AWS CLI

向队列添加权限

此示例允许指定 AWS 账户向指定队列发送消息。

命令:

```
aws sqs add-permission --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --label SendMessageFromMyQueue --aws-account-ids 12345EXAMPLE --actions SendMessage
```

输出:

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddPermission](#)中的。

### **cancel-message-move-task**

以下代码示例演示了如何使用 `cancel-message-move-task`。

#### AWS CLI

取消邮件移动任务

以下`cancel-message-move-task`示例取消了指定的邮件移动任务。

```
aws sqs cancel-message-move-task \
```

```
--task-handle AQEB6nR4...Hz1vZQ==
```

输出：

```
{
  "ApproximateNumberOfMessagesMoved": 102
}
```

有关更多信息，请参阅开发者指南中的 [Amazon SQS API 权限：操作和资源参考](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CancelMessageMoveTask](#) 中的。

## change-message-visibility-batch

以下代码示例演示了如何使用 change-message-visibility-batch。

### AWS CLI

批量更改多条消息的超时可见性

此示例将 2 条指定消息的超时可见性更改为 10 小时 (10 小时 x 60 分钟 \* 60 秒)。

命令：

```
aws sqs change-message-visibility-batch --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --entries file://change-message-visibility-batch.json
```

输入文件 (change-message-visibility-batch.json)：

```
[
  {
    "Id": "FirstMessage",
    "ReceiptHandle": "AQEBhz2q...Jf3kaw==",
    "VisibilityTimeout": 36000
  },
  {
    "Id": "SecondMessage",
    "ReceiptHandle": "AQEBkTUH...HifSnw==",
    "VisibilityTimeout": 36000
  }
]
```

输出：

```
{
  "Successful": [
    {
      "Id": "SecondMessage"
    },
    {
      "Id": "FirstMessage"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ChangeMessageVisibilityBatch](#)中的。

## change-message-visibility

以下代码示例演示了如何使用 change-message-visibility。

### AWS CLI

#### 更改消息的超时可见性

此示例将指定消息的超时可见性更改为 10 小时（10 小时 \* 60 分钟 \* 60 秒）。

命令：

```
aws sqs change-message-visibility --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --receipt-handle AQEBTpyI...t6HyQg== --visibility-timeout 36000
```

输出：

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ChangeMessageVisibility](#)中的。

## create-queue

以下代码示例演示了如何使用 create-queue。

## AWS CLI

### 创建队列

此示例使用指定名称创建队列，将消息保留期设置为 3 天（3 天 \* 24 小时 \* 60 分钟 \* 60 秒），并将队列的死信队列设置为指定队列，其最大接收数量为 1,000 条消息。

命令:

```
aws sqs create-queue --queue-name MyQueue --attributes file://create-queue.json
```

输入文件 ( create-queue.json ) :

```
{
  "RedrivePolicy": "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:80398EXAMPLE:MyDeadLetterQueue\", \"maxReceiveCount\": \"1000\"}\",
  "MessageRetentionPeriod": "259200"
}
```

输出 :

```
{
  "QueueUrl": "https://queue.amazonaws.com/80398EXAMPLE/MyQueue"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateQueue](#)中的。

### delete-message-batch

以下代码示例演示了如何使用 delete-message-batch。

## AWS CLI

### 批量删除多条消息

此示例将删除指定消息。

命令:

```
aws sqs delete-message-batch --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --entries file://delete-message-batch.json
```

输入文件 (delete-message-batch.json) :

```
[
  {
    "Id": "FirstMessage",
    "ReceiptHandle": "AQEB1mg1...Z4GuLw=="
  },
  {
    "Id": "SecondMessage",
    "ReceiptHandle": "AQEBLsYM...VQubAA=="
  }
]
```

输出 :

```
{
  "Successful": [
    {
      "Id": "FirstMessage"
    },
    {
      "Id": "SecondMessage"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMessageBatch](#)中的。

## delete-message

以下代码示例演示了如何使用 delete-message。

### AWS CLI

#### 删除消息

此示例将删除指定消息。

命令:

```
aws sqs delete-message --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --receipt-handle AQEBRXTo...q2doVA==
```

输出：

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMessage](#)中的。

## **delete-queue**

以下代码示例演示了如何使用 delete-queue。

### AWS CLI

删除队列

此示例将删除指定队列。

命令：

```
aws sqs delete-queue --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyNewerQueue
```

输出：

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteQueue](#)中的。

## **get-queue-attributes**

以下代码示例演示了如何使用 get-queue-attributes。

### AWS CLI

获取队列的属性

此示例将获取指定队列的所有属性。

命令：

```
aws sqs get-queue-attributes --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --attribute-names All
```

输出：

```
{
  "Attributes": {
    "ApproximateNumberOfMessagesNotVisible": "0",
    "RedrivePolicy": "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:80398EXAMPLE:MyDeadLetterQueue\",\"maxReceiveCount\":1000}",
    "MessageRetentionPeriod": "345600",
    "ApproximateNumberOfMessagesDelayed": "0",
    "MaximumMessageSize": "262144",
    "CreatedTimestamp": "1442426968",
    "ApproximateNumberOfMessages": "0",
    "ReceiveMessageWaitTimeSeconds": "0",
    "DelaySeconds": "0",
    "VisibilityTimeout": "30",
    "LastModifiedTimestamp": "1442426968",
    "QueueArn": "arn:aws:sqs:us-east-1:80398EXAMPLE:MyNewQueue"
  }
}
```

此示例仅获取指定队列的最大消息大小和可见性超时属性。

命令：

```
aws sqs get-queue-attributes --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyNewQueue --attribute-names MaximumMessageSize
VisibilityTimeout
```

输出：

```
{
  "Attributes": {
    "VisibilityTimeout": "30",
    "MaximumMessageSize": "262144"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetQueueAttributes](#)中的。

## get-queue-url

以下代码示例演示了如何使用 get-queue-url。



## AWS CLI

### 获取队列 URL

此示例将获取指定队列的 URL。

命令:

```
aws sqs get-queue-url --queue-name MyQueue
```

输出:

```
{
  "QueueUrl": "https://queue.amazonaws.com/80398EXAMPLE/MyQueue"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetQueueUrl](#)中的。

## list-dead-letter-source-queues

以下代码示例演示了如何使用 list-dead-letter-source-queues。

## AWS CLI

### 列出死信来源队列

此示例列出了与指定死信源队列关联的队列。

命令:

```
aws sqs list-dead-letter-source-queues --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

输出:

```
{
  "queueUrls": [
    "https://queue.amazonaws.com/80398EXAMPLE/MyQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyOtherQueue"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDeadLetterSourceQueues](#)中的。

## list-message-move-tasks

以下代码示例演示了如何使用 `list-message-move-tasks`。

### AWS CLI

#### 列出邮件移动任务

以下`list-message-move-tasks`示例列出了指定队列中最近的 2 个邮件移动任务。

```
aws sqs list-message-move-tasks \  
  --source-arn arn:aws:sqs:us-west-2:80398EXAMPLE:MyQueue \  
  --max-results 2
```

#### 输出：

```
{  
  "Results": [  
    {  
      "TaskHandle": "AQEB6nR4...HzlvZQ==",  
      "Status": "RUNNING",  
      "SourceArn": "arn:aws:sqs:us-west-2:80398EXAMPLE:MyQueue1",  
      "DestinationArn": "arn:aws:sqs:us-west-2:80398EXAMPLE:MyQueue2",  
      "MaximumNumberOfMessagesPerSecond": 50,  
      "ApproximateNumberOfMessagesMoved": 203,  
      "ApproximateNumberOfMessagesToMove": 30,  
      "StartedTimestamp": 1442428276921  
    },  
    {  
      "Status": "COMPLETED",  
      "SourceArn": "arn:aws:sqs:us-west-2:80398EXAMPLE:MyQueue1",  
      "DestinationArn": "arn:aws:sqs:us-west-2:80398EXAMPLE:MyQueue2",  
      "ApproximateNumberOfMessagesMoved": 29,  
      "ApproximateNumberOfMessagesToMove": 0,  
      "StartedTimestamp": 1342428272093  
    }  
  ]  
}
```

有关更多信息，请参阅开发者指南中的 [Amazon SQS API 权限：操作和资源参考](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListMessageMoveTasks](#)中的。

## list-queue-tags

以下代码示例演示了如何使用 list-queue-tags。

### AWS CLI

列出队列的所有成本分配标签

以下list-queue-tags示例显示了与指定队列关联的所有成本分配标签。

```
aws sqs list-queue-tags \  
  --queue-url https://sqs.us-west-2.amazonaws.com/123456789012/MyQueue
```

输出：

```
{  
  "Tags": {  
    "Team": "Alpha"  
  }  
}
```

有关更多信息，请参阅《亚马逊简单队列服务开发者指南》中的[列出成本分配标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListQueueTags](#)中的。

## list-queues

以下代码示例演示了如何使用 list-queues。

### AWS CLI

列出队列

此示例将列出所有队列。

命令：

```
aws sqs list-queues
```

输出：

```
{
  "QueueUrls": [
    "https://queue.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyOtherQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/TestQueue1",
    "https://queue.amazonaws.com/80398EXAMPLE/TestQueue2"
  ]
}
```

此示例仅列出以“My”开头的队列。

命令：

```
aws sqs list-queues --queue-name-prefix My
```

输出：

```
{
  "QueueUrls": [
    "https://queue.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyOtherQueue"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListQueues](#)中的。

## purge-queue

以下代码示例演示了如何使用 purge-queue。

### AWS CLI

#### 清除队列

此示例删除指定队列中的所有消息。

命令：

```
aws sqs purge-queue --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyNewQueue
```

输出：

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PurgeQueue](#)中的。

## receive-message

以下代码示例演示了如何使用 `receive-message`。

### AWS CLI

#### 接收消息

此示例最多接收 10 条可用消息，返回所有可用属性。

命令：

```
aws sqs receive-message --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --attribute-names All --message-attribute-names All --max-number-of-messages 10
```

输出：

```
{
  "Messages": [
    {
      "Body": "My first message.",
      "ReceiptHandle": "AQEBzbVv...fqNzFw==",
      "MD5ofBody": "1000f835...a35411fa",
      "MD5ofMessageAttributes": "9424c491...26bc3ae7",
      "MessageId": "d6790f8d-d575-4f01-bc51-40122EXAMPLE",
      "Attributes": {
        "ApproximateFirstReceiveTimestamp": "1442428276921",
        "SenderId": "AIDAIKMSNQ7EXAMPLE",
        "ApproximateReceiveCount": "5",
        "SentTimestamp": "1442428276921"
      }
    },
  ],
}
```

```
"MessageAttributes": {
  "PostalCode": {
    "DataType": "String",
    "StringValue": "ABC123"
  },
  "City": {
    "DataType": "String",
    "StringValue": "Any City"
  }
}
]
```

此示例接收下一条可用消息，仅返回 SenderId 和 SentTimestamp 属性以及 m PostalCode message 属性。

命令:

```
aws sqs receive-message --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --attribute-names SenderId SentTimestamp
--message-attribute-names PostalCode
```

输出:

```
{
  "Messages": [
    {
      "Body": "My first message.",
      "ReceiptHandle": "AQEB6nR4...HzlvZQ==",
      "MD5ofBody": "1000f835...a35411fa",
      "MD5ofMessageAttributes": "b8e89563...e088e74f",
      "MessageId": "d6790f8d-d575-4f01-bc51-40122EXAMPLE",
      "Attributes": {
        "SenderId": "AIDAIAZKMSNQ7TEXAMPLE",
        "SentTimestamp": "1442428276921"
      },
      "MessageAttributes": {
        "PostalCode": {
          "DataType": "String",
          "StringValue": "ABC123"
        }
      }
    }
  ]
}
```

```
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReceiveMessage](#)中的。

## remove-permission

以下代码示例演示了如何使用 `remove-permission`。

### AWS CLI

#### 移除权限

此示例从指定队列中移除带有指定标签的权限。

#### 命令:

```
aws sqs remove-permission --queue-url https://sqs.us-  
east-1.amazonaws.com/80398EXAMPLE/MyQueue --label SendMessagesFromMyQueue
```

#### 输出 :

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemovePermission](#)中的。

## send-message-batch

以下代码示例演示了如何使用 `send-message-batch`。

### AWS CLI

#### 批量发送多条消息

此示例向指定队列发送 2 条具有指定消息正文、延迟时间和消息属性的消息。

#### 命令:

```
aws sqs send-message-batch --queue-url https://sqs.us-  
east-1.amazonaws.com/80398EXAMPLE/MyQueue --entries file://send-message-batch.json
```

输入文件 (send-message-batch.json) :

```
[
  {
    "Id": "FuelReport-0001-2015-09-16T140731Z",
    "MessageBody": "Fuel report for account 0001 on 2015-09-16 at 02:07:31 PM.",
    "DelaySeconds": 10,
    "MessageAttributes": {
      "SellerName": {
        "DataType": "String",
        "StringValue": "Example Store"
      },
      "City": {
        "DataType": "String",
        "StringValue": "Any City"
      },
      "Region": {
        "DataType": "String",
        "StringValue": "WA"
      },
      "PostalCode": {
        "DataType": "String",
        "StringValue": "99065"
      },
      "PricePerGallon": {
        "DataType": "Number",
        "StringValue": "1.99"
      }
    }
  },
  {
    "Id": "FuelReport-0002-2015-09-16T140930Z",
    "MessageBody": "Fuel report for account 0002 on 2015-09-16 at 02:09:30 PM.",
    "DelaySeconds": 10,
    "MessageAttributes": {
      "SellerName": {
        "DataType": "String",
        "StringValue": "Example Fuels"
      },
      "City": {
        "DataType": "String",
        "StringValue": "North Town"
      },
      "Region": {
```



```

        "DataType": "String",
        "StringValue": "WA"
    },
    "PostalCode": {
        "DataType": "String",
        "StringValue": "99123"
    },
    "PricePerGallon": {
        "DataType": "Number",
        "StringValue": "1.87"
    }
}
]

```

输出：

```

{
  "Successful": [
    {
      "MD50fMessageBody": "203c4a38...7943237e",
      "MD50fMessageAttributes": "10809b55...baf283ef",
      "Id": "FuelReport-0001-2015-09-16T140731Z",
      "MessageId": "d175070c-d6b8-4101-861d-adeb3EXAMPLE"
    },
    {
      "MD50fMessageBody": "2cf0159a...c1980595",
      "MD50fMessageAttributes": "55623928...ae354a25",
      "Id": "FuelReport-0002-2015-09-16T140930Z",
      "MessageId": "f9b7d55d-0570-413e-b9c5-a9264EXAMPLE"
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SendMessageBatch](#)中的。

## send-message

以下代码示例演示了如何使用 send-message。

### AWS CLI

#### 发送邮件

此示例向指定队列发送一条具有指定消息正文、延迟时间和消息属性的消息。

命令:

```
aws sqs send-message --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --message-body "Information about the largest city in Any Region." --delay-seconds 10 --message-attributes file://send-message.json
```

输入文件 ( send-message.json ) :

```
{
  "City": {
    "DataType": "String",
    "StringValue": "Any City"
  },
  "Greeting": {
    "DataType": "Binary",
    "BinaryValue": "Hello, World!"
  },
  "Population": {
    "DataType": "Number",
    "StringValue": "1250800"
  }
}
```

输出 :

```
{
  "MD5ofMessageBody": "51b0a325...39163aa0",
  "MD5ofMessageAttributes": "00484c68...59e48f06",
  "MessageId": "da68f62c-0c07-4bee-bf5f-7e856EXAMPLE"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SendMessage](#)中的。

## set-queue-attributes

以下代码示例演示了如何使用 set-queue-attributes。

AWS CLI

设置队列属性

此示例将指定队列的传输延迟设置为 10 秒，最大消息大小为 128 KB ( 128 KB \* 1,024 字节 )，消息保留期为 3 天 ( 3 天 \* 24 小时 \* 60 分钟 \* 60 秒 )，接收消息等待时间为 20 秒，默认可见性超时为 60 秒。此示例还将指定的死信队列与最大接收数 1,000 条消息相关联。

命令:

```
aws sqs set-queue-attributes --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyNewQueue --attributes file://set-queue-attributes.json
```

输入文件 (set-queue-attributes.json) :

```
{
  "DelaySeconds": "10",
  "MaximumMessageSize": "131072",
  "MessageRetentionPeriod": "259200",
  "ReceiveMessageWaitTimeSeconds": "20",
  "RedrivePolicy": "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:80398EXAMPLE:MyDeadLetterQueue\",\"maxReceiveCount\":\"1000\"}",
  "VisibilityTimeout": "60"
}
```

输出 :

```
None.
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[SetQueueAttributes](#)中的。

## start-message-move-task

以下代码示例演示了如何使用 start-message-move-task。

### AWS CLI

示例 1 : \*启动消息移动任务\*

以下start-message-move-task示例启动消息移动任务，将消息从指定的死信队列重新驱动到源队列。

```
aws sqs start-message-move-task \
```

```
--source-arn arn:aws:sqs:us-west-2:80398EXAMPLE:MyQueue
```

输出：

```
{
  "TaskHandle": "AQEB6nR4...Hz1vZQ=="
}
```

有关更多信息，请参阅[指南名称中的主题标题](#)。

示例 2：\*以最大速率启动邮件移动任务\*

以下start-message-move-task示例启动消息移动任务，以每秒 50 条消息的最大速率将消息从指定的死信队列重新驱动到指定的目标队列。

```
aws sqs start-message-move-task \
  --source-arn arn:aws:sqs:us-west-2:80398EXAMPLE:MyQueue1 \
  --destination-arn arn:aws:sqs:us-west-2:80398EXAMPLE:MyQueue2 \
  --max-number-of-messages-per-second 50
```

输出：

```
{
  "TaskHandle": "AQEB6nR4...Hz1vZQ=="
}
```

有关更多信息，请参阅开发者指南中的 [Amazon SQS API 权限：操作和资源参考](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartMessageMoveTask](#)中的。

## tag-queue

以下代码示例演示了如何使用 tag-queue。

### AWS CLI

向队列添加成本分配标签

以下tag-queue示例将成本分配标签添加到指定的 Amazon SQS 队列。

```
aws sqs tag-queue \
```

```
--queue-url https://sqs.us-west-2.amazonaws.com/123456789012/MyQueue \  
--tags Priority=Highest
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon 简单队列服务开发者指南》中的[添加成本分配标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[TagQueue](#)中的。

## untag-queue

以下代码示例演示了如何使用 untag-queue。

### AWS CLI

从队列中移除成本分配标签

以下untag-queue示例从指定的 Amazon SQS 队列中移除成本分配标签。

```
aws sqs untag-queue \  
--queue-url https://sqs.us-west-2.amazonaws.com/123456789012/MyQueue \  
--tag-keys "Priority"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon 简单队列服务开发者指南》中的[添加成本分配标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagQueue](#)中的。

## 使用 Storage Gateway 示例 AWS CLI

以下代码示例向您展示了如何使用 with Storage Gateway 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### **describe-gateway-information**

以下代码示例演示了如何使用 `describe-gateway-information`。

#### AWS CLI

##### 描述网关

以下 `describe-gateway-information` 命令返回有关指定网关的元数据。要指定要描述的网关，请在命令中使用网关的 Amazon 资源名称 (ARN)。

以下示例指定了账号 `sgw-12A3456B` 中带有 ID 的网关 `123456789012`：

```
aws storagegateway describe-gateway-information --gateway-arn
"arn:aws:storagegateway:us-west-2:123456789012:gateway/sgw-12A3456B"
```

此命令输出一个 JSON 块，其中包含有关网关的元数据，例如网关名称、网络接口、配置的时区和状态（网关是否正在运行）。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeGatewayInformation](#) 中的。

### **list-file-shares**

以下代码示例演示了如何使用 `list-file-shares`。

#### AWS CLI

##### 列出文件共享

以下 `command-name` 示例列出了您 AWS 账户中可用的微件。

```
aws storagegateway list-file-shares \
--gateway-arn arn:aws:storagegateway:us-east-1:209870788375:gateway/sgw-FB02E292
```

输出：

```
{
  "FileShareInfoList": [
    {
      "FileShareType": "NFS",
      "FileShareARN": "arn:aws:storagegateway:us-east-1:111122223333:share/
share-2FA12345",
      "FileShareId": "share-2FA12345",
      "FileShareStatus": "AVAILABLE",
      "GatewayARN": "arn:aws:storagegateway:us-east-1:111122223333:gateway/
sgw-FB0AAAAA"
    }
  ],
  "Marker": null
}
```

有关更多信息，请参阅 AWS Storage Gateway 服务 API 参考[ListFileShares](#)中的。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListFileShares](#)中的。

## list-gateways

以下代码示例演示了如何使用 list-gateways。

### AWS CLI

列出账户的网关

以下list-gateways命令列出了为账户定义的所有网关：

```
aws storagegateway list-gateways
```

此命令输出一个 JSON 块，其中包含网关 Amazon 资源名称 (ARN) 列表。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListGateways](#)中的。

## list-volumes

以下代码示例演示了如何使用 list-volumes。

### AWS CLI

列出为网关配置的卷

以下`list-volumes`命令返回为指定网关配置的卷列表。要指定要描述的网关，请在命令中使用网关的 Amazon 资源名称 (ARN)。

以下示例指定了账号`sgw-12A3456B`中带有 ID 的网关`123456789012`：

```
aws storagegateway list-volumes --gateway-arn "arn:aws:storagegateway:us-west-2:123456789012:gateway/sgw-12A3456B"
```

此命令输出一个 JSON 块，其中包含包含每个卷的类型和 ARN 的卷列表。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListVolumes](#)中的。

## refresh-cache

以下代码示例演示了如何使用 `refresh-cache`。

### AWS CLI

刷新文件共享缓存

以下`refresh-cache`示例刷新了指定文件共享的缓存。

```
aws storagegateway refresh-cache \  
  --file-share-arn arn:aws:storagegateway:us-east-1:111122223333:share/  
share-2FA12345
```

输出：

```
{  
  "FileShareARN": "arn:aws:storagegateway:us-east-1:111122223333:share/  
share-2FA12345",  
  "NotificationId": "4954d4b1-abcd-ef01-1234-97950a7d3483"  
}
```

有关更多信息，请参阅 AWS Storage Gateway 服务 API 参考[ListFileShares](#)中的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RefreshCache](#)中的。

## AWS STS 使用示例 AWS CLI

以下代码示例向您展示了如何使用`with`来执行操作和实现常见场景 AWS STS。AWS Command Line Interface



操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **assume-role-with-saml**

以下代码示例演示了如何使用 `assume-role-with-saml`。

AWS CLI

获取通过 SAML 进行身份验证的角色的短期证书

以下 `assume-role-with-saml` 命令将检索 IAM 角色 `TestSaml` 的一组短期凭证。本示例中的请求是在您向其进行身份验证时使用身份提供商提供的 SAML 断言进行身份验证的。

```
aws sts assume-role-with-saml \  
  --role-arn arn:aws:iam::123456789012:role/TestSaml \  
  --principal-arn arn:aws:iam::123456789012:saml-provider/SAML-test \  
  --saml-assertion  
  "VERYLONGENCODEDASSERTIONEXAMPLExyzYW1s0kF1ZG11bmN1PmJsYW5rPC9zYW1s0kF1ZG11bmN1Pjwvc2FtbDpBd  
+PHNhbWw6TmFtZU1EIEZvcmlhdD0idXJu0m9hc2lz0m5hbWVz0nRj01NBTUw6Mi4w0m5hbWVpZC1mb3JtYXQ6dHJhbnN  
+PHNhbWw6U3ViamVjdENvbmZpcm1hdGlvb2lnbnZXR0b2Q9InVybjpvYXNpczpuYW1lc2p0YzptQU1MOjIuMDpjbTpiZWw"
```

输出：

```
{  
  "Issuer": "https://integ.example.com/idp/shibboleth</Issuer",  
  "AssumedRoleUser": {  
    "Arn": "arn:aws:sts::123456789012:assumed-role/TestSaml",  
    "AssumedRoleId": "AR0456EXAMPLE789:TestSaml"  
  },  
  "Credentials": {
```

```

    "AccessKeyId": "ASIAV3ZUEFP6EXAMPLE",
    "SecretAccessKey": "8P+SQvWIuLnKhh8d++jpw0nNmQRBZvNEXAMPLEKEY",
    "SessionToken": "IQoJb3JpZ2luX2VjE0z//////////
wEXAMPLEtMSJHMEUCIDoKK3JH9uGQE1z0sINr5M4jk
+Na8KHDcCYRVjJCZEv0AiEA30vJGtw1EcVi0leS2vhs8VdCKFJQWPQrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburED
+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
+scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSYvKTr4rvx3iSI1TJabIQwj2ICCR/oLxBA==" ,
    "Expiration": "2019-11-01T20:26:47Z"
  },
  "Audience": "https://signin.aws.amazon.com/saml",
  "SubjectType": "transient",
  "PackedPolicySize": "6",
  "NameQualifier": "SbdG0nUkh1i4+EXAMPLExL/jEvs=",
  "Subject": "SamlExample"
}

```

有关更多信息，请参阅《AWS IAM 用户指南》中的[请求临时安全凭证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssumeRoleWithSaml](#)中的。

## assume-role-with-web-identity

以下代码示例演示了如何使用 `assume-role-with-web-identity`。

### AWS CLI

获取通过 Web 身份 ( OAuth 2.0 ) 进行身份验证的角色的短期凭证

以下 `assume-role-with-web-identity` 命令将检索 IAM 角色 `app1` 的一组短期凭证。使用由指定 Web 身份提供者提供的 Web 身份令牌对请求进行身份验证。两个附加策略应用于会话，以进一步限制用户可以执行的操作。返回的凭证在生成一个小时后过期。

```

aws sts assume-role-with-web-identity \
  --duration-seconds 3600 \
  --role-session-name "app1" \
  --provider-id "www.amazon.com" \
  --policy-arns "arn:aws:iam::123456789012:policy/
q=webidentitydemopolicy1","arn:aws:iam::123456789012:policy/webidentitydemopolicy2"
\
  --role-arn arn:aws:iam::123456789012:role/FederatedWebIdentityRole \
  --web-identity-token "Atza
%7CIQEBLjAsAhRFiXuWpUXuRvQ9PZL3GMFcYevydwIUFAHZwXZXXXXXXXXXJnrulxKDHwy87oGKPznh0D6bEQZTSCzyoC
CrKqjG7nPBjNIL016GGvuS5gSvPRUxWES3VYfm1w17WTI7jn-Pcb6M-

```

```
buCgHhF0zTQxod27L9Cqn0Lio7N3gZAGpsp6n1-  
AJB0CJckcyXe2c6uD0sr0JeZlKUm2eTDVMf8IehDVI0r1Q0nTV6KzzAI30Y87Vd_cVMQ"
```

输出：

```
{  
  "SubjectFromWebIdentityToken": "amzn1.account.AF6RH07KZU5XRVQJGXX6HB56KR2A"  
  "Audience": "client.5498841531868486423.1548@apps.example.com",  
  "AssumedRoleUser": {  
    "Arn": "arn:aws:sts::123456789012:assumed-role/FederatedWebIdentityRole/  
app1",  
    "AssumedRoleId": "AROACLKWSQRAOEXAMPLE:app1"  
  }  
  "Credentials": {  
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",  
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxrFiCYzEXAMPLEKEY",  
    "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4B1CFFxWNE1OPTgk5TthT  
+FvwqnKwRc0IfrrRh3c/LTo6UDdyJw00vEVPvLXCrrrUtdnniCEXAMPLE/  
IvU1dYug2RVAJBanLiHb4IgrmpRV3zrkuWJ0gQs8IZZaIv2BXIa2R40lgkBN9bkUDNCJiBeb/  
AX1zBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wn0E49mRl/+0tkIKG07fAE",  
    "Expiration": "2020-05-19T18:06:10+00:00"  
  },  
  "Provider": "www.amazon.com"  
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[请求临时安全凭证](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssumeRoleWithWebIdentity](#)中的。

## assume-role

以下代码示例演示了如何使用 `assume-role`。

### AWS CLI

#### 要代入角色

以下 `assume-role` 命令将检索 IAM 角色 `s3-access-example` 的一组短期凭证。

```
aws sts assume-role \  
  --role-arn arn:aws:iam::123456789012:role/xaccounts3access \  
  --role-session-name s3-access-example
```

输出：

```
{
  "AssumedRoleUser": {
    "AssumedRoleId": "ARO3XFRBF535PLBIFPI4:s3-access-example",
    "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-access-example"
  },
  "Credentials": {
    "SecretAccessKey": "9drTJvcXLB89EXAMPLEL8923FB892xMFI",
    "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/qwjzP2iEXAMPLEbw/
m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtY1FVgAUj
+7Indz3LU0aTWk1WKIjHmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8BRi2
IcrxSpnWEXAMPLEXSDFTAQAM6D19zR0tXoybnlrZIwML1Mi1Kcgo50ytwU=",
    "Expiration": "2016-03-15T00:05:07Z",
    "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
  }
}
```

该命令的输出包含访问密钥、私有密钥和会话令牌，您可以使用它们对 AWS 进行身份验证。

要使用 AWS CLI，您可以设置与角色关联的命名配置文件。当您使用配置文件时，AWS CLI 将调用 `assume-role` 并为您管理证书。有关更多信息，请参阅 [CLI 用户指南中的在 AWS CLI 中使用 IAM 角色](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 AssumeRole](#) 中的。

## decode-authorization-message

以下代码示例演示了如何使用 `decode-authorization-message`。

### AWS CLI

解码为响应请求而返回的编码授权消息

以下 `decode-authorization-message` 示例从为响应 Amazon Web Services 请求而返回的编码消息中解码有关请求授权状态的其他信息。

```
aws sts decode-authorization-message \
```

```
--encoded-message EXAMPLEWodyRNrt1QARDip-
eTA6i6Dr1UhHhPQrLWB_lAb15pAKx19mPDLexYcGBreyIKQC1BGBIpBKr3dFDkwqe07e2NMk5j_hmzAiChJN-8oy3Ewi
Ojau7BMj0TWw0tHPHv_Zaz87yENdipr745EjQwRd5LaoL3vN8_5ZfA9UiBMKDGvH1gjqZJFUiQoubv78V1RbHNYnK44E
p0u3FZjwYStfvTb3GHs3-6rLribG09jZ0ktkfE6vqx1FzLyeDr4P2ihC1wty9tArCvvGzIAUNmARQJ2VWVPxioqgoqCz
JWP5pwe_mAyqh0NLw-r1S56YC_90onj9A80sNrH1I-
tIiNd7tgNTYzDuPQYD2FMDBnp82V9eVmYgtPp5NIeSpuf3f0HanFuBZgENxZQZ2dlH3xJGMTtYayzZrRXjiq_SfX9zeB
FaOPiB8LmmKVBLpIB0iFhU9sEHPqKHVPi6jdxXqKaZaFGvYVmVOiuQdNQKuyk0p067POFrZECLjj0tNPBOZCcuEKEXAM
```

输出：

```
{
  "DecodedMessage": "{\"allowed\":false,\"explicitDeny\":true,\"matchedStatements\
\":{\\"items\":[{\\"statementId\":"\"VisualEditor0\"\",\"effect\":"\"DENY\"\",\"principals\
\":{\\"items\":[{\\"value\":"\"AROA123456789EXAMPLE\"}]}],\"principalGroups\
\":{\\"items\":[]}],\"actions\":{\\"items\":[{\\"value\":"\"ec2:RunInstances\
\"}]}],\"resources\":{\\"items\":[{\\"value\":"\"*\"}]}],\"conditions\":{\\"items\
\":[[]]}],\"failures\":{\\"items\":[]}],\"context\":{\\"principal\":{\\"id\":"
\"AROA123456789EXAMPLE:Ana\"\",\"arn\":"\"arn:aws:sts::111122223333:assumed-role/
Developer/Ana\"},\"action\":"\"RunInstances\"\",\"resource\":"\"arn:aws:ec2:us-
east-1:111122223333:instance/*\"\",\"conditions\":{\\"items\":[{\\"key\":"
\"ec2:MetadataHttpPutResponseHopLimit\"\",\"values\":{\\"items\":[{\\"value\":"
\"2\"}]}],{\\"key\":"\"ec2:InstanceMarketType\"\",\"values\":{\\"items\":[{\\"value\
\":\\"on-demand\"}]}],{\\"key\":"\"aws:Resource\"\",\"values\":{\\"items\":[{\\"value\
\":\\"instance/*\"}]}],{\\"key\":"\"aws:Account\"\",\"values\":{\\"items\":[{\\"value\
\":\\"111122223333\"}]}],{\\"key\":"\"ec2:AvailabilityZone\"\",\"values\":{\\"items\
\":[{\\"value\":"\"us-east-1f\"}]}],{\\"key\":"\"ec2:ebsoptimized\"\",\"values\":{\\"items\
\":[{\\"value\":"\"false\"}]}],{\\"key\":"\"ec2:IsLaunchTemplateResource\"\",\"values\
\":{\\"items\":[{\\"value\":"\"false\"}]}],{\\"key\":"\"ec2:InstanceType\"\",\"values\
\":{\\"items\":[{\\"value\":"\"t2.micro\"}]}],{\\"key\":"\"ec2:RootDeviceType\"\",\"values\
\":{\\"items\":[{\\"value\":"\"efs\"}]}],{\\"key\":"\"aws:Region\"\",\"values\":{\\"items\
\":[{\\"value\":"\"us-east-1\"}]}],{\\"key\":"\"ec2:MetadataHttpEndpoint\"\",\"values\
\":{\\"items\":[{\\"value\":"\"enabled\"}]}],{\\"key\":"\"aws:Service\"\",\"values\":
{\\"items\":[{\\"value\":"\"ec2\"}]}],{\\"key\":"\"ec2:InstanceID\"\",\"values\":{\\"items\
\":[{\\"value\":"\"*\"}]}],{\\"key\":"\"ec2:MetadataHttpTokens\"\",\"values\":{\\"items\
\":[{\\"value\":"\"required\"}]}],{\\"key\":"\"aws:Type\"\",\"values\":{\\"items\":
[{\\"value\":"\"instance\"}]}],{\\"key\":"\"ec2:Tenancy\"\",\"values\":{\\"items\":
[{\\"value\":"\"default\"}]}],{\\"key\":"\"ec2:Region\"\",\"values\":{\\"items\":[{\\"value\
\":\\"us-east-1\"}]}],{\\"key\":"\"aws:ARN\"\",\"values\":{\\"items\":[{\\"value\":"
\"arn:aws:ec2:us-east-1:111122223333:instance/*\"}]}]}]}]"
}
```

有关更多信息，请参阅《AWS IAM 用户指南》中的[策略评估逻辑](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DecodeAuthorizationMessage](#) 中的。

## get-caller-identity

以下代码示例演示了如何使用 `get-caller-identity`。

### AWS CLI

获取有关当前 IAM 身份的详细信息

以下 `get-caller-identity` 命令显示有关用于对请求进行身份验证的 IAM 身份的信息。来电者是 IAM 用户。

```
aws sts get-caller-identity
```

输出：

```
{
  "UserId": "AIDASAMPLEUSERID",
  "Account": "123456789012",
  "Arn": "arn:aws:iam::123456789012:user/DevAdmin"
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetCallerIdentity](#) 中的。

## get-federation-token

以下代码示例演示了如何使用 `get-federation-token`。

### AWS CLI

使用 IAM 用户访问密钥凭证返回一组临时的安全凭证

以下 `get-federation-token` 示例返回用户的一组临时安全凭证（由访问密钥 ID、秘密访问密钥和安全令牌组成）。您必须使用 IAM 用户的长期安全凭证调用 `GetFederationToken` 操作。

```
aws sts get-federation-token \
  --name Bob \
  --policy file://myfile.json \
  --policy-arns arn=arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess \
  --duration-seconds 900
```

`myfile.json` 的内容：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "elasticloadbalancing:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:Describe*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "autoscaling:Describe*",
      "Resource": "*"
    }
  ]
}
```

输出：

```
{
  "Credentials": {
    "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY",
    "SessionToken": "EXAMPLEpZ2luX2VjEGoaCXVzLXdlc3QtMiJIMEYCIQC/  
W9pL5ArQyDD5JwFL3/h5+WGopQ24GEXweNctwhi9sgIhAMkg  
+MZE35iWM8s4r5Lr25f9rSTVPFH98G42QQuNWMTfKq0DCOP/////////  
wEQAxoMNDUy0TI1MTcwNTA3Igxuy3A0puuoLsk3MJwqgQPg8Q0d9HuoClUxq26wnc/nm  
+eZLjHDyGf2KUAHK2DuaS/nrGSEXAMPLE",
    "Expiration": "2023-12-20T02:06:07+00:00"
  },
}
```

```

    "FederatedUser": {
      "FederatedUserId": "111122223333:Bob",
      "Arn": "arn:aws:sts::111122223333:federated-user/Bob"
    },
    "PackedPolicySize": 36
  }

```

有关更多信息，请参阅《AWS IAM 用户指南》中的[请求临时安全凭证](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetFederationToken](#)中的。

## get-session-token

以下代码示例演示了如何使用 `get-session-token`。

### AWS CLI

要获取 IAM 身份的一组短期凭证

以下 `get-session-token` 命令将检索进行调用的 IAM 身份的一组短期凭证。生成的凭证可用于策略要求多重身份验证 (MFA) 的请求。凭证在生成 15 分钟后过期。

```

aws sts get-session-token \
  --duration-seconds 900 \
  --serial-number "YourMFADeviceSerialNumber" \
  --token-code 123456

```

输出：

```

{
  "Credentials": {
    "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
    "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvqwqKwRc0IfrRh3c/LTo6UDdyJw00vEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgrmpRV3zrkuWJ0gQs8IZZaIv2BXIa2R40lgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wn0E49mRl/+0tkIKG07fAE",
    "Expiration": "2020-05-19T18:06:10+00:00"
  }
}

```

有关更多信息，请参阅《AWS IAM 用户指南》中的[请求临时安全凭证](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSessionToken](#)中的。

## AWS Support 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Support。 AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### add-attachments-to-set

以下代码示例演示了如何使用 add-attachments-to-set。

AWS CLI

向集合中添加附件

以下add-attachments-to-set示例向一组图片添加了一张图片，然后您可以为 AWS 账户中的支持案例指定该图片。

```
aws support add-attachments-to-set \  
  --attachment-set-id "as-2f5a6faa2a4a1e600-mu-nk5xQ1Br70-  
G1cUos5LZkd38K0AHZa9BMDVzNEXAMPLE" \  
  --attachments fileName=troubleshoot-screenshot.png,data=base64-encoded-string
```

输出：

```
{  
  "attachmentSetId": "as-2f5a6faa2a4a1e600-mu-nk5xQ1Br70-  
G1cUos5LZkd38K0AHZa9BMDVzNEXAMPLE",
```

```
"expiryTime": "2020-05-14T17:04:40.790+0000"
}
```

有关更多信息，请参阅《AWS Support 用户指南》中的[案例管理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddAttachmentsToSet](#)中的。

## add-communication-to-case

以下代码示例演示了如何使用 add-communication-to-case。

### AWS CLI

为案例添加沟通

以下add-communication-to-case示例将通信添加到您 AWS 账户中的支持案例中。

```
aws support add-communication-to-case \
  --case-id "case-12345678910-2013-c4c1d2bf33c5cf47" \
  --communication-body "I'm attaching a set of images to this case." \
  --cc-email-addresses "myemail@example.com" \
  --attachment-set-id "as-2f5a6faa2a4a1e600-mu-nk5xQ1Br70-
G1cUos5LZkd38K0AHZa9BMDVzNEXAMPLE"
```

输出：

```
{
  "result": true
}
```

有关更多信息，请参阅《AWS Support 用户指南》中的[案例管理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddCommunicationToCase](#)中的。

## create-case

以下代码示例演示了如何使用 create-case。

### AWS CLI

创建案例

以下 `create-case` 示例为您的 AWS 账户创建了一个支持案例。

```
aws support create-case \  
  --category-code "using-aws" \  
  --cc-email-addresses "myemail@example.com" \  
  --communication-body "I want to learn more about an AWS service." \  
  --issue-type "technical" \  
  --language "en" \  
  --service-code "general-info" \  
  --severity-code "low" \  
  --subject "Question about my account"
```

输出：

```
{  
  "caseId": "case-12345678910-2013-c4c1d2bf33c5cf47"  
}
```

有关更多信息，请参阅《AWS Support 用户指南》中的[案例管理](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateCase](#) 中的。

## describe-attachment

以下代码示例演示了如何使用 `describe-attachment`。

### AWS CLI

描述附件

以下 `describe-attachment` 示例返回有关带指定 ID 的附件的信息。

```
aws support describe-attachment \  
  --attachment-id "attachment-KBnjRNrePd9D6Jx0-Mm00xZuDEaL2JAj_0-  
gJv9qqDooTipsz3V1Nb19rCfkZneeQeDPgp8X1iVJyHH7UuhZDdNeqGoduZsPrAhyMakq1c60-  
iJjL5HqyYGiT1FG8EXAMPLE"
```

输出：

```
{
```

```
"attachment": {
  "fileName": "troubleshoot-screenshot.png",
  "data": "base64-blob"
}
```

有关更多信息，请参阅《AWS Support 用户指南》中的[案例管理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAttachment](#)中的。

## describe-cases

以下代码示例演示了如何使用 describe-cases。

### AWS CLI

#### 描述案例

以下describe-cases示例返回有关您 AWS 账户中指定支持案例的信息。

```
aws support describe-cases \
  --display-id "1234567890" \
  --after-time "2020-03-23T21:31:47.774Z" \
  --include-resolved-cases \
  --language "en" \
  --no-include-communications \
  --max-item 1
```

#### 输出：

```
{
  "cases": [
    {
      "status": "resolved",
      "ccEmailAddresses": [],
      "timeCreated": "2020-03-23T21:31:47.774Z",
      "caseId": "case-12345678910-2013-c4c1d2bf33c5cf47",
      "severityCode": "low",
      "language": "en",
      "categoryCode": "using-aws",
      "serviceCode": "general-info",
      "submittedBy": "myemail@example.com",
```

```

        "displayId": "1234567890",
        "subject": "Question about my account"
      }
    ]
  }

```

有关更多信息，请参阅《AWS Support 用户指南》中的[案例管理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCases](#)中的。

## describe-communications

以下代码示例演示了如何使用 describe-communications。

### AWS CLI

描述案例的最新通信

以下describe-communications示例返回您 AWS 账户中有关指定支持案例的最新通信。

```

aws support describe-communications \
  --case-id "case-12345678910-2013-c4c1d2bf33c5cf47" \
  --after-time "2020-03-23T21:31:47.774Z" \
  --max-item 1

```

输出：

```

{
  "communications": [
    {
      "body": "I want to learn more about an AWS service.",
      "attachmentSet": [],
      "caseId": "case-12345678910-2013-c4c1d2bf33c5cf47",
      "timeCreated": "2020-05-12T23:12:35.000Z",
      "submittedBy": "Amazon Web Services"
    }
  ],
  "NextToken": "eyJ1ZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQEXAMPLE=="
}

```

有关更多信息，请参阅《AWS Support 用户指南》中的[案例管理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeCommunications](#)中的。

## describe-services

以下代码示例演示了如何使用 describe-services。

### AWS CLI

列出 AWS 服务和类别

以下 describe-services 示例列出了用于请求一般信息的可用服务类别。

```
aws support describe-services \
  --service-code-list "general-info"
```

输出：

```
{
  "services": [
    {
      "code": "general-info",
      "name": "General Info and Getting Started",
      "categories": [
        {
          "code": "charges",
          "name": "How Will I Be Charged?"
        },
        {
          "code": "gdpr-queries",
          "name": "Data Privacy Query"
        },
        {
          "code": "reserved-instances",
          "name": "Reserved Instances"
        },
        {
          "code": "resource",
          "name": "Where is my Resource?"
        },
        {
          "code": "using-aws",
          "name": "Using AWS & Services"
        }
      ]
    }
  ]
}
```

```
    },
    {
      "code": "free-tier",
      "name": "Free Tier"
    },
    {
      "code": "security-and-compliance",
      "name": "Security & Compliance"
    },
    {
      "code": "account-structure",
      "name": "Account Structure"
    }
  ]
}
]
```

有关更多信息，请参阅《AWS Support 用户指南》中的[案例管理](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeServices](#) 中的。

## describe-severity-levels

以下代码示例演示了如何使用 `describe-severity-levels`。

### AWS CLI

列出可用的严重性级别

以下 `describe-severity-levels` 示例列出了支持案例的可用严重性级别。

```
aws support describe-severity-levels
```

输出：

```
{
  "severityLevels": [
    {
      "code": "low",
      "name": "Low"
    },
    {
```

```
        "code": "normal",
        "name": "Normal"
    },
    {
        "code": "high",
        "name": "High"
    },
    {
        "code": "urgent",
        "name": "Urgent"
    },
    {
        "code": "critical",
        "name": "Critical"
    }
]
}
```

有关更多信息，请参阅《AWS Support 用户指南》中的[选择严重性](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeSeverityLevels](#) 中的。

## describe-trusted-advisor-check-refresh-statuses

以下代码示例演示了如何使用 `describe-trusted-advisor-check-refresh-statuses`。

### AWS CLI

列出 Truste AWS d Advisor 检查的刷新状态

以下 `describe-trusted-advisor-check-refresh-statuses` 示例列出了两个 Trusted Advisor 检查的刷新状态：Amazon S3 存储桶权限和 IAM 使用。

```
aws support describe-trusted-advisor-check-refresh-statuses \
  --check-id "Pfx0RwqBli" "zXCkfM1nI3"
```

输出：

```
{
  "statuses": [
    {
      "checkId": "Pfx0RwqBli",
```



```
    "status": "none",
    "millisUntilNextRefreshable": 0
  },
  {
    "checkId": "zXCkfM1nI3",
    "status": "none",
    "millisUntilNextRefreshable": 0
  }
]
```

有关更多信息，请参阅《AWS 支持用户指南》中的 [T AWS trusted Advisor](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeTrustedAdvisorCheckRefreshStatuses](#) 中的。

## describe-trusted-advisor-check-result

以下代码示例演示了如何使用 `describe-trusted-advisor-check-result`。

### AWS CLI

列出 Tru AWS sted Advisor 检查的结果

以下 `describe-trusted-advisor-check-result` 示例列出了 IAM 使用检查的结果。

```
aws support describe-trusted-advisor-check-result \
  --check-id "zXCkfM1nI3"
```

输出：

```
{
  "result": {
    "checkId": "zXCkfM1nI3",
    "timestamp": "2020-05-13T21:38:05Z",
    "status": "ok",
    "resourcesSummary": {
      "resourcesProcessed": 1,
      "resourcesFlagged": 0,
      "resourcesIgnored": 0,
      "resourcesSuppressed": 0
    }
  },
```

```

    "categorySpecificSummary": {
      "costOptimizing": {
        "estimatedMonthlySavings": 0.0,
        "estimatedPercentMonthlySavings": 0.0
      }
    },
    "flaggedResources": [
      {
        "status": "ok",
        "resourceId": "47DEQpj8HBSa-_TImW-5JCeuQeRkm5NMpJWZEXAMPLE",
        "isSuppressed": false
      }
    ]
  }
}

```

有关更多信息，请参阅《AWS 支持用户指南》中的 [T AWS rusted Advisor](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeTrustedAdvisorCheckResult](#) 中的。

## describe-trusted-advisor-check-summaries

以下代码示例演示了如何使用 `describe-trusted-advisor-check-summaries`。

### AWS CLI

列出 Tru AWS sted Advisor 支票摘要

以下 `describe-trusted-advisor-check-summaries` 示例列出了两项 Trusted Advisor 检查的结果：Amazon S3 存储桶权限和 IAM 使用。

```

aws support describe-trusted-advisor-check-summaries \
  --check-ids "Pfx0RwqBli" "zXCkFM1nI3"

```

输出：

```

{
  "summaries": [
    {
      "checkId": "Pfx0RwqBli",
      "timestamp": "2020-05-13T21:38:12Z",
      "status": "ok",

```

```

    "hasFlaggedResources": true,
    "resourcesSummary": {
      "resourcesProcessed": 44,
      "resourcesFlagged": 0,
      "resourcesIgnored": 0,
      "resourcesSuppressed": 0
    },
    "categorySpecificSummary": {
      "costOptimizing": {
        "estimatedMonthlySavings": 0.0,
        "estimatedPercentMonthlySavings": 0.0
      }
    }
  },
  {
    "checkId": "zXCkfM1nI3",
    "timestamp": "2020-05-13T21:38:05Z",
    "status": "ok",
    "hasFlaggedResources": true,
    "resourcesSummary": {
      "resourcesProcessed": 1,
      "resourcesFlagged": 0,
      "resourcesIgnored": 0,
      "resourcesSuppressed": 0
    },
    "categorySpecificSummary": {
      "costOptimizing": {
        "estimatedMonthlySavings": 0.0,
        "estimatedPercentMonthlySavings": 0.0
      }
    }
  }
]
}

```

有关更多信息，请参阅《AWS 支持用户指南》中的 [T AWS rusted Advisor](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeTrustedAdvisorCheckSummaries](#) 中的。

## describe-trusted-advisor-checks

以下代码示例演示了如何使用 describe-trusted-advisor-checks。

## AWS CLI

### 列出可用的 T AWS trusted Advisor 支票

以下describe-trusted-advisor-checks示例列出了您 AWS 账户中可用的 Trusted Advisor 支票。这些信息包括支票名称、ID、描述、类别和元数据。请注意，为了便于阅读，输出被缩短了。

```
aws support describe-trusted-advisor-checks \
  --language "en"
```

输出：

```
{
  "checks": [
    {
      "id": "zXCkfM1nI3",
      "name": "IAM Use",
      "description": "Checks for your use of AWS Identity and Access Management (IAM). You can use IAM to create users, groups, and roles in AWS, and you can use permissions to control access to AWS resources. \n<br>\n<br>\n<b>Alert Criteria</b><br>\nYellow: No IAM users have been created for this account.\n<br>\n<br>\n<b>Recommended Action</b><br>\nCreate one or more IAM users and groups in your account. You can then create additional users whose permissions are limited to perform specific tasks in your AWS environment. For more information, see <a href=\"https://docs.aws.amazon.com/IAM/latest/UserGuide/IAMGettingStarted.html\" target=\"_blank\">Getting Started</a>. \n<br><br>\n<b>Additional Resources</b><br>\n<a href=\"https://docs.aws.amazon.com/IAM/latest/UserGuide/IAM_Introduction.html\" target=\"_blank\">What Is IAM?</a>",
      "category": "security",
      "metadata": []
    }
  ]
}
```

有关更多信息，请参阅《AWS 支持用户指南》中的 [T AWS trusted Advisor](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTrustedAdvisorChecks](#)中的。

### refresh-trusted-advisor-check

以下代码示例演示了如何使用 refresh-trusted-advisor-check。

## AWS CLI

### 刷新 Tru AWS sted Advisor 支票

以下`refresh-trusted-advisor-check`示例刷新了您 AWS 账户中的 Amazon S3 存储桶权限 Trusted Advisor 支票。

```
aws support refresh-trusted-advisor-check \  
  --check-id "Pfx0RwqBli"
```

输出：

```
{  
  "status": {  
    "checkId": "Pfx0RwqBli",  
    "status": "enqueued",  
    "millisUntilNextRefreshable": 3599992  
  }  
}
```

有关更多信息，请参阅《AWS 支持用户指南》中的 [T AWS rusted Advisor](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RefreshTrustedAdvisorCheck](#)中的。

## resolve-case

以下代码示例演示了如何使用 `resolve-case`。

### AWS CLI

#### 处理支持案例

以下`resolve-case`示例解决了您 AWS 账户中的一个支持案例。

```
aws support resolve-case \  
  --case-id "case-12345678910-2013-c4c1d2bf33c5cf47"
```

输出：

```
{  
  "finalCaseStatus": "resolved",
```

```
"initialCaseStatus": "work-in-progress"  
}
```

有关更多信息，请参阅《AWS Support 用户指南》中的[案例管理](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ResolveCase](#)中的。

## 使用 Amazon SWF 示例 AWS CLI

以下代码示例向您展示如何在 Amazon SWF 中使用来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### count-closed-workflow-executions

以下代码示例演示了如何使用 count-closed-workflow-executions。

AWS CLI

计算已关闭的工作流程执行次数

您可以使用检索swf count-closed-workflow-executions给定域的已关闭工作流程执行次数。您可以指定过滤器来计算特定的执行类别。

--domain和 o --close-time-filter --start-time-filter 参数是必需的。所有其他参数都是可选的。

```
aws swf count-closed-workflow-executions \
```

```
--domain DataFrobtzz \  
--close-time-filter "{ \"latestDate\" : 1377129600, \"oldestDate\" :  
1370044800 }"
```

输出：

```
{  
  "count": 2,  
  "truncated": false  
}
```

如果“截断”为true，则“计数”表示 Amazon SWF 可以返回的最大数量。任何进一步的结果都将被截断。

要减少返回的结果数量，您可以：

修改--close-time-filter或--start-time-filter值以缩小搜索的时间范围。每个参数都是互斥的：你只能在请求中指定其中一个。使用--close-status-filter--execution-filter、--tag-filter或--type-filter参数进一步筛选结果。但是，这些论点也是相互排斥的。

另[CountClosedWorkflowExecutions](#)请参阅 Amazon 简单工作流程服务 API 参考

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CountClosedWorkflowExecutions](#)中的。

## count-open-workflow-executions

以下代码示例演示了如何使用 count-open-workflow-executions。

### AWS CLI

计算未完成的工作流程执行次数

您可以使用检索swf count-open-workflow-executions给定域的已打开工作流程执行次数。您可以指定过滤器来计算特定的执行类别。

--domain和--start-time-filter参数是必需的。所有其他参数都是可选的。

```
aws swf count-open-workflow-executions \  
--domain DataFrobtzz \  
--start-time-filter "2013-10-01T00:00:00.000Z"
```

```
--start-time-filter "{ \"latestDate\" : 1377129600, \"oldestDate\" :  
1370044800 }"
```

输出：

```
{  
  "count": 4,  
  "truncated": false  
}
```

如果“截断”为true，则“计数”表示 Amazon SWF 可以返回的最大数量。任何进一步的结果都将被截断。

要减少返回的结果数量，您可以：

修改--start-time-filter值以缩小搜索的时间范围。使用--close-status-filter--execution-filter、--tag-filter或--type-filter参数进一步筛选结果。其中每一个都是互斥的：你只能在请求中指定其中一个。

有关更多信息，请参阅 CountOpenWorkflowExecutions 《亚马逊简单工作流程服务 API 参考》

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CountOpenWorkflowExecutions](#)中的。

## deprecate-domain

以下代码示例演示了如何使用 deprecate-domain。

### AWS CLI

#### 弃用域名

要弃用域（您仍可以看到它，但不能在它上面创建新工作流程执行或注册类型），请使用 swf deprecate-domain。它只有一个必需参数 --name，此参数用于指定要弃用的域的名称。

```
aws swf deprecate-domain \  
  --name MyNeatNewDomain ""
```

与 register-domain 一样，不会返回任何输出。但是，如果您list-domains使用查看已注册的域名，则会看到该域名已被弃用，并且不再出现在返回的数据中。

```
aws swf list-domains \  
  --registration-status REGISTERED
```



```
{
  "domainInfos": [
    {
      "status": "REGISTERED",
      "name": "DataFrobotz"
    },
    {
      "status": "REGISTERED",
      "name": "erontest"
    }
  ]
}
```

如果您--registration-status DEPRECATED与一起使用list-domains，则会看到已弃用的域名。

```
aws swf list-domains \
  --registration-status DEPRECATED
{
  "domainInfos": [
    {
      "status": "DEPRECATED",
      "name": "MyNeatNewDomain"
    }
  ]
}
```

您仍然可以使用describe-domain来获取有关已弃用域的信息。

```
aws swf describe-domain \
  --name MyNeatNewDomain
{
  "domainInfo": {
    "status": "DEPRECATED",
    "name": "MyNeatNewDomain"
  },
  "configuration": {
    "workflowExecutionRetentionPeriodInDays": "0"
  }
}
```

另[DeprecateDomain](#)请参阅 Amazon 简单工作流程服务 API 参考

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeprecateDomain](#)中的。

## describe-domain

以下代码示例演示了如何使用 describe-domain。

### AWS CLI

获取有关域名的信息

要获取有关特定域的详细信息，请使用swf describe-domain命令。有一个必需参数：--name，此参数用于指定您要获取其信息的域的名称。

```
aws swf describe-domain \  
  --name DataFrobotz  
  {  
    "domainInfo": {  
      "status": "REGISTERED",  
      "name": "DataFrobotz"  
    },  
    "configuration": {  
      "workflowExecutionRetentionPeriodInDays": "1"  
    }  
  }
```

您还可以使用获取describe-domain有关已弃用域的信息。

```
aws swf describe-domain \  
  --name MyNeatNewDomain  
  {  
    "domainInfo": {  
      "status": "DEPRECATED",  
      "name": "MyNeatNewDomain"  
    },  
    "configuration": {  
      "workflowExecutionRetentionPeriodInDays": "0"  
    }  
  }
```

另[DescribeDomain](#)请参阅 Amazon 简单工作流程服务 API 参考

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDomain](#)中的。

## list-activity-types

以下代码示例演示了如何使用 `list-activity-types`。

### AWS CLI

#### 列出活动类型

要获取网域的活动类型列表，请使用 `swf list-activity-types`。 `--domain` 和 `--registration-status` 参数是必需的。

```
aws swf list-activity-types \
  --domain DataFrobtzz \
  --registration-status REGISTERED
```

输出：

```
{
  "typeInfos": [
    {
      "status": "REGISTERED",
      "creationDate": 1371454150.451,
      "activityType": {
        "version": "1",
        "name": "confirm-user-email"
      },
      "description": "subscribe confirm-user-email activity"
    },
    {
      "status": "REGISTERED",
      "creationDate": 1371454150.709,
      "activityType": {
        "version": "1",
        "name": "confirm-user-phone"
      },
      "description": "subscribe confirm-user-phone activity"
    },
    {
      "status": "REGISTERED",
      "creationDate": 1371454149.871,
      "activityType": {
        "version": "1",
        "name": "get-subscription-info"
      }
    }
  ]
}
```

```

    },
    "description": "subscribe get-subscription-info activity"
  },
  {
    "status": "REGISTERED",
    "creationDate": 1371454150.909,
    "activityType": {
      "version": "1",
      "name": "send-subscription-success"
    },
    "description": "subscribe send-subscription-success activity"
  },
  {
    "status": "REGISTERED",
    "creationDate": 1371454150.085,
    "activityType": {
      "version": "1",
      "name": "subscribe-user-sns"
    },
    "description": "subscribe subscribe-user-sns activity"
  }
]
}

```

您可以使用 `--name` 参数仅选择具有特定名称的活动类型：

```

aws swf list-activity-types \
  --domain DataFrobtzz \
  --registration-status REGISTERED \
  --name "send-subscription-success"

```

输出：

```

{
  "typeInfos": [
    {
      "status": "REGISTERED",
      "creationDate": 1371454150.909,
      "activityType": {
        "version": "1",
        "name": "send-subscription-success"
      },
      "description": "subscribe send-subscription-success activity"
    }
  ]
}

```

```

    }
  ]
}

```

要在页面中检索结果，可以设置 `--maximum-page-size` 参数。如果返回的结果超过一页结果所能容纳的范围，则结果集中将返回 `nextPageToken` ：

```

aws swf list-activity-types \
  --domain DataFrobtzz \
  --registration-status REGISTERED \
  --maximum-page-size 2

```

输出：

```

{
  "nextPageToken": "AAAAKgAAAAEAAAAAAAAA1Gp1BelJq
+PmHvAnDxJYbup8+0R4LVtbXLD17QNY7C30pHo9Ssz06D/GuFz10yC73umBQ1t0PJ/gC/
aYpzDMqUIWIA1T9W0s2DryyZX40C/6Lhk9/
o5kdsuWMSBkHhgaZjgwp3WJINIFJFdaSMxY2vYAX7AtRtpcqJuBDDRE9RaRqDGYqIYUM1tarkiqpSY1ZVveBasBvlvyU
WGAaqehiDz7/JzLT/wNNUM0d+Nhe",
  "typeInfos": [
    {
      "status": "REGISTERED",
      "creationDate": 1371454150.451,
      "activityType": {
        "version": "1",
        "name": "confirm-user-email"
      },
      "description": "subscribe confirm-user-email activity"
    },
    {
      "status": "REGISTERED",
      "creationDate": 1371454150.709,
      "activityType": {
        "version": "1",
        "name": "confirm-user-phone"
      },
      "description": "subscribe confirm-user-phone activity"
    }
  ]
}

```

你可以将该 `nextPageToken` 值传递给 `--next-page-token` 参数 `list-activity-types` 中的下一个调用，检索下一页的结果：

```
aws swf list-activity-types \
  --domain DataFrobtzz \
  --registration-status REGISTERED \
  --maximum-page-size 2 \
  --next-page-token "AAAAKgAAAAEAAAAAAAAAA1Gp1Be1Jq
+PmHvAnDxJYbup8+0R4LVtbXLD17QNY7C30pHo9Ssz06D/GuFz10yC73umBQ1t0PJ/gC/
aYpzDMqUIWIA1T9W0s2DryyZX40C/6Lhk9/
o5kdsuWMSBkHhgaZjgwp3WJINIFJFdaSMxY2vYAX7AtRtpcqJuBDDRE9RaRqDGYqIYUM1tarki qpSY1ZVveBasBv1vyU
WGAaqehiDz7/JzLT/wWNUM0d+Nhe"
```

输出：

```
{
  "nextPageToken": "AAAAKgAAAAEAAAAAAAAAAw+7LZ4GRZPzTqBHsp2wBxWB8m1sgLCclgCuq3J+h/
m3+vOfFqtckjLwV5cc40jNAzTCuq/
Xcy1PumGwkjbajtqpZpbq0cVnFjFxFgoi0LB20lbvv0krbUISBvlpFPmSWpDSZJsxg5UxCcweteS1Fn1PNSZ/
MoinBZo80TkjMuzcsTuK0zH9wCaR8ITcALJ3SaQHU3pyIRS5hPmFA30LIc8zaAepj1aujo6hntNSCruB4"
  "typeInfos": [
    {
      "status": "REGISTERED",
      "creationDate": 1371454149.871,
      "activityType": {
        "version": "1",
        "name": "get-subscription-info"
      },
      "description": "subscribe get-subscription-info activity"
    },
    {
      "status": "REGISTERED",
      "creationDate": 1371454150.909,
      "activityType": {
        "version": "1",
        "name": "send-subscription-success"
      },
      "description": "subscribe send-subscription-success activity"
    }
  ]
}
```

如果还有更多结果要返回，则将随结果一起返回 `nextPageToken`。如果没有其他要返回的结果页，则不会在结果集中返回 `nextPageToken`。

您可以使用 `--reverse-order` 参数来颠倒返回结果的顺序。这也会影响分页结果。

```
aws swf list-activity-types \  
  --domain DataFrobtzz \  
  --registration-status REGISTERED \  
  --maximum-page-size 2 \  
  --reverse-order
```

输出：

```
{  
  "nextPageToken": "AAAAKgAAAAEAAAAAAAAAAwXcpu5ePSyQkrC  
+8WMbmSrenuZC2ZkIXQYBPB/b9xIOVkj+bMEFhGj0KmmJ4rF7iddhjf7UMYCsfGkEn7mk  
+yMCgVc1JxDWmB0EH46bhcmclmYNQihMDmUwocpr7To6/R7CLu0St1gkFayx0idJXErQW0zdNfQaIWAnF/  
cwioBbXlkz1fQzmDeU3M5oYGMPQIrUqkPq7pMEW0q01K5eDN97NzFYdZZ/r1cLDWPZhUjY",  
  "typeInfos": [  
    {  
      "status": "REGISTERED",  
      "creationDate": 1371454150.085,  
      "activityType": {  
        "version": "1",  
        "name": "subscribe-user-sns"  
      },  
      "description": "subscribe subscribe-user-sns activity"  
    },  
    {  
      "status": "REGISTERED",  
      "creationDate": 1371454150.909,  
      "activityType": {  
        "version": "1",  
        "name": "send-subscription-success"  
      },  
      "description": "subscribe send-subscription-success activity"  
    }  
  ]  
}
```

另[ListActivityTypes](#)请参阅 Amazon 简单工作流程服务 API 参考

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListActivityTypes](#)中的。

## list-domains

以下代码示例演示了如何使用 `list-domains`。

### AWS CLI

示例 1：列出您的注册域名

以下 `list-domains` 命令示例列出了您为账户注册的 REGISTERED SWF 域。

```
aws swf list-domains \  
  --registration-status REGISTERED
```

输出：

```
{  
  "domainInfos": [  
    {  
      "status": "REGISTERED",  
      "name": "DataFrobotz"  
    },  
    {  
      "status": "REGISTERED",  
      "name": "erontest"  
    }  
  ]  
}
```

有关更多信息，请参阅 [ListDomains](#) 《亚马逊简单工作流程服务 API 参考》

示例 2：列出已弃用的域名

以下 `list-domains` 命令示例列出了您为账户注册的 DEPRECATED SWF 域。已弃用的域名是指无法注册新工作流程或活动，但仍可以查询的域名。

```
aws swf list-domains \  
  --registration-status DEPRECATED
```

输出：

```
{
```



```
"domainInfos": [  
  {  
    "status": "DEPRECATED",  
    "name": "MyNeatNewDomain"  
  }  
]
```

有关更多信息，请参阅[ListDomains](#) 《亚马逊简单工作流程服务 API 参考》

### 示例 3：列出注册域名的第一页

以下list-domains命令示例列出了您使用该--maximum-page-size选项为账户注册的第一页REGISTERED SWF 域名。

```
aws swf list-domains \  
  --registration-status REGISTERED \  
  --maximum-page-size 1
```

输出：

```
{  
  "domainInfos": [  
    {  
      "status": "REGISTERED",  
      "name": "DataFrobotz"  
    }  
  ],  
  "nextPageToken": "AAAAKgAAAAEAAAAAAAAA2QJKNtidVgd49TTeNwYcpD  
+KQT2ynuEbibcQWe2QKrs1MGe63gpS0MgZGpcpoKttL40CXRFn98Xif557it  
+wSZUsvUDtImjDLvguyuyyFdIZtvIxIKE0Pm3k2r40jAGaFsG0uVbrK1jv1a7wdU7FYH301kNCP8b7PBj9SBkUyGoiAg  
}
```

有关更多信息，请参阅[ListDomains](#) 《亚马逊简单工作流程服务 API 参考》

### 示例 4：列出指定的单页注册域名

以下list-domains命令示例列出了您使用该--maximum-page-size选项为账户注册的第一页REGISTERED SWF 域名。

当你再次调用时，这次在--next-page-token参数nextPageToken中提供值时，你会得到另一页的结果。

```
aws swf list-domains \
  --registration-status REGISTERED \
  --maximum-page-size 1 \
  --next-page-token "AAAAKgAAAAEAAAAAAAAAA2QJKNtidVgd49TTeNwYcpD
+QKT2ynuEbibcQWe2QKrs1MGe63gpS0MgZGpcpoKttL40CXRFn98Xif557it
+wSZUsvUDtImjDLvguyuyyFdIZtvIxIKE0Pm3k2r40jAGaFsG0uVbrK1jv1a7wdU7FYH301kNCP8b7PBj9SBkUyGoiAg
```

输出：

```
{
  "domainInfos": [
    {
      "status": "REGISTERED",
      "name": "erontest"
    }
  ]
}
```

当没有要检索的其他结果页时，nextPageToken 不会在结果中返回。

有关更多信息，请参阅[ListDomains](#) 《亚马逊简单工作流程服务 API 参考》

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDomains](#)中的。

## list-workflow-types

以下代码示例演示了如何使用 list-workflow-types。

AWS CLI

列出工作流程类型

要获取域的工作流程类型列表，请使用swf list-workflow-types。--domain和--registration-status参数是必需的。这是一个简单的例子。

```
aws swf list-workflow-types \
  --domain DataFrobtzz \
  --registration-status REGISTERED
```

输出：

```
{
  "typeInfos": [
    {
      "status": "REGISTERED",
      "creationDate": 1371454149.598,
      "description": "DataFrobtzz subscribe workflow",
      "workflowType": {
        "version": "v3",
        "name": "subscribe"
      }
    }
  ]
}
```

与一样 `list-activity-types`，您可以使用 `--name` 参数仅选择具有特定名称的工作流程类型，并使用该 `--maximum-page-size` 参数 `--next-page-token` 来查看结果。要颠倒返回结果的顺序，请使用 `--reverse-order`。

另 [ListWorkflowTypes](#) 请参阅 Amazon 简单工作流程服务 API 参考

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListWorkflowTypes](#) 中的。

## register-domain

以下代码示例演示了如何使用 `register-domain`。

### AWS CLI

#### 注册域名

您可以使用 AWS CLI 注册新域名。使用 `swf register-domain` 命令。 < [https://aws.amazon.com/swf/faqs/#retain\\_limit](https://aws.amazon.com/swf/faqs/#retain_limit) > 有两个必需的参数 `--name`，分别是域名 `--workflow-execution-retention-period-in-days`，以及使用整数来指定在此域上保留工作流程执行数据的天数，最长为 90 天（有关更多信息，请参阅 SWF 常见问题解答）。经过指定的天数后，工作流程执行数据将不会被保留。

```
aws swf register-domain \
  --name MyNeatNewDomain \
  --workflow-execution-retention-period-in-days 0
""
```

注册域时，不会返回任何内容 (""），但您可以使用 `swf list-domains` 或 `swf describe-domain` 查看新域。

```
aws swf list-domains \
  --registration-status REGISTERED
{
  "domainInfos": [
    {
      "status": "REGISTERED",
      "name": "DataFrobotz"
    },
    {
      "status": "REGISTERED",
      "name": "MyNeatNewDomain"
    },
    {
      "status": "REGISTERED",
      "name": "erontest"
    }
  ]
}
```

使用 `swf describe-domain`：

```
aws swf describe-domain --name MyNeatNewDomain
{
  "domainInfo": {
    "status": "REGISTERED",
    "name": "MyNeatNewDomain"
  },
  "configuration": {
    "workflowExecutionRetentionPeriodInDays": "0"
  }
}
```

另[RegisterDomain](#)请参阅 Amazon 简单工作流程服务 API 参考

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterDomain](#)中的。

## register-workflow-type

以下代码示例演示了如何使用 `register-workflow-type`。

## AWS CLI

### 注册工作流程类型

要向 C AWS LI 注册工作流程类型，请使用 `swf register-workflow-type` 命令。

```
aws swf register-workflow-type \  
  --domain DataFrobtzz \  
  --name "MySimpleWorkflow" \  
  --workflow-version "v1"
```

如果成功，该命令将不产生任何输出。

出现错误时（例如，如果您尝试注册相同的工作流程类型两次，或者指定了一个不存在的域），您将收到 JSON 格式的响应。

```
{  
  "message": "WorkflowType=[name=MySimpleWorkflow, version=v1]",  
  "__type": "com.amazonaws.swf.base.model#TypeAlreadyExistsFault"  
}
```

`--domain`、`--name`和`--workflow-version`为必填项。您还可以设置工作流程描述、超时和子工作流程策略。

有关更多信息，请参阅[RegisterWorkflowType](#) 《亚马逊简单工作流程服务 API 参考》

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterWorkflowType](#)中的。

## 使用 Systems Manager 示例 AWS CLI

以下代码示例向您展示了如何使用与 Systems Manager AWS Command Line Interface 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### add-tags-to-resource

以下代码示例演示了如何使用 `add-tags-to-resource`。

#### AWS CLI

##### 示例 1：向维护时段添加标签

以下 `add-tags-to-resource` 示例向指定的维护时段添加标签。

```
aws ssm add-tags-to-resource \  
  --resource-type "MaintenanceWindow" \  
  --resource-id "mw-03eb9db428EXAMPLE" \  
  --tags "Key=Stack,Value=Production"
```

此命令不生成任何输出。

##### 示例 2：向参数添加标签

以下 `add-tags-to-resource` 示例向指定参数添加两个标签。

```
aws ssm add-tags-to-resource \  
  --resource-type "Parameter" \  
  --resource-id "My-Parameter" \  
  --tags '[{"Key":"Region","Value":"East"}, {"Key":"Environment",  
  "Value":"Production"}]'
```

此命令不生成任何输出。

##### 示例 3：向 SSM 文档添加标签

以下 `add-tags-to-resource` 示例向指定文档添加标签。

```
aws ssm add-tags-to-resource \  
  --resource-type "Document" \  
  --resource-id "My-Document" \  
  --tags "Key=Quarter,Value=Q322"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[标记 Systems Manager 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddTagsToResource](#)中的。

## associate-ops-item-related-item

以下代码示例演示了如何使用 `associate-ops-item-related-item`。

### AWS CLI

#### 关联相关项目

以下`associate-ops-item-related-item`示例将相关项目关联到 OpsItem。

```
aws ssm associate-ops-item-related-item \
  --ops-item-id "oi-649fExample" \
  --association-type "RelatesTo" \
  --resource-type "AWS::SSMIncidents::IncidentRecord" \
  --resource-uri "arn:aws:ssm-incidents::111122223333:incident-record/Example-Response-Plan/c2bde883-f7d5-343a-b13a-bf5fe9ea689f"
```

输出：

```
{
  "AssociationId": "61d7178d-a30d-4bc5-9b4e-a9e74EXAMPLE"
}
```

有关更多信息，请参阅《S AWS system s Manager 用户指南》[OpsCenter](#)中的“[处理事件管理器事件](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateOpsItemRelatedItem](#)中的。

## cancel-command

以下代码示例演示了如何使用 `cancel-command`。

### AWS CLI

示例 1：取消所有实例的命令

以下 `cancel-command` 示例尝试取消已对所有实例运行的指定命令。

```
aws ssm cancel-command \  
  --command-id "662add3d-5831-4a10-b64a-f2ff3EXAMPLE"
```

此命令不生成任何输出。

示例 2：取消特定实例的命令

以下 `cancel-command` 示例仅尝试取消指定实例的命令。

```
aws ssm cancel-command \  
  --command-id "662add3d-5831-4a10-b64a-f2ff3EXAMPLE" \  
  --instance-ids "i-02573cafcfEXAMPLE"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[标记 Systems Manager 参数](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CancelCommand](#)中的。

## cancel-maintenance-window-execution

以下代码示例演示了如何使用 `cancel-maintenance-window-execution`。

### AWS CLI

取消维护时段的执行

此 `cancel-maintenance-window-execution` 示例停止了已在进行的指定维护时段执行。

```
aws ssm cancel-maintenance-window-execution \  
  --window-execution-id j218d5b5c-mw66-tk4d-r3g9-1d4d1EXAMPLE
```

输出：

```
{  
  "WindowExecutionId": "j218d5b5c-mw66-tk4d-r3g9-1d4d1EXAMPLE"  
}
```

有关更多信息，请参阅《[系统管理器用户指南](#)》中的 [Syst AWS ems Manager 维护 Windows 教程 \(AWS CLI\)](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CancelMaintenanceWindowExecution](#)中的。

## create-activation

以下代码示例演示了如何使用 create-activation。

### AWS CLI

#### 创建托管式实例激活

以下 create-activation 示例创建托管式实例激活。

```
aws ssm create-activation \  
  --default-instance-name "HybridWebServers" \  
  --iam-role "HybridWebServersRole" \  
  --registration-limit 5
```

输出：

```
{  
  "ActivationId": "5743558d-563b-4457-8682-d16c3EXAMPLE",  
  "ActivationCode": "dRmgnYaFv567vEXAMPLE"  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[步骤 4：为混合环境创建托管式实例激活](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateActivation](#)中的。

## create-association-batch

以下代码示例演示了如何使用 create-association-batch。

### AWS CLI

#### 创建多个关联

此示例将一个配置文档与多个实例相关联。如果适用，输出将返回成功和失败操作的列表。

命令：

```
aws ssm create-association-batch --entries "Name=AWS-UpdateSSMAgent,InstanceId=i-1234567890abcdef0" "Name=AWS-UpdateSSMAgent,InstanceId=i-9876543210abcdef0"
```

输出：

```
{
  "Successful": [
    {
      "Name": "AWS-UpdateSSMAgent",
      "InstanceId": "i-1234567890abcdef0",
      "AssociationVersion": "1",
      "Date": 1550504725.007,
      "LastUpdateAssociationDate": 1550504725.007,
      "Status": {
        "Date": 1550504725.007,
        "Name": "Associated",
        "Message": "Associated with AWS-UpdateSSMAgent"
      },
      "Overview": {
        "Status": "Pending",
        "DetailedStatus": "Creating"
      },
      "DocumentVersion": "$DEFAULT",
      "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
      "Targets": [
        {
          "Key": "InstanceIds",
          "Values": [
            "i-1234567890abcdef0"
          ]
        }
      ]
    },
    {
      "Name": "AWS-UpdateSSMAgent",
      "InstanceId": "i-9876543210abcdef0",
      "AssociationVersion": "1",
      "Date": 1550504725.057,
      "LastUpdateAssociationDate": 1550504725.057,
      "Status": {
        "Date": 1550504725.057,
        "Name": "Associated",
```

```
    "Message": "Associated with AWS-UpdateSSMAgent"
  },
  "Overview": {
    "Status": "Pending",
    "DetailedStatus": "Creating"
  },
  "DocumentVersion": "$DEFAULT",
  "AssociationId": "9c9f7f20-5154-4fed-a83e-0123456789ab",
  "Targets": [
    {
      "Key": "InstanceIds",
      "Values": [
        "i-9876543210abcdef0"
      ]
    }
  ]
},
"Failed": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAssociationBatch](#)中的。

## create-association

以下代码示例演示了如何使用 create-association。

### AWS CLI

示例 1：使用实例 ID 关联文档

此示例使用实例 ID 将配置文档与实例关联起来。

```
aws ssm create-association \
  --instance-id "i-0cb2b964d3e14fd9f" \
  --name "AWS-UpdateSSMAgent"
```

输出：

```
{
  "AssociationDescription": {
    "Status": {
```

```

        "Date": 1487875500.33,
        "Message": "Associated with AWS-UpdateSSMAgent",
        "Name": "Associated"
    },
    "Name": "AWS-UpdateSSMAgent",
    "InstanceId": "i-0cb2b964d3e14fd9f",
    "Overview": {
        "Status": "Pending",
        "DetailedStatus": "Creating"
    },
    "AssociationId": "b7c3266e-a544-44db-877e-b20d3a108189",
    "DocumentVersion": "$DEFAULT",
    "LastUpdateAssociationDate": 1487875500.33,
    "Date": 1487875500.33,
    "Targets": [
        {
            "Values": [
                "i-0cb2b964d3e14fd9f"
            ],
            "Key": "InstanceIds"
        }
    ]
}

```

有关更多信息，请参阅 S AWS systems Manager API 参考[CreateAssociation](#)中的。

#### 示例 2：使用目标关联文档

此示例使用目标将配置文档与实例关联起来。

```

aws ssm create-association \
  --name "AWS-UpdateSSMAgent" \
  --targets "Key=instanceids,Values=i-0cb2b964d3e14fd9f"

```

输出：

```

{
  "AssociationDescription": {
    "Status": {
      "Date": 1487875500.33,
      "Message": "Associated with AWS-UpdateSSMAgent",
      "Name": "Associated"
    }
  }
}

```

```

    },
    "Name": "AWS-UpdateSSMAgent",
    "InstanceId": "i-0cb2b964d3e14fd9f",
    "Overview": {
      "Status": "Pending",
      "DetailedStatus": "Creating"
    },
    "AssociationId": "b7c3266e-a544-44db-877e-b20d3a108189",
    "DocumentVersion": "$DEFAULT",
    "LastUpdateAssociationDate": 1487875500.33,
    "Date": 1487875500.33,
    "Targets": [
      {
        "Values": [
          "i-0cb2b964d3e14fd9f"
        ],
        "Key": "InstanceIds"
      }
    ]
  }
}

```

有关更多信息，请参阅 S AWS systems Manager API 参考[CreateAssociation](#)中的。

### 示例 3：创建仅运行一次的关联

此示例创建一个仅在指定日期和时间运行一次的新关联。使用过去或现在的日期创建的关联（处理关联时该日期已过去）会立即运行。

```

aws ssm create-association \
  --name "AWS-UpdateSSMAgent" \
  --targets "Key=instanceids,Values=i-0cb2b964d3e14fd9f" \
  --schedule-expression "at(2020-05-14T15:55:00)" \
  --apply-only-at-cron-interval

```

输出：

```

{
  "AssociationDescription": {
    "Status": {
      "Date": 1487875500.33,
      "Message": "Associated with AWS-UpdateSSMAgent",
      "Name": "Associated"
    }
  }
}

```

```

    },
    "Name": "AWS-UpdateSSMAgent",
    "InstanceId": "i-0cb2b964d3e14fd9f",
    "Overview": {
      "Status": "Pending",
      "DetailedStatus": "Creating"
    },
    "AssociationId": "b7c3266e-a544-44db-877e-b20d3a108189",
    "DocumentVersion": "$DEFAULT",
    "LastUpdateAssociationDate": 1487875500.33,
    "Date": 1487875500.33,
    "Targets": [
      {
        "Values": [
          "i-0cb2b964d3e14fd9f"
        ],
        "Key": "InstanceIds"
      }
    ]
  }
}

```

有关更多信息，请参阅 [CreateAssociation](#) S AWS systems Manager API 参考或参考：[Systems Manager 用户指南中的 AWS Cron 和速率表达式](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateAssociation](#) 中的。

## create-document

以下代码示例演示了如何使用 create-document。

### AWS CLI

#### 创建文档

以下 create-document 示例创建一个 Systems Manager 文档。

```

aws ssm create-document \
  --content file://exampleDocument.yml \
  --name "Example" \
  --document-type "Automation" \
  --document-format YAML

```

输出：

```
{
  "DocumentDescription": {
    "Hash": "fc2410281f40779e694a8b95975d0f9f316da8a153daa94e3d9921102EXAMPLE",
    "HashType": "Sha256",
    "Name": "Example",
    "Owner": "29884EXAMPLE",
    "CreateDate": 1583256349.452,
    "Status": "Creating",
    "DocumentVersion": "1",
    "Description": "Document Example",
    "Parameters": [
      {
        "Name": "AutomationAssumeRole",
        "Type": "String",
        "Description": "(Required) The ARN of the role that allows
Automation to perform the actions on your behalf. If no role is specified, Systems
Manager Automation uses your IAM permissions to execute this document.",
        "DefaultValue": ""
      },
      {
        "Name": "InstanceId",
        "Type": "String",
        "Description": "(Required) The ID of the Amazon EC2 instance.",
        "DefaultValue": ""
      }
    ],
    "PlatformTypes": [
      "Windows",
      "Linux"
    ],
    "DocumentType": "Automation",
    "SchemaVersion": "0.3",
    "LatestVersion": "1",
    "DefaultVersion": "1",
    "DocumentFormat": "YAML",
    "Tags": []
  }
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[创建 Systems Manager 文档](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateDocument](#) 中的。

## create-maintenance-window

以下代码示例演示了如何使用 `create-maintenance-window`。

### AWS CLI

#### 示例 1：创建维护时段

以下 `create-maintenance-window` 示例创建一个新的维护时段，每五分钟执行一次，最多持续两个小时（根据需要），防止新任务在维护时段执行结束后的一小时内启动，允许未关联的目标（您尚未向维护时段注册的实例），并通过使用自定义标签表明其创建者打算在教程中进行使用。

```
aws ssm create-maintenance-window \  
  --name "My-Tutorial-Maintenance-Window" \  
  --schedule "rate(5 minutes)" \  
  --duration 2 --cutoff 1 \  
  --allow-unassociated-targets \  
  --tags "Key=Purpose,Value=Tutorial"
```

输出：

```
{  
  "WindowId": "mw-0c50858d01EXAMPLE"  
}
```

#### 示例 2：创建仅运行一次的维护时段

以下 `create-maintenance-window` 示例创建了一个仅在指定日期和时间运行一次的新维护时段。

```
aws ssm create-maintenance-window \  
  --name My-One-Time-Maintenance-Window \  
  --schedule "at(2020-05-14T15:55:00)" \  
  --duration 5 \  
  --cutoff 2 \  
  --allow-unassociated-targets \  
  --tags "Key=Environment,Value=Production"
```

输出：

```
{
```



```
"WindowId": "mw-01234567890abcdef"
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[维护时段](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateMaintenanceWindow](#)中的。

## create-ops-item

以下代码示例演示了如何使用 create-ops-item。

### AWS CLI

#### 要创建 OpsItems

以下create-ops-item示例使用中的 /aws/resources 密钥创建 OpsItem 具有亚马逊 OperationalData 索引 DynamoDB 相关资源的资源。

```
aws ssm create-ops-item \
  --title "EC2 instance disk full" \
  --description "Log clean up may have failed which caused the disk to be full" \
  --priority 2 \
  --source ec2 \
  --operational-data '{"aws/resources":{"Value":[{"arn": "\narn:aws:dynamodb:us-west-2:12345678:table/OpsItems\n"}]},"Type":"SearchableString"}' \
  --notifications Arn="arn:aws:sns:us-west-2:12345678:TestUser"
```

输出：

```
{
  "OpsItemId": "oi-1a2b3c4d5e6f"
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》OpsItems中的“[创建](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateOpsItem](#)中的。

## create-patch-baseline

以下代码示例演示了如何使用 create-patch-baseline。

## AWS CLI

### 示例 1：创建具有自动批准功能的补丁基准

以下 `create-patch-baseline` 示例创建一个 Windows Server 的补丁基准，该基准在 Microsoft 发布补丁 7 天后批准生产环境的补丁。

```
aws ssm create-patch-baseline \  
  --name "Windows-Production-Baseline-AutoApproval" \  
  --operating-system "WINDOWS" \  
  --approval-rules  
  "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Critical,Important  
{Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,UpdateRollups,CriticalUpdates]}]},Approv  
  \  
  --description "Baseline containing all updates approved for Windows Server  
  production systems"
```

输出：

```
{  
  "BaselineId": "pb-045f10b4f3EXAMPLE"  
}
```

### 示例 2：创建带有批准截止日期的补丁基准

以下 `create-patch-baseline` 示例为 Windows Server 创建补丁基准，其批准 2020 年 7 月 7 日或之前在生产环境中发布的所有补丁。

```
aws ssm create-patch-baseline \  
  --name "Windows-Production-Baseline-AutoApproval" \  
  --operating-system "WINDOWS" \  
  --approval-rules  
  "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Critical,Important  
{Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,UpdateRollups,CriticalUpdates]}]},Approv  
  \  
  --description "Baseline containing all updates approved for Windows Server  
  production systems"
```

输出：

```
{
```

```
"BaselineId": "pb-045f10b4f3EXAMPLE"  
}
```

### 示例 3：创建批准规则存储在 JSON 文件中的补丁基准

以下 `create-patch-baseline` 示例为 Amazon Linux 2017.09 创建补丁基准，其将在补丁发布 7 天后批准生产环境的补丁，指定补丁基准的批准规则，并指定补丁的自定义存储库。

```
aws ssm create-patch-baseline \  
  --cli-input-json file://my-amazon-linux-approval-rules-and-repo.json
```

`my-amazon-linux-approval-rules-and-repo.json` 的内容：

```
{  
  "Name": "Amazon-Linux-2017.09-Production-Baseline",  
  "Description": "My approval rules patch baseline for Amazon Linux 2017.09  
instances",  
  "OperatingSystem": "AMAZON_LINUX",  
  "Tags": [  
    {  
      "Key": "Environment",  
      "Value": "Production"  
    }  
  ],  
  "ApprovalRules": {  
    "PatchRules": [  
      {  
        "ApproveAfterDays": 7,  
        "EnableNonSecurity": true,  
        "PatchFilterGroup": {  
          "PatchFilters": [  
            {  
              "Key": "SEVERITY",  
              "Values": [  
                "Important",  
                "Critical"  
              ]  
            }  
          ],  
          "Key": "CLASSIFICATION",  
          "Values": [  
            "Security",
```

```

        "Bugfix"
      ]
    },
    {
      "Key": "PRODUCT",
      "Values": [
        "AmazonLinux2017.09"
      ]
    }
  ]
}
]
},
"Sources": [
  {
    "Name": "My-AL2017.09",
    "Products": [
      "AmazonLinux2017.09"
    ],
    "Configuration": "[amzn-main] \nname=amzn-main-Base
\nmirrorlist=http://repo./$awsregion./$awsdomain//$releasever/main/mirror.list //
\nmirrorlist_expire=300//\nmetadata_expire=300 \npriority=10 \nfailovermethod=priority
\nfastestmirror_enabled=0 \ngpgcheck=1 \ngpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-
KEY-amazon-ga \nenabled=1 \nretries=3 \ntimeout=5\nreport_instanceid=yes"
  }
]
}

```

#### 示例 4：创建指定已批准和已拒绝补丁的补丁基准

以下 `create-patch-baseline` 示例明确指定要批准和拒绝的补丁，作为默认批准规则的例外情况。

```

aws ssm create-patch-baseline \
  --name "Amazon-Linux-2017.09-Alpha-Baseline" \
  --description "My custom approve/reject patch baseline for Amazon Linux 2017.09
instances" \
  --operating-system "AMAZON_LINUX" \
  --approved-patches "CVE-2018-1234567,example-pkg-EE-2018*.amzn1.noarch" \
  --approved-patches-compliance-level "HIGH" \
  --approved-patches-enable-non-security \
  --tags "Key=Environment,Value=Alpha"

```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[创建自定义补丁基准](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePatchBaseline](#)中的。

## create-resource-data-sync

以下代码示例演示了如何使用 create-resource-data-sync。

### AWS CLI

#### 创建资源数据同步

此示例创建了资源数据同步。如果此命令成功，则无任何输出。

命令：

```
aws ssm create-resource-data-sync --sync-name "ssm-resource-data-sync" --s3-destination "BucketName=ssm-bucket,Prefix=inventory,SyncFormat=JsonSerDe,Region=us-east-1"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateResourceDataSync](#)中的。

## delete-activation

以下代码示例演示了如何使用 delete-activation。

### AWS CLI

#### 删除托管式实例激活

以下 delete-activation 示例删除托管式实例激活。

```
aws ssm delete-activation \  
  --activation-id "aa673477-d926-42c1-8757-1358cEXAMPLE"
```

此命令不生成任何输出。

有关更多信息，请参阅《[S AWS systems Manager 用户指南](#)》中的为混合环境设置AWS Systems Manager。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteActivation](#)中的。

## delete-association

以下代码示例演示了如何使用 delete-association。

### AWS CLI

#### 示例 1：使用关联 ID 删除关联

以下 delete-association 示例删除指定关联 ID 的关联。如果此命令成功，则无任何输出。

```
aws ssm delete-association \  
  --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[编辑和创建关联的新版本](#)。

#### 示例 2：删除关联

以下 delete-association 示例删除实例和文档之间的关联。如果此命令成功，则无任何输出。

```
aws ssm delete-association \  
  --instance-id "i-1234567890abcdef0" \  
  --name "AWS-UpdateSSMAgent"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[在 Systems Manager 中使用关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteAssociation](#)中的。

## delete-document

以下代码示例演示了如何使用 delete-document。

### AWS CLI

#### 删除文档

以下 delete-document 示例删除一个 Systems Manager 文档。

```
aws ssm delete-document \  
  --name "AWS-UpdateSSMAgent"
```

```
--name "Example"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[创建 Systems Manager 文档](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteDocument](#)中的。

## delete-inventory

以下代码示例演示了如何使用 delete-inventory。

### AWS CLI

#### 删除自定义库存类型

此示例删除了自定义清单架构。

命令：

```
aws ssm delete-inventory --type-name "Custom:RackInfo" --schema-delete-option  
"DeleteSchema"
```

输出：

```
{  
  "DeletionId": "d72ac9e8-1f60-4d40-b1c6-bf8c78c68c4d",  
  "TypeName": "Custom:RackInfo",  
  "DeletionSummary": {  
    "TotalCount": 1,  
    "RemainingCount": 1,  
    "SummaryItems": [  
      {  
        "Version": "1.0",  
        "Count": 1,  
        "RemainingCount": 1  
      }  
    ]  
  }  
}
```

#### 禁用自定义库存类型

此示例禁用了自定义清单架构。

命令:

```
aws ssm delete-inventory --type-name "Custom:RackInfo" --schema-delete-option
"DisableSchema"
```

输出:

```
{
  "DeletionId": "6961492a-8163-44ec-aa1e-923364dd0850",
  "TypeName": "Custom:RackInformation",
  "DeletionSummary": {
    "TotalCount": 0,
    "RemainingCount": 0,
    "SummaryItems": []
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteInventory](#)中的。

## delete-maintenance-window

以下代码示例演示了如何使用 delete-maintenance-window。

AWS CLI

删除维护时段

此 delete-maintenance-window 示例删除指定的维护时段。

```
aws ssm delete-maintenance-window \
  --window-id "mw-1a2b3c4d5e6f7g8h9"
```

输出:

```
{
  "WindowId": "mw-1a2b3c4d5e6f7g8h9"
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[删除维护窗口 \(AWS CLI\)](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMaintenanceWindow](#)中的。

## delete-parameter

以下代码示例演示了如何使用 delete-parameter。

### AWS CLI

#### 删除参数

以下 delete-parameter 示例将删除指定的一个参数。

```
aws ssm delete-parameter \  
  --name "MyParameter"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Parameter Store](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteParameter](#)中的。

## delete-parameters

以下代码示例演示了如何使用 delete-parameters。

### AWS CLI

#### 删除参数列表

以下delete-parameters示例删除了指定的参数。

```
aws ssm delete-parameters \  
  --names "MyFirstParameter" "MySecondParameter" "MyInvalidParameterName"
```

输出：

```
{  
  "DeletedParameters": [  
    "MyFirstParameter",  
    "MySecondParameter"  
  ],  
  "InvalidParameters": [  
    "MyInvalidParameterName"  
  ]  
}
```

```
    "MyInvalidParameterName"  
  ]  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Parameter Store](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteParameters](#)中的。

## delete-patch-baseline

以下代码示例演示了如何使用 delete-patch-baseline。

### AWS CLI

#### 删除补丁基准

以下 delete-patch-baseline 示例将删除指定的补丁基准。

```
aws ssm delete-patch-baseline \  
  --baseline-id "pb-045f10b4f382baeda"
```

输出：

```
{  
  "BaselineId": "pb-045f10b4f382baeda"  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[更新或删除补丁基准（控制台）](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePatchBaseline](#)中的。

## delete-resource-data-sync

以下代码示例演示了如何使用 delete-resource-data-sync。

### AWS CLI

#### 删除资源数据同步

此示例删除资源数据同步。如果此命令成功，则无任何输出。

命令：

```
aws ssm delete-resource-data-sync --sync-name "ssm-resource-data-sync"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteResourceDataSync](#)中的。

## deregister-managed-instance

以下代码示例演示了如何使用 deregister-managed-instance。

### AWS CLI

取消注册托管式实例

以下 deregister-managed-instance 示例取消注册指定的托管式实例。

```
aws ssm deregister-managed-instance
  --instance-id "mi-08ab247cdfEXAMPLE"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[在混合环境中取消注册托管式实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterManagedInstance](#)中的。

## deregister-patch-baseline-for-patch-group

以下代码示例演示了如何使用 deregister-patch-baseline-for-patch-group。

### AWS CLI

从补丁基准取消注册补丁组

以下 deregister-patch-baseline-for-patch-group 示例从指定的补丁基准中取消注册指定的补丁组。

```
aws ssm deregister-patch-baseline-for-patch-group \
  --patch-group "Production" \
  --baseline-id "pb-0ca44a362fEXAMPLE"
```

输出：

```
{
```

```
"PatchGroup": "Production",
"BaselineId": "pb-0ca44a362fEXAMPLE"
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[将补丁组添加到补丁基准](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeregisterPatchBaselineForPatchGroup](#) 中的。

## deregister-target-from-maintenance-window

以下代码示例演示了如何使用 `deregister-target-from-maintenance-window`。

### AWS CLI

从维护时段删除目标

以下 `deregister-target-from-maintenance-window` 示例从指定的维护时段中删除指定的目标。

```
aws ssm deregister-target-from-maintenance-window \
  --window-id "mw-ab12cd34ef56gh78" \
  --window-target-id "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
```

输出：

```
{
  "WindowId": "mw-ab12cd34ef56gh78",
  "WindowTargetId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[“更新维护窗口 \(AWS CLI\)”](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeregisterTargetFromMaintenanceWindow](#) 中的。

## deregister-task-from-maintenance-window

以下代码示例演示了如何使用 `deregister-task-from-maintenance-window`。

### AWS CLI

从维护时段删除任务

以下 `deregister-task-from-maintenance-window` 示例从指定的维护时段中删除指定的任务。

```
aws ssm deregister-task-from-maintenance-window \  
  --window-id "mw-ab12cd34ef56gh78" \  
  --window-task-id "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e6c"
```

输出：

```
{  
  "WindowTaskId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e6c",  
  "WindowId": "mw-ab12cd34ef56gh78"  
}
```

有关更多信息，请参阅 [《系统管理器用户指南》中的 Syst AWS ems Manager 维护 Windows 教程 \(AWS CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeregisterTaskFromMaintenanceWindow](#) 中的。

## describe-activations

以下代码示例演示了如何使用 `describe-activations`。

### AWS CLI

描述激活

以下 `describe-activations` 示例列出了有关您 AWS 账户中激活的详细信息。

```
aws ssm describe-activations
```

输出：

```
{  
  "ActivationList": [  
    {  
      "ActivationId": "5743558d-563b-4457-8682-d16c3EXAMPLE",  
      "Description": "Example1",  
      "IamRole": "HybridWebServersRole",  
      "RegistrationLimit": 5,  
      "RegistrationsCount": 5,  
    }  
  ]  
}
```

```

    "ExpirationDate": 1584316800.0,
    "Expired": false,
    "CreateDate": 1581954699.792
  },
  {
    "ActivationId": "3ee0322b-f62d-40eb-b672-13ebfEXAMPLE",
    "Description": "Example2",
    "IamRole": "HybridDatabaseServersRole",
    "RegistrationLimit": 5,
    "RegistrationsCount": 5,
    "ExpirationDate": 1580515200.0,
    "Expired": true,
    "CreateDate": 1578064132.002
  },
]
}

```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[步骤 4：为混合环境创建托管式实例激活](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeActivations](#)中的。

## describe-association-execution-targets

以下代码示例演示了如何使用 describe-association-execution-targets。

### AWS CLI

获取关联执行的详细信息

以下 describe-association-execution-targets 示例描述指定的关联执行。

```

aws ssm describe-association-execution-targets \
  --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
  --execution-id "7abb6378-a4a5-4f10-8312-0123456789ab"

```

输出：

```

{
  "AssociationExecutionTargets": [
    {
      "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
      "AssociationVersion": "1",

```

```

    "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
    "ResourceId": "i-1234567890abcdef0",
    "ResourceType": "ManagedInstance",
    "Status": "Success",
    "DetailedStatus": "Success",
    "LastExecutionDate": 1550505538.497,
    "OutputSource": {
      "OutputSourceId": "97fff367-fc5a-4299-aed8-0123456789ab",
      "OutputSourceType": "RunCommand"
    }
  }
]
}

```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[查看关联历史记录](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAssociationExecutionTargets](#) 中的。

## describe-association-executions

以下代码示例演示了如何使用 describe-association-executions。

### AWS CLI

示例 1：获取关联所有执行的详细信息

以下 describe-association-executions 示例描述指定关联的所有执行。

```

aws ssm describe-association-executions \
  --association-id "8dfe3659-4309-493a-8755-0123456789ab"

```

输出：

```

{
  "AssociationExecutions": [
    {
      "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
      "AssociationVersion": "1",
      "ExecutionId": "474925ef-1249-45a2-b93d-0123456789ab",
      "Status": "Success",
      "DetailedStatus": "Success",
      "CreatedTime": 1550505827.119,
      "ResourceCountByStatus": "{Success=1}"
    }
  ]
}

```

```

    },
    {
      "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
      "AssociationVersion": "1",
      "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
      "Status": "Success",
      "DetailedStatus": "Success",
      "CreatedTime": 1550505536.843,
      "ResourceCountByStatus": "{Success=1}"
    },
    ...
  ]
}

```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[查看关联历史记录](#)。

示例 2：获取特定日期和时间之后关联的所有执行的详细信息

以下 describe-association-executions 示例描述指定日期和时间之后关联的所有执行。

```

aws ssm describe-association-executions \
  --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
  --filters "Key=CreatedTime,Value=2019-02-18T16:00:00Z,Type=GREATER_THAN"

```

输出：

```

{
  "AssociationExecutions": [
    {
      "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
      "AssociationVersion": "1",
      "ExecutionId": "474925ef-1249-45a2-b93d-0123456789ab",
      "Status": "Success",
      "DetailedStatus": "Success",
      "CreatedTime": 1550505827.119,
      "ResourceCountByStatus": "{Success=1}"
    },
    {
      "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
      "AssociationVersion": "1",
      "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
      "Status": "Success",
      "DetailedStatus": "Success",

```



```

        "CreatedTime": 1550505536.843,
        "ResourceCountByStatus": "{Success=1}"
    },
    ...
]
}

```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[查看关联历史记录](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAssociationExecutions](#) 中的。

## describe-association

以下代码示例演示了如何使用 describe-association。

### AWS CLI

示例 1：获取关联的详细信息

以下 describe-association 示例描述指定关联 ID 的关联。

```

aws ssm describe-association \
  --association-id "8dfe3659-4309-493a-8755-0123456789ab"

```

输出：

```

{
  "AssociationDescription": {
    "Name": "AWS-GatherSoftwareInventory",
    "AssociationVersion": "1",
    "Date": 1534864780.995,
    "LastUpdateAssociationDate": 1543235759.81,
    "Overview": {
      "Status": "Success",
      "AssociationStatusAggregatedCount": {
        "Success": 2
      }
    },
    "DocumentVersion": "$DEFAULT",
    "Parameters": {
      "applications": [
        "Enabled"
      ],
    },
  },
}

```

```
    "awsComponents": [
      "Enabled"
    ],
    "customInventory": [
      "Enabled"
    ],
    "files": [
      ""
    ],
    "instanceDetailedInformation": [
      "Enabled"
    ],
    "networkConfig": [
      "Enabled"
    ],
    "services": [
      "Enabled"
    ],
    "windowsRegistry": [
      ""
    ],
    "windowsRoles": [
      "Enabled"
    ],
    "windowsUpdates": [
      "Enabled"
    ]
  },
  "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
  "Targets": [
    {
      "Key": "InstanceIds",
      "Values": [
        "*"
      ]
    }
  ],
  "ScheduleExpression": "rate(24 hours)",
  "LastExecutionDate": 1550501886.0,
  "LastSuccessfulExecutionDate": 1550501886.0,
  "AssociationName": "Inventory-Association"
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[编辑和创建关联的新版本](#)。

示例 2：获取特定实例和文档的关联的详细信息

以下 describe-association 示例描述实例和文档之间的关联。

```
aws ssm describe-association \  
  --instance-id "i-1234567890abcdef0" \  
  --name "AWS-UpdateSSMAgent"
```

输出：

```
{  
  "AssociationDescription": {  
    "Status": {  
      "Date": 1487876122.564,  
      "Message": "Associated with AWS-UpdateSSMAgent",  
      "Name": "Associated"  
    },  
    "Name": "AWS-UpdateSSMAgent",  
    "InstanceId": "i-1234567890abcdef0",  
    "Overview": {  
      "Status": "Pending",  
      "DetailedStatus": "Associated",  
      "AssociationStatusAggregatedCount": {  
        "Pending": 1  
      }  
    },  
    "AssociationId": "d8617c07-2079-4c18-9847-1234567890ab",  
    "DocumentVersion": "$DEFAULT",  
    "LastUpdateAssociationDate": 1487876122.564,  
    "Date": 1487876122.564,  
    "Targets": [  
      {  
        "Values": [  
          "i-1234567890abcdef0"  
        ],  
        "Key": "InstanceIds"  
      }  
    ]  
  }  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[编辑和创建关联的新版本](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAssociation](#) 中的。

## describe-automation-executions

以下代码示例演示了如何使用 describe-automation-executions。

### AWS CLI

#### 描述自动化执行

以下 describe-automation-executions 示例显示了有关自动化执行的详细信息。

```
aws ssm describe-automation-executions \
  --filters Key=ExecutionId,Values=73c8eef8-f4ee-4a05-820c-e354fEXAMPLE
```

#### 输出：

```
{
  "AutomationExecutionMetadataList": [
    {
      "AutomationExecutionId": "73c8eef8-f4ee-4a05-820c-e354fEXAMPLE",
      "DocumentName": "AWS-StartEC2Instance",
      "DocumentVersion": "1",
      "AutomationExecutionStatus": "Success",
      "ExecutionStartTime": 1583737233.748,
      "ExecutionEndTime": 1583737234.719,
      "ExecutedBy": "arn:aws:sts::29884EXAMPLE:assumed-role/mw_service_role/OrchestrationService",
      "LogFile": "",
      "Outputs": {},
      "Mode": "Auto",
      "Targets": [],
      "ResolvedTargets": {
        "ParameterValues": [],
        "Truncated": false
      },
      "AutomationType": "Local"
    }
  ]
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[运行简单的自动化工作流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAutomationExecutions](#) 中的。

## describe-automation-step-executions

以下代码示例演示了如何使用 describe-automation-step-executions。

### AWS CLI

示例 1：描述自动化执行的所有步骤

以下 describe-automation-step-executions 示例显示有关自动化执行步骤的详细信息。

```
aws ssm describe-automation-step-executions \
  --automation-execution-id 73c8eef8-f4ee-4a05-820c-e354fEXAMPLE
```

输出：

```
{
  "StepExecutions": [
    {
      "StepName": "startInstances",
      "Action": "aws:changeInstanceState",
      "ExecutionStartTime": 1583737234.134,
      "ExecutionEndTime": 1583737234.672,
      "StepStatus": "Success",
      "Inputs": {
        "DesiredState": "\"running\"",
        "InstanceIds": "[\"i-0cb99161f6EXAMPLE\"]"
      },
      "Outputs": {
        "InstanceStates": [
          "running"
        ]
      },
      "StepExecutionId": "95e70479-cf20-4d80-8018-7e4e2EXAMPLE",
      "OverriddenParameters": {}
    }
  ]
}
```

示例 2：描述自动化执行的特定步骤

以下 `describe-automation-step-executions` 示例显示了有关自动化执行中特定步骤的详细信息。

```
aws ssm describe-automation-step-executions \  
  --automation-execution-id 73c8eef8-f4ee-4a05-820c-e354fEXAMPLE \  
  --filters Key=StepExecutionId,Values=95e70479-cf20-4d80-8018-7e4e2EXAMPLE
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[分步运行自动化工作流程（命令行）](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeAutomationStepExecutions](#)中的。

## describe-available-patches

以下代码示例演示了如何使用 `describe-available-patches`。

### AWS CLI

获取可用补丁

以下 `describe-available-patches` 示例检索有关 MSRC 严重性为“严重”的所有 Windows Server 2019 可用补丁的详细信息。

```
aws ssm describe-available-patches \  
  --filters "Key=PRODUCT,Values=WindowsServer2019"  
  "Key=MSRC_SEVERITY,Values=Critical"
```

输出：

```
{  
  "Patches": [  
    {  
      "Id": "fe6bd8c2-3752-4c8b-ab3e-1a7ed08767ba",  
      "ReleaseDate": 1544047205.0,  
      "Title": "2018-11 Update for Windows Server 2019 for x64-based Systems  
(KB4470788)",  
      "Description": "Install this update to resolve issues in Windows. For a  
complete listing of the issues that are included in this update, see the associated  
Microsoft Knowledge Base article for more information. After you install this item,  
you may have to restart your computer.",  
      "ContentUrl": "https://support.microsoft.com/en-us/kb/4470788",  
      "Vendor": "Microsoft",
```

```

        "ProductFamily": "Windows",
        "Product": "WindowsServer2019",
        "Classification": "SecurityUpdates",
        "MsrcSeverity": "Critical",
        "KbNumber": "KB4470788",
        "MsrcNumber": "",
        "Language": "All"
    },
    {
        "Id": "c96115e1-5587-4115-b851-22baa46a3f11",
        "ReleaseDate": 1549994410.0,
        "Title": "2019-02 Security Update for Adobe Flash Player for Windows
Server 2019 for x64-based Systems (KB4487038)",
        "Description": "A security issue has been identified in a Microsoft
software product that could affect your system. You can help protect your system
by installing this update from Microsoft. For a complete listing of the issues that
are included in this update, see the associated Microsoft Knowledge Base article.
After you install this update, you may have to restart your system.",
        "ContentUrl": "https://support.microsoft.com/en-us/kb/4487038",
        "Vendor": "Microsoft",
        "ProductFamily": "Windows",
        "Product": "WindowsServer2019",
        "Classification": "SecurityUpdates",
        "MsrcSeverity": "Critical",
        "KbNumber": "KB4487038",
        "MsrcNumber": "",
        "Language": "All"
    },
    ...
]
}

```

### 获取特定补丁的详细信息

以下 `describe-available-patches` 示例将检索有关指定补丁的详细信息。

```
aws ssm describe-available-patches \
  --filters "Key=PATCH_ID,Values=KB4480979"
```

输出：

```
{
  "Patches": [
```

```
{
  "Id": "680861e3-fb75-432e-818e-d72e5f2be719",
  "ReleaseDate": 1546970408.0,
  "Title": "2019-01 Security Update for Adobe Flash Player for Windows
Server 2016 for x64-based Systems (KB4480979)",
  "Description": "A security issue has been identified in a Microsoft
software product that could affect your system. You can help protect your system
by installing this update from Microsoft. For a complete listing of the issues that
are included in this update, see the associated Microsoft Knowledge Base article.
After you install this update, you may have to restart your system.",
  "ContentUrl": "https://support.microsoft.com/en-us/kb/4480979",
  "Vendor": "Microsoft",
  "ProductFamily": "Windows",
  "Product": "WindowsServer2016",
  "Classification": "SecurityUpdates",
  "MsrcSeverity": "Critical",
  "KbNumber": "KB4480979",
  "MsrcNumber": "",
  "Language": "All"
}
]
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的 [Patch Manager 工作原理](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeAvailablePatches](#) 中的。

## describe-document-permission

以下代码示例演示了如何使用 describe-document-permission。

### AWS CLI

#### 描述文档权限

以下 describe-document-permission 示例显示有关公开共享 Systems Manager 文档的权限详细信息。

```
aws ssm describe-document-permission \
  --name "Example" \
  --permission-type "Share"
```

输出：



```
{
  "AccountIds": [
    "all"
  ],
  "AccountSharingInfoList": [
    {
      "AccountId": "all",
      "SharedDocumentVersion": "$DEFAULT"
    }
  ]
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[共享 Systems Manager 文档](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDocumentPermission](#) 中的。

## describe-document

以下代码示例演示了如何使用 describe-document。

### AWS CLI

显示文档的详细信息

以下 describe-document 示例显示了有关您 AWS 账户中 Systems Manager 文档的详细信息。

```
aws ssm describe-document \
  --name "Example"
```

输出：

```
{
  "Document": {
    "Hash": "fc2410281f40779e694a8b95975d0f9f316da8a153daa94e3d9921102EXAMPLE",
    "HashType": "Sha256",
    "Name": "Example",
    "Owner": "29884EXAMPLE",
    "CreateDate": 1583257938.266,
    "Status": "Active",
    "DocumentVersion": "1",
    "Description": "Document Example",
    "Parameters": [
```

```
    {
      "Name": "AutomationAssumeRole",
      "Type": "String",
      "Description": "(Required) The ARN of the role that allows
Automation to perform the actions on your behalf. If no role is specified, Systems
Manager Automation uses your IAM permissions to execute this document.",
      "DefaultValue": ""
    },
    {
      "Name": "InstanceId",
      "Type": "String",
      "Description": "(Required) The ID of the Amazon EC2 instance.",
      "DefaultValue": ""
    }
  ],
  "PlatformTypes": [
    "Windows",
    "Linux"
  ],
  "DocumentType": "Automation",
  "SchemaVersion": "0.3",
  "LatestVersion": "1",
  "DefaultVersion": "1",
  "DocumentFormat": "YAML",
  "Tags": []
}
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[创建 Systems Manager 文档](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeDocument](#) 中的。

## describe-effective-instance-associations

以下代码示例演示了如何使用 describe-effective-instance-associations。

### AWS CLI

获取实例有效关联的详细信息

以下 describe-effective-instance-associations 示例检索有关实例有效关联的详细信息。

命令:

```
aws ssm describe-effective-instance-associations --instance-id "i-1234567890abcdef0"
```

输出：

```
{
  "Associations": [
    {
      "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
      "InstanceId": "i-1234567890abcdef0",
      "Content": "{\n  \"schemaVersion\": \"1.2\",\n  \"description\":\n  \"Update the Amazon SSM Agent to the latest version or specified version.\",\n  \"parameters\": {\n    \"version\": {\n      \"default\": \"\",\n      \"description\": \"(Optional) A specific version of the Amazon SSM Agent\n  to install. If not specified, the agent will be updated to the latest version.\",\n      \"type\": \"String\"\n    },\n    \"allowDowngrade\": {\n      \"default\": \"false\",\n      \"description\": \"(Optional)\n  Allow the Amazon SSM Agent service to be downgraded to an earlier version. If\n  set to false, the service can be upgraded to newer versions only (default). If\n  set to true, specify the earlier version.\",\n      \"type\": \"String\",\n      \"allowedValues\": [\n        \"true\",\n        \"false\"\n      ]\n    },\n    \"runtimeConfig\": {\n      \"aws:updateSsmAgent\": {\n        \"properties\": [\n          {\n            \"agentName\": \"amazon-ssm-agent\",\n            \"source\":\n            \"https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/ssm-agent-manifest.json\",\n            \"allowDowngrade\": \"{{ allowDowngrade }}\",\n            \"targetVersion\": \"{{ version }}\"\n          }\n        ]\n      }\n    }\n  }\n  \"AssociationVersion\": \"1\"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeEffectiveInstanceAssociations](#)中的。

## describe-effective-patches-for-patch-baseline

以下代码示例演示了如何使用 describe-effective-patches-for-patch-baseline。

### AWS CLI

示例 1：获取自定义补丁基准定义的所有补丁

以下describe-effective-patches-for-patch-baseline示例返回当前 AWS 账户中由自定义补丁基准定义的补丁。请注意，对于自定义基准，--baseline-id 只需要 ID。

```
aws ssm describe-effective-patches-for-patch-baseline \  
  --baseline-id "pb-08b654cf9b9681f04"
```

输出：

```
{  
  "EffectivePatches": [  
    {  
      "Patch": {  
        "Id": "fe6bd8c2-3752-4c8b-ab3e-1a7ed08767ba",  
        "ReleaseDate": 1544047205.0,  
        "Title": "2018-11 Update for Windows Server 2019 for x64-based  
Systems (KB4470788)",  
        "Description": "Install this update to resolve issues in Windows.  
For a complete listing of the issues that are included in this update, see the  
associated Microsoft Knowledge Base article for more information. After you install  
this item, you may have to restart your computer.",  
        "ContentUrl": "https://support.microsoft.com/en-us/kb/4470788",  
        "Vendor": "Microsoft",  
        "ProductFamily": "Windows",  
        "Product": "WindowsServer2019",  
        "Classification": "SecurityUpdates",  
        "MsrcSeverity": "Critical",  
        "KbNumber": "KB4470788",  
        "MsrcNumber": "",  
        "Language": "All"  
      },  
      "PatchStatus": {  
        "DeploymentStatus": "APPROVED",  
        "ComplianceLevel": "CRITICAL",  
        "ApprovalDate": 1544047205.0  
      }  
    },  
    {  
      "Patch": {  
        "Id": "915a6b1a-f556-4d83-8f50-b2e75a9a7e58",  
        "ReleaseDate": 1549994400.0,  
        "Title": "2019-02 Cumulative Update for .NET Framework 3.5 and 4.7.2  
for Windows Server 2019 for x64 (KB4483452)",
```

```

        "Description": "A security issue has been identified in a Microsoft
software product that could affect your system. You can help protect your system by
installing this update from Microsoft. For a complete listing of the issues that
are included in this update, see the associated Microsoft Knowledge Base article.
After you install this update, you may have to restart your system.",
        "ContentUrl": "https://support.microsoft.com/en-us/kb/4483452",
        "Vendor": "Microsoft",
        "ProductFamily": "Windows",
        "Product": "WindowsServer2019",
        "Classification": "SecurityUpdates",
        "MsrcSeverity": "Important",
        "KbNumber": "KB4483452",
        "MsrcNumber": "",
        "Language": "All"
    },
    "PatchStatus": {
        "DeploymentStatus": "APPROVED",
        "ComplianceLevel": "CRITICAL",
        "ApprovalDate": 1549994400.0
    }
},
...
],
"NextToken": "--token string truncated--"
}

```

## 示例 2：获取由 AWS 托管补丁基准定义的所有补丁

以下 `describe-effective-patches-for-patch-baseline` 示例返回由 AWS 托管补丁基准定义的修补程序。请注意，对于 AWS 托管基准，需要完整的基准 ARN `--baseline-id`

```

aws ssm describe-effective-patches-for-patch-baseline \
  --baseline-id "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-020d361a05defe4ed"

```

有关输出示例，请参阅示例 1。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[如何选择安全补丁](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeEffectivePatchesForPatchBaseline](#) 中的。

## describe-instance-associations-status

以下代码示例演示了如何使用 `describe-instance-associations-status`。

### AWS CLI

描述实例关联的状态

此示例显示实例关联的详细信息。

命令:

```
aws ssm describe-instance-associations-status --instance-id "i-1234567890abcdef0"
```

输出:

```
{
  "InstanceAssociationStatusInfos": [
    {
      "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
      "Name": "AWS-GatherSoftwareInventory",
      "DocumentVersion": "1",
      "AssociationVersion": "1",
      "InstanceId": "i-1234567890abcdef0",
      "ExecutionDate": 1550501886.0,
      "Status": "Success",
      "ExecutionSummary": "1 out of 1 plugin processed, 1 success, 0 failed, 0
      timedout, 0 skipped. ",
      "AssociationName": "Inventory-Association"
    },
    {
      "AssociationId": "5c5a31f6-6dae-46f9-944c-0123456789ab",
      "Name": "AWS-UpdateSSMAgent",
      "DocumentVersion": "1",
      "AssociationVersion": "1",
      "InstanceId": "i-1234567890abcdef0",
      "ExecutionDate": 1550505828.548,
      "Status": "Success",
      "DetailedStatus": "Success",
      "AssociationName": "UpdateSSMAgent"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInstanceAssociationsStatus](#)中的。

## describe-instance-information

以下代码示例演示了如何使用 describe-instance-information。

### AWS CLI

#### 示例 1：描述托管式实例信息

以下 describe-instance-information 示例检索每个托管式实例的详细信息。

```
aws ssm describe-instance-information
```

#### 示例 2：描述有关特定托管式实例的信息

以下 describe-instance-information 示例显示托管式实例 i-028ea792daEXAMPLE 的详细信息。

```
aws ssm describe-instance-information \  
  --filters "Key=InstanceIds,Values=i-028ea792daEXAMPLE"
```

#### 示例 3：描述有关具有特定标签键的托管式实例的信息

以下 describe-instance-information 示例显示具有标签键 DEV 的托管式实例的详细信息。

```
aws ssm describe-instance-information \  
  --filters "Key=tag-key,Values=DEV"
```

输出：

```
{  
  "InstanceInformationList": [  
    {  
      "InstanceId": "i-028ea792daEXAMPLE",  
      "PingStatus": "Online",  
      "LastPingDateTime": 1582221233.421,  
      "AgentVersion": "2.3.842.0",  
      "IsLatestVersion": true,  
      "PlatformType": "Linux",  
      "PlatformName": "SLES",  
      "PlatformVersion": "15.1",
```

```

    "ResourceType": "EC2Instance",
    "IPAddress": "192.0.2.0",
    "ComputerName": "ip-198.51.100.0.us-east-2.compute.internal",
    "AssociationStatus": "Success",
    "LastAssociationExecutionDate": 1582220806.0,
    "LastSuccessfulAssociationExecutionDate": 1582220806.0,
    "AssociationOverview": {
      "DetailedStatus": "Success",
      "InstanceAssociationStatusAggregatedCount": {
        "Success": 2
      }
    }
  }
]
}

```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[托管式实例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeInstanceInformation](#) 中的。

## describe-instance-patch-states-for-patch-group

以下代码示例演示了如何使用 `describe-instance-patch-states-for-patch-group`。

### AWS CLI

#### 示例 1：获取补丁组的实例状态

以下 `describe-instance-patch-states-for-patch-group` 示例检索有关指定补丁组每个实例的补丁摘要状态的详细信息。

```
aws ssm describe-instance-patch-states-for-patch-group \
  --patch-group "Production"
```

输出：

```

{
  "InstancePatchStates": [
    {
      "InstanceId": "i-02573cafcfEXAMPLE",
      "PatchGroup": "Production",
      "BaselineId": "pb-0c10e65780EXAMPLE",
      "SnapshotId": "a3f5ff34-9bc4-4d2c-a665-4d1c1EXAMPLE",

```



```

    "OwnerInformation": "",
    "InstalledCount": 32,
    "InstalledOtherCount": 1,
    "InstalledPendingRebootCount": 0,
    "InstalledRejectedCount": 0,
    "MissingCount": 2,
    "FailedCount": 0,
    "UnreportedNotApplicableCount": 2671,
    "NotApplicableCount": 400,
    "OperationStartTime": "2021-08-04T11:03:50.590000-07:00",
    "OperationEndTime": "2021-08-04T11:04:21.555000-07:00",
    "Operation": "Scan",
    "RebootOption": "NoReboot",
    "CriticalNonCompliantCount": 0,
    "SecurityNonCompliantCount": 1,
    "OtherNonCompliantCount": 0
  },
  {
    "InstanceId": "i-0471e04240EXAMPLE",
    "PatchGroup": "Production",
    "BaselineId": "pb-09ca3fb51fEXAMPLE",
    "SnapshotId": "05d8ffb0-1bbe-4812-ba2d-d9b7bEXAMPLE",
    "OwnerInformation": "",
    "InstalledCount": 32,
    "InstalledOtherCount": 1,
    "InstalledPendingRebootCount": 0,
    "InstalledRejectedCount": 0,
    "MissingCount": 2,
    "FailedCount": 0,
    "UnreportedNotApplicableCount": 2671,
    "NotApplicableCount": 400,
    "OperationStartTime": "2021-08-04T22:06:20.340000-07:00",
    "OperationEndTime": "2021-08-04T22:07:11.220000-07:00",
    "Operation": "Scan",
    "RebootOption": "NoReboot",
    "CriticalNonCompliantCount": 0,
    "SecurityNonCompliantCount": 1,
    "OtherNonCompliantCount": 0
  }
]
}

```

示例 2：获取缺失五个补丁以上的补丁组的实例状态

以下 `describe-instance-patch-states-for-patch-group` 示例针对缺失五个补丁以上的实例的指定补丁组，检索补丁摘要状态详细信息。

```
aws ssm describe-instance-patch-states-for-patch-group \
  --filters Key=MissingCount,Type=GreaterThan,Values=5 \
  --patch-group "Production"
```

输出：

```
{
  "InstancePatchStates": [
    {
      "InstanceId": "i-02573cafcfEXAMPLE",
      "PatchGroup": "Production",
      "BaselineId": "pb-0c10e65780EXAMPLE",
      "SnapshotId": "a3f5ff34-9bc4-4d2c-a665-4d1c1EXAMPLE",
      "OwnerInformation": "",
      "InstalledCount": 46,
      "InstalledOtherCount": 4,
      "InstalledPendingRebootCount": 1,
      "InstalledRejectedCount": 1,
      "MissingCount": 7,
      "FailedCount": 0,
      "UnreportedNotApplicableCount": 232,
      "NotApplicableCount": 654,
      "OperationStartTime": "2021-08-04T11:03:50.590000-07:00",
      "OperationEndTime": "2021-08-04T11:04:21.555000-07:00",
      "Operation": "Scan",
      "RebootOption": "NoReboot",
      "CriticalNonCompliantCount": 0,
      "SecurityNonCompliantCount": 1,
      "OtherNonCompliantCount": 1
    }
  ]
}
```

示例 3：获取需要重启的实例少于 10 个的补丁组的实例状态

以下 `describe-instance-patch-states-for-patch-group` 示例针对需要重启的实例少于 10 个实例的指定补丁组，检索补丁摘要状态的详细信息。

```
aws ssm describe-instance-patch-states-for-patch-group \
```

```
--filters Key=InstalledPendingRebootCount,Type=LessThan,Values=10 \  
--patch-group "Production"
```

输出：

```
{  
  "InstancePatchStates": [  
    {  
      "InstanceId": "i-02573cafcfEXAMPLE",  
      "BaselineId": "pb-0c10e65780EXAMPLE",  
      "SnapshotId": "a3f5ff34-9bc4-4d2c-a665-4d1c1EXAMPLE",  
      "PatchGroup": "Production",  
      "OwnerInformation": "",  
      "InstalledCount": 32,  
      "InstalledOtherCount": 1,  
      "InstalledPendingRebootCount": 4,  
      "InstalledRejectedCount": 0,  
      "MissingCount": 2,  
      "FailedCount": 0,  
      "UnreportedNotApplicableCount": 846,  
      "NotApplicableCount": 212,  
      "OperationStartTime": "2021-08-04T11:03:50.590000-07:00",  
      "OperationEndTime": "2021-08-06T11:04:21.555000-07:00",  
      "Operation": "Scan",  
      "RebootOption": "NoReboot",  
      "CriticalNonCompliantCount": 0,  
      "SecurityNonCompliantCount": 1,  
      "OtherNonCompliantCount": 0  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[了解补丁合规性状态值](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInstancePatchStatesForPatchGroup](#)中的。

## describe-instance-patch-states

以下代码示例演示了如何使用 describe-instance-patch-states。

## AWS CLI

### 获取实例的补丁摘要状态

此 `describe-instance-patch-states` 示例获取实例的补丁摘要状态。

```
aws ssm describe-instance-patch-states \  
  --instance-ids "i-1234567890abcdef0"
```

输出：

```
{  
  "InstancePatchStates": [  
    {  
      "InstanceId": "i-1234567890abcdef0",  
      "PatchGroup": "my-patch-group",  
      "BaselineId": "pb-0713accee01234567",  
      "SnapshotId": "521c3536-930c-4aa9-950e-01234567abcd",  
      "CriticalNonCompliantCount": 2,  
      "SecurityNonCompliantCount": 2,  
      "OtherNonCompliantCount": 1,  
      "InstalledCount": 123,  
      "InstalledOtherCount": 334,  
      "InstalledPendingRebootCount": 0,  
      "InstalledRejectedCount": 0,  
      "MissingCount": 1,  
      "FailedCount": 2,  
      "UnreportedNotApplicableCount": 11,  
      "NotApplicableCount": 2063,  
      "OperationStartTime": "2021-05-03T11:00:56-07:00",  
      "OperationEndTime": "2021-05-03T11:01:09-07:00",  
      "Operation": "Scan",  
      "LastNoRebootInstallOperationTime": "2020-06-14T12:17:41-07:00",  
      "RebootOption": "RebootIfNeeded"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[关于补丁合规性](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeInstancePatchStates](#) 中的。

## describe-instance-patches

以下代码示例演示了如何使用 describe-instance-patches。

### AWS CLI

示例 1：获取实例的补丁状态详细信息

以下 describe-instance-patches 示例将检索有关指定实例补丁的详细信息。

```
aws ssm describe-instance-patches \  
  --instance-id "i-1234567890abcdef0"
```

输出：

```
{  
  "Patches": [  
    {  
      "Title": "2019-01 Security Update for Adobe Flash Player for Windows  
Server 2016 for x64-based Systems (KB4480979)",  
      "KBId": "KB4480979",  
      "Classification": "SecurityUpdates",  
      "Severity": "Critical",  
      "State": "Installed",  
      "InstalledTime": "2019-01-09T00:00:00+00:00"  
    },  
    {  
      "Title": "",  
      "KBId": "KB4481031",  
      "Classification": "",  
      "Severity": "",  
      "State": "InstalledOther",  
      "InstalledTime": "2019-02-08T00:00:00+00:00"  
    },  
    ...  
  ],  
  "NextToken": "--token string truncated--"  
}
```

示例 2：获取实例的处于“缺失”状态的补丁列表

以下 describe-instance-patches 示例检索有关指定实例处于“缺失”状态的补丁的信息。

```
aws ssm describe-instance-patches \  
  --instance-id "i-1234567890abcdef0" \  
  --filters Key=State,Values=Missing
```

输出：

```
{  
  "Patches": [  
    {  
      "Title": "Windows Malicious Software Removal Tool x64 - February 2019  
(KB890830)",  
      "KBId": "KB890830",  
      "Classification": "UpdateRollups",  
      "Severity": "Unspecified",  
      "State": "Missing",  
      "InstalledTime": "1970-01-01T00:00:00+00:00"  
    },  
    ...  
  ],  
  "NextToken": "--token string truncated--"  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[关于补丁合规性状态](#)。

示例 3：获取自 InstalledTime 为实例指定以来安装的补丁列表

以下 describe-instance-patches 示例通过组合使用 --filters 和 --query，检索指定实例自指定时间以来所安装补丁的信息。

```
aws ssm describe-instance-patches \  
  --instance-id "i-1234567890abcdef0" \  
  --filters Key=State,Values=Installed \  
  --query "Patches[?InstalledTime >= `2023-01-01T16:00:00`]"
```

输出：

```
{  
  "Patches": [  
    {  
      "Title": "2023-03 Cumulative Update for Windows Server 2019 (1809) for  
x64-based Systems (KB5023702)",  
      "KBId": "KB5023702",  
      "Classification": "UpdateRollups",  
      "Severity": "Unspecified",  
      "State": "Installed",  
      "InstalledTime": "2023-03-14T00:00:00+00:00"  
    }  
  ]  
}
```

```
        "Classification": "SecurityUpdates",
        "Severity": "Critical",
        "State": "Installed",
        "InstalledTime": "2023-03-16T11:00:00+00:00"
    },
    ...
],
"NextToken": "--token string truncated--"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInstancePatches](#)中的。

## describe-inventory-deletions

以下代码示例演示了如何使用 describe-inventory-deletions。

### AWS CLI

#### 要删除库存

此示例检索库存删除操作的详细信息。

命令:

```
aws ssm describe-inventory-deletions
```

输出:

```
{
  "InventoryDeletions": [
    {
      "DeletionId": "6961492a-8163-44ec-aa1e-01234567850",
      "TypeName": "Custom:RackInformation",
      "DeletionStartTime": 1550254911.0,
      "LastStatus": "InProgress",
      "LastStatusMessage": "The Delete is in progress",
      "DeletionSummary": {
        "TotalCount": 0,
        "RemainingCount": 0,
        "SummaryItems": []
      },
      "LastStatusUpdateTime": 1550254911.0
    },
  ],
}
```

```
{
  "DeletionId": "d72ac9e8-1f60-4d40-b1c6-987654321c4d",
  "TypeName": "Custom:RackInfo",
  "DeletionStartTime": 1550254859.0,
  "LastStatus": "InProgress",
  "LastStatusMessage": "The Delete is in progress",
  "DeletionSummary": {
    "TotalCount": 1,
    "RemainingCount": 1,
    "SummaryItems": [
      {
        "Version": "1.0",
        "Count": 1,
        "RemainingCount": 1
      }
    ]
  },
  "LastStatusUpdateTime": 1550254859.0
}
]
```

### 获取特定库存删除的详情

此示例检索特定库存删除操作的详细信息。

#### 命令:

```
aws ssm describe-inventory-deletions --deletion-id "d72ac9e8-1f60-4d40-
b1c6-987654321c4d"
```

#### 输出:

```
{
  "InventoryDeletions": [
    {
      "DeletionId": "d72ac9e8-1f60-4d40-b1c6-987654321c4d",
      "TypeName": "Custom:RackInfo",
      "DeletionStartTime": 1550254859.0,
      "LastStatus": "InProgress",
      "LastStatusMessage": "The Delete is in progress",
      "DeletionSummary": {
        "TotalCount": 1,
```



```

        "RemainingCount": 1,
        "SummaryItems": [
            {
                "Version": "1.0",
                "Count": 1,
                "RemainingCount": 1
            }
        ]
    },
    "LastStatusUpdateTime": 1550254859.0
}
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInventoryDeletions](#)中的。

## describe-maintenance-window-execution-task-invocations

以下代码示例演示了如何使用 `describe-maintenance-window-execution-task-invocations`。

### AWS CLI

获取为执行维护时段任务而执行的特定任务调用

以下 `describe-maintenance-window-execution-task-invocations` 示例列出作为指定维护时段执行组成部分来执行的指定任务的调用。

```

aws ssm describe-maintenance-window-execution-task-invocations \
  --window-execution-id "518d5565-5969-4cca-8f0e-da3b2a638355" \
  --task-id "ac0c6ae1-daa3-4a89-832e-d384503b6586"

```

输出：

```

{
  "WindowExecutionTaskInvocationIdentities": [
    {
      "Status": "SUCCESS",
      "Parameters": "{\"documentName\":\"AWS-RunShellScript\",\"instanceIds\":[\"i-0000293ffd8c57862\"],\"parameters\":{\"commands\":[\"df\"]},\"maxConcurrency\":\"1\",\"maxErrors\":\"1\"}",
      "InvocationId": "e274b6e1-fe56-4e32-bd2a-8073c6381d8b",
    }
  ]
}

```

```
        "StartTime": 1487692834.723,  
        "EndTime": 1487692834.871,  
        "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2a638355",  
        "TaskExecutionId": "ac0c6ae1-daa3-4a89-832e-d384503b6586"  
    }  
]  
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的“[查看有关任务和任务执行的信息 \(AWS CLI\)](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeMaintenanceWindowExecutionTaskInvocations](#)中的。

## describe-maintenance-window-execution-tasks

以下代码示例演示了如何使用 describe-maintenance-window-execution-tasks。

### AWS CLI

列出与维护时段执行相关的所有任务

以下 ssm describe-maintenance-window-execution-tasks 示例列出与指定维护时段执行相关的任务。

```
aws ssm describe-maintenance-window-execution-tasks \  
  --window-execution-id "518d5565-5969-4cca-8f0e-da3b2EXAMPLE"
```

输出：

```
{  
  "WindowExecutionTaskIdentities": [  
    {  
      "Status": "SUCCESS",  
      "TaskArn": "AWS-RunShellScript",  
      "StartTime": 1487692834.684,  
      "TaskType": "RUN_COMMAND",  
      "EndTime": 1487692835.005,  
      "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2EXAMPLE",  
      "TaskExecutionId": "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE"  
    }  
  ]  
}
```

```
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的“[查看有关任务和任务执行的信息 \(AWS CLI\)](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeMaintenanceWindowExecutionTasks](#)中的。

## describe-maintenance-window-executions

以下代码示例演示了如何使用 describe-maintenance-window-executions。

### AWS CLI

示例 1：列出维护时段内的所有执行

以下 describe-maintenance-window-executions 示例列出指定维护时段的所有执行。

```
aws ssm describe-maintenance-window-executions \
  --window-id "mw-ab12cd34eEXAMPLE"
```

输出：

```
{
  "WindowExecutions": [
    {
      "WindowId": "mw-ab12cd34eEXAMPLE",
      "WindowExecutionId": "6027b513-64fe-4cf0-be7d-1191aEXAMPLE",
      "Status": "IN_PROGRESS",
      "StartTime": "2021-08-04T11:00:00.000000-07:00"
    },
    {
      "WindowId": "mw-ab12cd34eEXAMPLE",
      "WindowExecutionId": "ff75b750-4834-4377-8f61-b3cadEXAMPLE",
      "Status": "SUCCESS",
      "StartTime": "2021-08-03T11:00:00.000000-07:00",
      "EndTime": "2021-08-03T11:37:21.450000-07:00"
    },
    {
      "WindowId": "mw-ab12cd34eEXAMPLE",
      "WindowExecutionId": "9fac7dd9-ff21-42a5-96ad-bbc4bEXAMPLE",
```

```
        "Status": "FAILED",
        "StatusDetails": "One or more tasks in the orchestration failed.",
        "StartTime": "2021-08-02T11:00:00.000000-07:00",
        "EndTime": "2021-08-02T11:22:36.190000-07:00"
    }
]
}
```

### 示例 2：列出指定日期之前维护时段内的所有执行

以下 `describe-maintenance-window-executions` 示例列出指定日期之前指定维护时段内的所有执行。

```
aws ssm describe-maintenance-window-executions \
  --window-id "mw-ab12cd34eEXAMPLE" \
  --filters "Key=ExecutedBefore,Values=2021-08-03T00:00:00Z"
```

输出：

```
{
  "WindowExecutions": [
    {
      "WindowId": "mw-ab12cd34eEXAMPLE",
      "WindowExecutionId": "9fac7dd9-ff21-42a5-96ad-bbc4bEXAMPLE",
      "Status": "FAILED",
      "StatusDetails": "One or more tasks in the orchestration failed.",
      "StartTime": "2021-08-02T11:00:00.000000-07:00",
      "EndTime": "2021-08-02T11:22:36.190000-07:00"
    }
  ]
}
```

### 示例 3：列出指定日期之后维护时段内的所有执行

以下 `describe-maintenance-window-executions` 示例列出指定日期之后指定维护时段内的所有执行。

```
aws ssm describe-maintenance-window-executions \
  --window-id "mw-ab12cd34eEXAMPLE" \
  --filters "Key=ExecutedAfter,Values=2021-08-04T00:00:00Z"
```

输出：

```
{
  "WindowExecutions": [
    {
      "WindowId": "mw-ab12cd34eEXAMPLE",
      "WindowExecutionId": "6027b513-64fe-4cf0-be7d-1191aEXAMPLE",
      "Status": "IN_PROGRESS",
      "StartTime": "2021-08-04T11:00:00.000000-07:00"
    }
  ]
}
```

有关更多信息，请参阅 [AWS Systems Manager 用户指南中的查看有关任务和任务执行 \(AWS CLI\)](#) 的信息。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 DescribeMaintenanceWindowExecutions](#) 中的。

## describe-maintenance-window-schedule

以下代码示例演示了如何使用 `describe-maintenance-window-schedule`。

### AWS CLI

示例 1：列出维护时段即将执行的任务

以下 `describe-maintenance-window-schedule` 示例列出了指定维护时段内所有即将执行的任务。

```
aws ssm describe-maintenance-window-schedule \
  --window-id mw-ab12cd34eEXAMPLE
```

输出：

```
{
  "ScheduledWindowExecutions": [
    {
      "WindowId": "mw-ab12cd34eEXAMPLE",
      "Name": "My-First-Maintenance-Window",
      "ExecutionTime": "2020-02-19T16:00Z"
    },
    {
```

```

        "WindowId": "mw-ab12cd34eEXAMPLE",
        "Name": "My-First-Maintenance-Window",
        "ExecutionTime": "2020-02-26T16:00Z"
    },
    ...
]
}

```

示例 2：列出指定日期之前维护时段内所有即将执行的任务

以下 `describe-maintenance-window-schedule` 示例列出了在指定日期之前在指定维护时段内即将执行的所有任务。

```

aws ssm describe-maintenance-window-schedule \
  --window-id mw-0ecb1226dd7b2e9a6 \
  --filters "Key=ScheduledBefore,Values=2020-02-15T06:00:00Z"

```

示例 3：列出指定日期之后维护时段内所有即将执行的任务

以下 `describe-maintenance-window-schedule` 示例列出了在指定日期之后在指定维护时段内即将执行的所有任务。

```

aws ssm describe-maintenance-window-schedule \
  --window-id mw-0ecb1226dd7b2e9a6 \
  --filters "Key=ScheduledAfter,Values=2020-02-15T06:00:00Z"

```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的“[查看有关维护窗口的信息 \(AWS CLI\)](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeMaintenanceWindowSchedule](#) 中的。

## describe-maintenance-window-targets

以下代码示例演示了如何使用 `describe-maintenance-window-targets`。

### AWS CLI

示例 1：列出维护时段内的所有目标

以下 `describe-maintenance-window-targets` 示例列出维护时段内的所有目标。

```

aws ssm describe-maintenance-window-targets \

```

```
--window-id "mw-06cf17cbefEXAMPLE"
```

输出：

```
{
  "Targets": [
    {
      "ResourceType": "INSTANCE",
      "OwnerInformation": "Single instance",
      "WindowId": "mw-06cf17cbefEXAMPLE",
      "Targets": [
        {
          "Values": [
            "i-0000293ffdEXAMPLE"
          ],
          "Key": "InstanceIds"
        }
      ],
      "WindowTargetId": "350d44e6-28cc-44e2-951f-4b2c9EXAMPLE"
    },
    {
      "ResourceType": "INSTANCE",
      "OwnerInformation": "Two instances in a list",
      "WindowId": "mw-06cf17cbefEXAMPLE",
      "Targets": [
        {
          "Values": [
            "i-0000293ffdEXAMPLE",
            "i-0cb2b964d3EXAMPLE"
          ],
          "Key": "InstanceIds"
        }
      ],
      "WindowTargetId": "e078a987-2866-47be-bedd-d9cf4EXAMPLE"
    }
  ]
}
```

示例 2：列出匹配特定所有者信息值的维护时段的所有目标

此 `describe-maintenance-window-targets` 示例列出具有特定值的维护时段的所有目标。

```
aws ssm describe-maintenance-window-targets \
```

```
--window-id "mw-0ecb1226ddEXAMPLE" \  
--filters "Key=OwnerInformation,Values=CostCenter1"
```

输出：

```
{  
  "Targets": [  
    {  
      "WindowId": "mw-0ecb1226ddEXAMPLE",  
      "WindowTargetId": "da89dcc3-7f9c-481d-ba2b-edcb7d0057f9",  
      "ResourceType": "INSTANCE",  
      "Targets": [  
        {  
          "Key": "tag:Environment",  
          "Values": [  
            "Prod"  
          ]  
        }  
      ],  
      "OwnerInformation": "CostCenter1",  
      "Name": "ProdTarget1"  
    }  
  ]  
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的“[查看有关维护窗口的信息 \(AWS CLI\)](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeMaintenanceWindowTargets](#) 中的。

## describe-maintenance-window-tasks

以下代码示例演示了如何使用 describe-maintenance-window-tasks。

### AWS CLI

示例 1：列出维护时段内的所有任务

以下 describe-maintenance-window-tasks 示例列出指定维护时段内的所有任务。

```
aws ssm describe-maintenance-window-tasks \  
--window-id "mw-06cf17cbefEXAMPLE"
```



输出：

```
{
  "Tasks": [
    {
      "WindowId": "mw-06cf17cbefEXAMPLE",
      "WindowTaskId": "018b31c3-2d77-4b9e-bd48-c91edEXAMPLE",
      "TaskArn": "AWS-RestartEC2Instance",
      "TaskParameters": {},
      "Type": "AUTOMATION",
      "Description": "Restarting EC2 Instance for maintenance",
      "MaxConcurrency": "1",
      "MaxErrors": "1",
      "Name": "My-Automation-Example-Task",
      "Priority": 0,
      "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
      "Targets": [
        {
          "Key": "WindowTargetIds",
          "Values": [
            "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
          ]
        }
      ]
    },
    {
      "WindowId": "mw-06cf17cbefEXAMPLE",
      "WindowTaskId": "1943dee0-0a17-4978-9bf4-3cc2fEXAMPLE",
      "TaskArn": "AWS-DisableS3BucketPublicReadWrite",
      "TaskParameters": {},
      "Type": "AUTOMATION",
      "Description": "Automation task to disable read/write access on public S3 buckets",
      "MaxConcurrency": "10",
      "MaxErrors": "5",
      "Name": "My-Disable-S3-Public-Read-Write-Access-Automation-Task",
      "Priority": 0,
      "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
      "Targets": [
        {
          "Key": "WindowTargetIds",
          "Values": [
```

```

        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
      ]
    }
  ]
}

```

示例 2：列出调用 AWS-RunPowerShellScript 命令文档的维护时段的所有任务

以下 describe-maintenance-window-tasks 示例列出在调用 AWS-RunPowerShellScript 命令文档的指定维护时段内的所有任务。

```

aws ssm describe-maintenance-window-tasks \
  --window-id "mw-ab12cd34eEXAMPLE" \
  --filters "Key=TaskArn,Values=AWS-RunPowerShellScript"

```

输出：

```

{
  "Tasks": [
    {
      "WindowId": "mw-ab12cd34eEXAMPLE",
      "WindowTaskId": "0d36e6b4-3a4f-411e-adcb-3558eEXAMPLE",
      "TaskArn": "AWS-RunPowerShellScript",
      "Type": "RUN_COMMAND",
      "Targets": [
        {
          "Key": "WindowTargetIds",
          "Values": [
            "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
          ]
        }
      ],
      "TaskParameters": {},
      "Priority": 1,
      "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
      "MaxConcurrency": "1",
      "MaxErrors": "1",
      "Name": "MyTask"
    }
  ]
}

```

```
}
```

### 示例 3：列出优先级为 3 的维护时段内的所有任务

以下 `describe-maintenance-window-tasks` 示例列出指定维护时段内 `Priority` 为 3 的所有任务。

```
aws ssm describe-maintenance-window-tasks \  
  --window-id "mw-ab12cd34eEXAMPLE" \  
  --filters "Key=Priority,Values=3"
```

输出：

```
{  
  "Tasks": [  
    {  
      "WindowId": "mw-ab12cd34eEXAMPLE",  
      "WindowTaskId": "0d36e6b4-3a4f-411e-adcb-3558eEXAMPLE",  
      "TaskArn": "AWS-RunPowerShellScript",  
      "Type": "RUN_COMMAND",  
      "Targets": [  
        {  
          "Key": "WindowTargetIds",  
          "Values": [  
            "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"  
          ]  
        }  
      ],  
      "TaskParameters": {},  
      "Priority": 3,  
      "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/  
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",  
      "MaxConcurrency": "1",  
      "MaxErrors": "1",  
      "Name": "MyRunCommandTask"  
    },  
    {  
      "WindowId": "mw-ab12cd34eEXAMPLE",  
      "WindowTaskId": "ee45feff-ad65-4a6c-b478-5cab8EXAMPLE",  
      "TaskArn": "AWS-RestartEC2Instance",  
      "Type": "AUTOMATION",  
      "Targets": [  
        {
```

```

        "Key": "WindowTargetIds",
        "Values": [
            "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
        ]
    },
    ],
    "TaskParameters": {},
    "Priority": 3,
    "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
    "MaxConcurrency": "10",
    "MaxErrors": "5",
    "Name": "My-Automation-Task",
    "Description": "A description for my Automation task"
}
]
}

```

示例 4：列出优先级为 1 并使用 Run Command 的维护时段内的所有任务

此 `describe-maintenance-window-tasks` 示例列出指定维护时段内 `Priority` 为 1 并使用 `Run Command` 的所有任务。

```

aws ssm describe-maintenance-window-tasks \
  --window-id "mw-ab12cd34eEXAMPLE" \
  --filters "Key=Priority,Values=1" "Key=TaskType,Values=RUN_COMMAND"

```

输出：

```

{
  "Tasks": [
    {
      "WindowId": "mw-ab12cd34eEXAMPLE",
      "WindowTaskId": "0d36e6b4-3a4f-411e-adcb-3558eEXAMPLE",
      "TaskArn": "AWS-RunPowerShellScript",
      "Type": "RUN_COMMAND",
      "Targets": [
        {
          "Key": "WindowTargetIds",
          "Values": [
            "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
          ]
        }
      ]
    }
  ]
}

```

```
    ],
    "TaskParameters": {},
    "Priority": 1,
    "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
    "MaxConcurrency": "1",
    "MaxErrors": "1",
    "Name": "MyRunCommandTask"
  }
]
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[“查看有关维护时段 \(AWS CLI\) 的信息”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeMaintenanceWindowTasks](#)中的。

## describe-maintenance-windows-for-target

以下代码示例演示了如何使用 describe-maintenance-windows-for-target。

### AWS CLI

列出与特定实例关联的所有维护时段

以下describe-maintenance-windows-for-target示例列出了目标或任务与指定实例关联的维护时段。

```
aws ssm describe-maintenance-windows-for-target \
  --targets Key=InstanceIds,Values=i-1234567890EXAMPLE \
  --resource-type INSTANCE
```

输出：

```
{
  "WindowIdentities": [
    {
      "WindowId": "mw-0c5ed765acEXAMPLE",
      "Name": "My-First-Maintenance-Window"
    }
  ]
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的“[查看有关维护窗口的信息 \(AWS CLI\)](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeMaintenanceWindowsForTarget](#)中的。

## describe-maintenance-windows

以下代码示例演示了如何使用 describe-maintenance-windows。

### AWS CLI

示例 1：列出所有维护时段

以下describe-maintenance-windows示例列出了当前区域中您 AWS 账户中的所有维护时段。

```
aws ssm describe-maintenance-windows
```

输出：

```
{
  "WindowIdentities": [
    {
      "WindowId": "mw-0ecb1226ddEXAMPLE",
      "Name": "MyMaintenanceWindow-1",
      "Enabled": true,
      "Duration": 2,
      "Cutoff": 1,
      "Schedule": "rate(180 minutes)",
      "NextExecutionTime": "2020-02-12T23:19:20.596Z"
    },
    {
      "WindowId": "mw-03eb9db428EXAMPLE",
      "Name": "MyMaintenanceWindow-2",
      "Enabled": true,
      "Duration": 3,
      "Cutoff": 1,
      "Schedule": "rate(7 days)",
      "NextExecutionTime": "2020-02-17T23:22:00.956Z"
    }
  ]
}
```

```
}
```

示例 2：列出所有已启用的维护时段

以下 `describe-maintenance-windows` 示例列出所有已启用的维护时段。

```
aws ssm describe-maintenance-windows \  
  --filters "Key=Enabled,Values=true"
```

示例 3：列出与特定名称匹配的维护时段

此 `describe-maintenance-windows` 示例列出具有指定名称的所有维护时段。

```
aws ssm describe-maintenance-windows \  
  --filters "Key=Name,Values=MyMaintenanceWindow"
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的“[查看有关维护窗口的信息 \(AWS CLI\)](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeMaintenanceWindows](#) 中的。

## describe-ops-items

以下代码示例演示了如何使用 `describe-ops-items`。

### AWS CLI

列出一组 OpsItems

以下 `describe-ops-items` 示例显示了您账户 OpsItems 中所有未结 AWS 账款项的列表。

```
aws ssm describe-ops-items \  
  --ops-item-filters "Key=Status,Values=Open,Operator=Equal"
```

输出：

```
{  
  "OpsItemSummaries": [  
    {  
      "CreatedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-CWE-Role/  
fbf77cbe264a33509569f23e4EXAMPLE",  
      "CreatedTime": "2020-03-14T17:02:46.375000-07:00",
```

```

    "LastModifiedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-CWE-
Role/fbf77cbe264a33509569f23e4EXAMPLE",
    "LastModifiedTime": "2020-03-14T17:02:46.375000-07:00",
    "Source": "SSM",
    "Status": "Open",
    "OpsItemId": "oi-7cfc5EXAMPLE",
    "Title": "SSM Maintenance Window execution failed",
    "OperationalData": {
      "/aws/dedup": {
        "Value": "{\"dedupString\": \"SSMOpsItems-SSM-maintenance-window-
execution-failed\"}",
        "Type": "SearchableString"
      },
      "/aws/resources": {
        "Value": "[{\"arn\": \"arn:aws:ssm:us-
east-2:111222333444:maintenancewindow/mw-034093d322EXAMPLE\"}]",
        "Type": "SearchableString"
      }
    },
    "Category": "Availability",
    "Severity": "3"
  },
  {
    "CreatedBy": "arn:aws:sts::1112223233444:assumed-role/OpsItem-CWE-Role/
fbf77cbe264a33509569f23e4EXAMPLE",
    "CreatedTime": "2020-02-26T11:43:15.426000-08:00",
    "LastModifiedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-CWE-
Role/fbf77cbe264a33509569f23e4EXAMPLE",
    "LastModifiedTime": "2020-02-26T11:43:15.426000-08:00",
    "Source": "EC2",
    "Status": "Open",
    "OpsItemId": "oi-6f966EXAMPLE",
    "Title": "EC2 instance stopped",
    "OperationalData": {
      "/aws/automations": {
        "Value": "[ { \"automationType\": \"AWS:SSM:Automation\",
\"automationId\": \"AWS-RestartEC2Instance\" } ]",
        "Type": "SearchableString"
      },
      "/aws/dedup": {
        "Value": "{\"dedupString\": \"SSMOpsItems-EC2-instance-stopped
\"}",
        "Type": "SearchableString"
      }
    },
  }

```



```

        "/aws/resources": {
            "Value": "[{\"arn\":\"arn:aws:ec2:us-east-2:111222333444:instance/i-0beccfbc02EXAMPLE\"}]",
            "Type": "SearchableString"
        }
    },
    "Category": "Availability",
    "Severity": "3"
}
]
}

```

有关更多信息，请参阅《S AWS systems Manager 用户指南》[OpsItems](#)中的“使用”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeOpsItems](#)中的。

## describe-parameters

以下代码示例演示了如何使用 describe-parameters。

### AWS CLI

#### 示例 1：列出所有参数

以下describe-parameters示例列出了当前 AWS 账户和区域中的所有参数。

```
aws ssm describe-parameters
```

输出：

```

{
  "Parameters": [
    {
      "Name": "MySecureStringParameter",
      "Type": "SecureString",
      "KeyId": "alias/aws/ssm",
      "LastModifiedDate": 1582155479.205,
      "LastModifiedUser": "arn:aws:sts::111222333444:assumed-role/Admin/Richard-Roe-Managed",
      "Description": "This is a SecureString parameter",
      "Version": 2,
      "Tier": "Advanced",
      "Policies": [

```

```

        {
            "PolicyText": "{\"Type\":\"Expiration\",\"Version\":\"1.0\",
\\\"Attributes\":{\"Timestamp\":\"2020-07-07T22:30:00Z\"}}",
            "PolicyType": "Expiration",
            "PolicyStatus": "Pending"
        },
        {
            "PolicyText": "{\"Type\":\"ExpirationNotification\",\"Version\":
\\\"1.0\",\\\"Attributes\":{\"Before\":\"12\",\\\"Unit\":\"Hours\"}}",
            "PolicyType": "ExpirationNotification",
            "PolicyStatus": "Pending"
        }
    ]
},
{
    "Name": "MyStringListParameter",
    "Type": "StringList",
    "LastModifiedDate": 1582154764.222,
    "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
    "Description": "This is a StringList parameter",
    "Version": 1,
    "Tier": "Standard",
    "Policies": []
},
{
    "Name": "MyStringParameter",
    "Type": "String",
    "LastModifiedDate": 1582154711.976,
    "LastModifiedUser": "arn:aws:iam::111222333444:user/Alejandro-Rosalez",
    "Description": "This is a String parameter",
    "Version": 1,
    "Tier": "Standard",
    "Policies": []
},
{
    "Name": "latestAmi",
    "Type": "String",
    "LastModifiedDate": 1580862415.521,
    "LastModifiedUser": "arn:aws:sts::111222333444:assumed-role/lambda-ssm-
role/Automation-UpdateSSM-Param",
    "Version": 3,
    "Tier": "Standard",
    "Policies": []
}

```

```
]
}
```

示例 2：列出与特定元数据匹配的所有参数

以下 `describe-parameters` 示例列出了与筛选器匹配的所有参数。

`aws ssm 描述参数`——过滤器“键=类型，值=” StringList

输出：

```
{
  "Parameters": [
    {
      "Name": "MyStringListParameter",
      "Type": "StringList",
      "LastModifiedDate": 1582154764.222,
      "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
      "Description": "This is a StringList parameter",
      "Version": 1,
      "Tier": "Standard",
      "Policies": []
    }
  ]
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[搜索 Systems Manager 参数](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeParameters](#) 中的。

## `describe-patch-baselines`

以下代码示例演示了如何使用 `describe-patch-baselines`。

AWS CLI

示例 1：列出所有补丁基准

以下 `describe-patch-baselines` 示例检索您账户中当前区域所有补丁基准的详细信息。

```
aws ssm describe-patch-baselines
```

输出：

```
{
  "BaselineIdentities": [
    {
      "BaselineName": "AWS-SuseDefaultPatchBaseline",
      "DefaultBaseline": true,
      "BaselineDescription": "Default Patch Baseline for Suse Provided by
AWS.",
      "BaselineId": "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-0123fdb36e334a3b2",
      "OperatingSystem": "SUSE"
    },
    {
      "BaselineName": "AWS-DefaultPatchBaseline",
      "DefaultBaseline": false,
      "BaselineDescription": "Default Patch Baseline Provided by AWS.",
      "BaselineId": "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-020d361a05defe4ed",
      "OperatingSystem": "WINDOWS"
    },
    ...
    {
      "BaselineName": "MyWindowsPatchBaseline",
      "DefaultBaseline": true,
      "BaselineDescription": "My patch baseline for EC2 instances for Windows
Server",
      "BaselineId": "pb-0ad00e0dd7EXAMPLE",
      "OperatingSystem": "WINDOWS"
    }
  ]
}
```

### 示例 2：列出提供的所有补丁基准 AWS

以下 describe-patch-baselines 示例列出了提供的所有补丁基准。AWS

```
aws ssm describe-patch-baselines \
  --filters "Key=OWNER,Values=[AWS]"
```

### 示例 3：列出您拥有的所有补丁基准

以下 describe-patch-baselines 示例列出当前区域在您的账户中创建的所有自定义补丁基准。

```
aws ssm describe-patch-baselines \  
  --filters "Key=OWNER,Values=[Self]"
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[关于预定义和自定义补丁基准](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribePatchBaselines](#) 中的。

## describe-patch-group-state

以下代码示例演示了如何使用 describe-patch-group-state。

### AWS CLI

获取补丁组的状态

以下 describe-patch-group-state 示例检索补丁组的高级补丁合规性摘要。

```
aws ssm describe-patch-group-state \  
  --patch-group "Production"
```

输出：

```
{  
  "Instances": 21,  
  "InstancesWithCriticalNonCompliantPatches": 1,  
  "InstancesWithFailedPatches": 2,  
  "InstancesWithInstalledOtherPatches": 3,  
  "InstancesWithInstalledPatches": 21,  
  "InstancesWithInstalledPendingRebootPatches": 2,  
  "InstancesWithInstalledRejectedPatches": 1,  
  "InstancesWithMissingPatches": 3,  
  "InstancesWithNotApplicablePatches": 4,  
  "InstancesWithOtherNonCompliantPatches": 1,  
  "InstancesWithSecurityNonCompliantPatches": 1,  
  "InstancesWithUnreportedNotApplicablePatches": 2  
}
```

有关更多信息，请参阅 Systems Manager AWS 用户指南中的关于补丁组 < <https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-patchgroups.html> > 和 [了解补丁合规性状态值](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribePatchGroupState](#)中的。

## describe-patch-groups

以下代码示例演示了如何使用 describe-patch-groups。

### AWS CLI

显示补丁组注册

以下 describe-patch-groups 示例列出补丁组注册。

```
aws ssm describe-patch-groups
```

输出：

```
{
  "Mappings": [
    {
      "PatchGroup": "Production",
      "BaselineIdentity": {
        "BaselineId": "pb-0123456789abcdef0",
        "BaselineName": "ProdPatching",
        "OperatingSystem": "WINDOWS",
        "BaselineDescription": "Patches for Production",
        "DefaultBaseline": false
      }
    },
    {
      "PatchGroup": "Development",
      "BaselineIdentity": {
        "BaselineId": "pb-0713acce01234567",
        "BaselineName": "DevPatching",
        "OperatingSystem": "WINDOWS",
        "BaselineDescription": "Patches for Development",
        "DefaultBaseline": true
      }
    },
    ...
  ]
}
```

有关更多信息，请参阅 Systems Manager AWS 用户指南中的创建补丁组 < <https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-group-tagging.html> > 和 [向补丁基准添加补丁组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribePatchGroups](#) 中的。

## describe-patch-properties

以下代码示例演示了如何使用 describe-patch-properties。

### AWS CLI

列出亚马逊 Linux 补丁的可用性

以下 describe-patch-properties 示例显示了您的 AWS 账户中已提供补丁的 Amazon Linux 产品列表。

```
aws ssm describe-patch-properties \
  --operating-system AMAZON_LINUX \
  --property PRODUCT
```

输出：

```
{
  "Properties": [
    {
      "Name": "AmazonLinux2012.03"
    },
    {
      "Name": "AmazonLinux2012.09"
    },
    {
      "Name": "AmazonLinux2013.03"
    },
    {
      "Name": "AmazonLinux2013.09"
    },
    {
      "Name": "AmazonLinux2014.03"
    },
    {
      "Name": "AmazonLinux2014.09"
    }
  ]
}
```

```
    },  
    {  
      "Name": "AmazonLinux2015.03"  
    },  
    {  
      "Name": "AmazonLinux2015.09"  
    },  
    {  
      "Name": "AmazonLinux2016.03"  
    },  
    {  
      "Name": "AmazonLinux2016.09"  
    },  
    {  
      "Name": "AmazonLinux2017.03"  
    },  
    {  
      "Name": "AmazonLinux2017.09"  
    },  
    {  
      "Name": "AmazonLinux2018.03"  
    }  
  ]  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[关于补丁基准](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribePatchProperties](#) 中的。

## describe-sessions

以下代码示例演示了如何使用 `describe-sessions`。

### AWS CLI

示例 1：列出所有活动的会话管理器会话

此 `describe-sessions` 示例检索指定用户在过去 30 天内最近创建的活动会话（包括已连接和已断开连接的会话）的列表。此命令仅返回使用会话管理器启动的与目标的连接的结果。它不列出通过其他方式（例如远程桌面连接或 SSH）建立的连接。

```
aws ssm describe-sessions \  
  --state "Active" \  
  --max-items 100
```



```
--filters "key=Owner,value=arn:aws:sts::123456789012:assumed-role/Administrator/Shirley-Rodriguez"
```

输出：

```
{
  "Sessions": [
    {
      "SessionId": "John-07a16060613c408b5",
      "Target": "i-1234567890abcdef0",
      "Status": "Connected",
      "StartDate": 1550676938.352,
      "Owner": "arn:aws:sts::123456789012:assumed-role/Administrator/Shirley-Rodriguez",
      "OutputUrl": {}
    },
    {
      "SessionId": "John-01edf534b8b56e8eb",
      "Target": "i-9876543210abcdef0",
      "Status": "Connected",
      "StartDate": 1550676842.194,
      "Owner": "arn:aws:sts::123456789012:assumed-role/Administrator/Shirley-Rodriguez",
      "OutputUrl": {}
    }
  ]
}
```

示例 2：列出所有已终止的会话管理器会话

此describe-sessions示例检索过去 30 天内所有用户最近终止的会话列表。

```
aws ssm describe-sessions \
  --state "History"
```

输出：

```
{
  "Sessions": [
    {
      "SessionId": "Mary-Major-0022b1eb2b0d9e3bd",
      "Target": "i-1234567890abcdef0",
```

```

        "Status": "Terminated",
        "StartDate": 1550520701.256,
        "EndDate": 1550521931.563,
        "Owner": "arn:aws:sts::123456789012:assumed-role/Administrator/Mary-
Major"
      },
      {
        "SessionId": "Jane-Roe-0db53f487931ed9d4",
        "Target": "i-9876543210abcdef0",
        "Status": "Terminated",
        "StartDate": 1550161369.149,
        "EndDate": 1550162580.329,
        "Owner": "arn:aws:sts::123456789012:assumed-role/Administrator/Jane-Roe"
      },
      ...
    ],
    "NextToken": "--token string truncated--"
  }

```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的“[查看会话历史记录](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSessions](#)中的。

## disassociate-ops-item-related-item

以下代码示例演示了如何使用 disassociate-ops-item-related-item。

### AWS CLI

#### 删除相关项目关联

以下disassociate-ops-item-related-item示例删除了与相关项目之间的关联。 OpsItem

```

aws ssm disassociate-ops-item-related-item \
  --ops-item-id "oi-f99f2EXAMPLE" \
  --association-id "e2036148-cccb-490e-ac2a-390e5EXAMPLE"

```

此命令不生成任何输出。

有关更多信息，请参阅《S AWS system s Manager 用户指南》 [OpsCenter](#)中的“[处理事件管理器事件](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateOpsItemRelatedItem](#)中的。

## get-automation-execution

以下代码示例演示了如何使用 `get-automation-execution`。

### AWS CLI

显示有关自动化执行的详细信息

以下 `get-automation-execution` 示例显示有关自动化执行的详细信息。

```
aws ssm get-automation-execution \  
  --automation-execution-id 73c8eef8-f4ee-4a05-820c-e354fEXAMPLE
```

输出：

```
{  
  "AutomationExecution": {  
    "AutomationExecutionId": "73c8eef8-f4ee-4a05-820c-e354fEXAMPLE",  
    "DocumentName": "AWS-StartEC2Instance",  
    "DocumentVersion": "1",  
    "ExecutionStartTime": 1583737233.748,  
    "ExecutionEndTime": 1583737234.719,  
    "AutomationExecutionStatus": "Success",  
    "StepExecutions": [  
      {  
        "StepName": "startInstances",  
        "Action": "aws:changeInstanceState",  
        "ExecutionStartTime": 1583737234.134,  
        "ExecutionEndTime": 1583737234.672,  
        "StepStatus": "Success",  
        "Inputs": {  
          "DesiredState": "\"running\"",  
          "InstanceIds": "[\"i-0cb99161f6EXAMPLE\"]"  
        },  
        "Outputs": {  
          "InstanceStates": [  
            "running"  
          ]  
        },  
        "StepExecutionId": "95e70479-cf20-4d80-8018-7e4e2EXAMPLE",  
        "OverriddenParameters": {}  
      }  
    ],  
  },  
}
```

```
    "StepExecutionsTruncated": false,
    "Parameters": {
      "AutomationAssumeRole": [
        ""
      ],
      "InstanceId": [
        "i-0cb99161f6EXAMPLE"
      ]
    },
    "Outputs": {},
    "Mode": "Auto",
    "ExecutedBy": "arn:aws:sts::29884EXAMPLE:assumed-role/mw_service_role/OrchestrationService",
    "Targets": [],
    "ResolvedTargets": {
      "ParameterValues": [],
      "Truncated": false
    }
  }
}
```

有关更多信息，请参阅 S AWS systems Manager 用户指南中的[演练：修补 Linux AM AWS I \(CLI\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetAutomationExecution](#)中的。

## get-calendar-state

以下代码示例演示了如何使用 get-calendar-state。

### AWS CLI

示例 1：获取更改日历的当前状态

此 get-calendar-state 示例返回日历在当前时间的状态。由于该示例未指定时间，因此会报告日历的当前状态。

```
aws ssm get-calendar-state \
  --calendar-names "MyCalendar"
```

输出：

```
{
```

```
"State": "OPEN",
"AtTime": "2020-02-19T22:28:51Z",
"NextTransitionTime": "2020-02-24T21:15:19Z"
}
```

示例 2：获取更改日历在指定时间的状态

此 `get-calendar-state` 示例返回日历在指定时间的状态。

```
aws ssm get-calendar-state \
  --calendar-names "MyCalendar" \
  --at-time "2020-07-19T21:15:19Z"
```

输出：

```
{
  "State": "CLOSED",
  "AtTime": "2020-07-19T21:15:19Z"
}
```

有关更多信息，请参阅 [《S AWS systems Manager 用户指南》](#) 中的“获取变更日历的状态”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetCalendarState](#) 中的。

## get-command-invocation

以下代码示例演示了如何使用 `get-command-invocation`。

### AWS CLI

显示命令调用的详细信息

以下 `get-command-invocation` 示例列出对指定实例上指定命令的所有调用。

```
aws ssm get-command-invocation \
  --command-id "ef7fd8-9b57-4151-a15c-db9a12345678" \
  --instance-id "i-1234567890abcdef0"
```

输出：

```
{
  "CommandId": "ef7fd8-9b57-4151-a15c-db9a12345678",
```

```
"InstanceId": "i-1234567890abcdef0",
"Comment": "b48291dd-ba76-43e0-b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-
d6ce8EXAMPLE",
"DocumentName": "AWS-UpdateSSMAgent",
"DocumentVersion": "",
"PluginName": "aws:updateSsmAgent",
"ResponseCode": 0,
"ExecutionStartDateTime": "2020-02-19T18:18:03.419Z",
"ExecutionElapsedTime": "PT0.091S",
"ExecutionEndDateTime": "2020-02-19T18:18:03.419Z",
"Status": "Success",
"StatusDetails": "Success",
"StandardOutputContent": "Updating amazon-ssm-agent from 2.3.842.0 to latest
\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/
ssm-agent-manifest.json\namazon-ssm-agent 2.3.842.0 has already been installed,
update skipped\n",
"StandardOutputUrl": "",
"StandardErrorContent": "",
"StandardErrorUrl": "",
"CloudWatchOutputConfig": {
  "CloudWatchLogGroupName": "",
  "CloudWatchOutputEnabled": false
}
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[了解命令状态](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetCommandInvocation](#)中的。

## get-connection-status

以下代码示例演示了如何使用 get-connection-status。

### AWS CLI

显示托管式实例的连接状态

此 get-connection-status 示例返回指定托管式实例的连接状态。

```
aws ssm get-connection-status \
  --target i-1234567890abcdef0
```

输出：

```
{
  "Target": "i-1234567890abcdef0",
  "Status": "connected"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetConnectionStatus](#)中的。

## get-default-patch-baseline

以下代码示例演示了如何使用 `get-default-patch-baseline`。

### AWS CLI

示例 1：显示默认 Windows 补丁基准

以下 `get-default-patch-baseline` 示例检索 Windows Server 默认补丁基准的详细信息。

```
aws ssm get-default-patch-baseline
```

输出：

```
{
  "BaselineId": "pb-0713accee01612345",
  "OperatingSystem": "WINDOWS"
}
```

示例 2：显示 Amazon Linux 的默认补丁基准

以下 `get-default-patch-baseline` 示例检索 Amazon Linux 默认补丁基准的详细信息。

```
aws ssm get-default-patch-baseline \
  --operating-system AMAZON_LINUX
```

输出：

```
{
  "BaselineId": "pb-047c6eb9c8fc12345",
  "OperatingSystem": "AMAZON_LINUX"
}
```

有关更多信息，请参阅《Systems Manager 用户指南》中的“关于预定义和自定义补丁基准 < <https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-baselines.html> >”和“[将现有补丁基准设置为默认补丁基准](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDefaultPatchBaseline](#) 中的。

## get-deployable-patch-snapshot-for-instance

以下代码示例演示了如何使用 `get-deployable-patch-snapshot-for-instance`。

### AWS CLI

检索实例使用的补丁基准的当前快照

以下 `get-deployable-patch-snapshot-for-instance` 示例检索实例使用的指定补丁基准当前快照的详细信息。此命令必须使用实例凭证从实例运行。为确保其使用实例凭证，请运行 `aws configure` 并仅指定您的实例的区域。将 Access Key 和 Secret Key 字段留空。

提示：使用 `uuidgen` 生成 `snapshot-id`。

```
aws ssm get-deployable-patch-snapshot-for-instance \
  --instance-id "i-1234567890abcdef0" \
  --snapshot-id "521c3536-930c-4aa9-950e-01234567abcd"
```

输出：

```
{
  "InstanceId": "i-1234567890abcdef0",
  "SnapshotId": "521c3536-930c-4aa9-950e-01234567abcd",
  "Product": "AmazonLinux2018.03",
  "SnapshotDownloadUrl": "https://patch-baseline-snapshot-us-east-1.s3.amazonaws.com/ed85194ef27214f5984f28b4d664d14f7313568fea7d4b6ac6c10ad1f729d7e7-773304212436/AMAZON_LINUX-521c3536-930c-4aa9-950e-01234567abcd?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20190215T164031Z&X-Amz-SignedHeaders=host&X-Amz-Expires=86400&X-Amz-Credential=AKIAJ5C56P35AEBRX2QQ%2F20190215%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=efaaaf6e3878e77f48a6697e015efdbda9c426b09c5822055075c062f6ad2149"
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[参数名称：快照 ID](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDeployablePatchSnapshotForInstance](#) 中的。



## get-document

以下代码示例演示了如何使用 get-document。

### AWS CLI

#### 获取文档内容

以下 get-document 示例显示 Systems Manager 文档的内容。

```
aws ssm get-document \
  --name "AWS-RunShellScript"
```

输出：

```
{
  "Name": "AWS-RunShellScript",
  "DocumentVersion": "1",
  "Status": "Active",
  "Content": "{\n  \"schemaVersion\": \"1.2\",\n  \"description\": \"Run a\nshell script or specify the commands to run.\",\n  \"parameters\": {\n    \"commands\": {\n      \"type\": \"StringList\",\n      \"description\n\": \"(Required) Specify a shell script or a command to run.\",\n      \"minItems\": 1,\n      \"displayType\": \"textarea\"\n    },\n    \"workingDirectory\": {\n      \"type\": \"String\",\n      \"default\n\": \"\",\n      \"description\": \"(Optional) The path to the working\ndirectory on your instance.\",\n      \"maxChars\": 4096\n    },\n    \"executionTimeout\": {\n      \"type\": \"String\",\n      \"default\n\": \"3600\",\n      \"description\": \"(Optional) The time in seconds for a\ncommand to complete before it is considered to have failed. Default is 3600 (1\nhour). Maximum is 172800 (48 hours).\",\n      \"allowedPattern\": \"([1-9]\n[0-9]{0,4})|(1[0-6][0-9]{4})|(17[0-1][0-9]{3})|(172[0-7][0-9]{2})|(172800)\"\n    }\n  },\n  \"runtimeConfig\": {\n    \"aws:runShellScript\": {\n      \"properties\": [\n        {\n          \"id\":\n\"0.aws:runShellScript\",\n          \"runCommand\": \"{{ commands }}\",\n          \"workingDirectory\": \"{{ workingDirectory }}\",\n          \"timeoutSeconds\": \"{{ executionTimeout }}\"\n        }\n      ]\n    }\n  }\n},\n  \"DocumentType\": \"Command\",\n  \"DocumentFormat\": \"JSON\"\n}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的 [AWS Systems Manager 文档](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDocument](#)中的。

## get-inventory-schema

以下代码示例演示了如何使用 get-inventory-schema。

### AWS CLI

查看您的清单架构

此示例返回账户清单类型名称列表。

命令:

```
aws ssm get-inventory-schema
```

输出:

```
{
  "Schemas": [
    {
      "TypeName": "AWS:AWSComponent",
      "Version": "1.0",
      "Attributes": [
        {
          "Name": "Name",
          "DataType": "STRING"
        },
        {
          "Name": "ApplicationType",
          "DataType": "STRING"
        },
        {
          "Name": "Publisher",
          "DataType": "STRING"
        },
        {
          "Name": "Version",
          "DataType": "STRING"
        },
        {
          "Name": "InstalledTime",
          "DataType": "STRING"
        }
      ]
    }
  ]
}
```

```
    },
    {
      "Name": "Architecture",
      "DataType": "STRING"
    },
    {
      "Name": "URL",
      "DataType": "STRING"
    }
  ]
},
...
],
"NextToken": "--token string truncated--"
}
```

查看特定清单类型的清单架构

此示例返回 AWS:AWS 组件清单类型的清单架构。

命令:

```
aws ssm get-inventory-schema --type-name "AWS:AWSComponent"
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetInventorySchema](#) 中的。

## get-inventory

以下代码示例演示了如何使用 get-inventory。

AWS CLI

查看您的清单

此示例获取清单的自定义元数据。

命令:

```
aws ssm get-inventory
```

输出:

```
{
```

```

"Entities": [
  {
    "Data": {
      "AWS:InstanceInformation": {
        "Content": [
          {
            "ComputerName": "ip-172-31-44-222.us-
west-2.compute.internal",
            "InstanceId": "i-0cb2b964d3e14fd9f",
            "IpAddress": "172.31.44.222",
            "AgentType": "amazon-ssm-agent",
            "ResourceType": "EC2Instance",
            "AgentVersion": "2.0.672.0",
            "PlatformVersion": "2016.09",
            "PlatformName": "Amazon Linux AMI",
            "PlatformType": "Linux"
          }
        ],
        "TypeName": "AWS:InstanceInformation",
        "SchemaVersion": "1.0",
        "CaptureTime": "2017-02-20T18:03:58Z"
      }
    },
    "Id": "i-0cb2b964d3e14fd9f"
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetInventory](#)中的。

## get-maintenance-window-execution-task-invocation

以下代码示例演示了如何使用 `get-maintenance-window-execution-task-invocation`。

### AWS CLI

获取有关维护时段任务调用的信息

以下`get-maintenance-window-execution-task-invocation`示例列出了有关指定任务调用的信息，该任务调用是指定维护时段执行的一部分。

```

aws ssm get-maintenance-window-execution-task-invocation \
  --window-execution-id "bc494bfa-e63b-49f6-8ad1-aa9f2EXAMPLE" \

```

```
--task-id "96f2ad59-97e3-461d-a63d-40c8aEXAMPLE" \  
--invocation-id "a5273e2c-d2c6-4880-b3e1-5e550EXAMPLE"
```

输出：

```
{  
  "Status": "SUCCESS",  
  "Parameters": "{\"comment\":\"\", \"documentName\":\"AWS-RunPowerShellScript\",  
  \"instanceIds\": [\"i-1234567890EXAMPLE\"], \"maxConcurrency\": \"1\", \"maxErrors\":  
  \"1\", \"parameters\": {\"executionTimeout\": [\"3600\"], \"workingDirectory\": [\"\"],  
  \"commands\": [\"echo Hello\"]}, \"timeoutSeconds\": 600}",  
  "ExecutionId": "03b6baa0-5460-4e15-83f2-ea685EXAMPLE",  
  "InvocationId": "a5273e2c-d2c6-4880-b3e1-5e550EXAMPLE",  
  "StartTime": 1549998326.421,  
  "TaskType": "RUN_COMMAND",  
  "EndTime": 1550001931.784,  
  "WindowExecutionId": "bc494bfa-e63b-49f6-8ad1-aa9f2EXAMPLE",  
  "StatusDetails": "Failed",  
  "TaskExecutionId": "96f2ad59-97e3-461d-a63d-40c8aEXAMPLE"  
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[“查看有关任务和任务执行的信息 \(AWS CLI\)”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMaintenanceWindowExecutionTaskInvocation](#)中的。

## get-maintenance-window-execution-task

以下代码示例演示了如何使用 get-maintenance-window-execution-task。

### AWS CLI

获取有关维护时段任务执行的信息

以下 get-maintenance-window-execution-task 示例列出有关作为指定维护时段执行组成部分的任务的信息。

```
aws ssm get-maintenance-window-execution-task \  
  --window-execution-id "518d5565-5969-4cca-8f0e-da3b2EXAMPLE" \  
  --task-id "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE"
```

输出：

```
{
  "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2EXAMPLE",
  "TaskExecutionId": "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE",
  "TaskArn": "AWS-RunPatchBaseline",
  "ServiceRole": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
  "Type": "RUN_COMMAND",
  "TaskParameters": [
    {
      "BaselineOverride": {
        "Values": [
          ""
        ]
      },
      "InstallOverrideList": {
        "Values": [
          ""
        ]
      },
      "Operation": {
        "Values": [
          "Scan"
        ]
      },
      "RebootOption": {
        "Values": [
          "RebootIfNeeded"
        ]
      },
      "SnapshotId": {
        "Values": [
          "{{ aws:ORCHESTRATION_ID }}"
        ]
      },
      "aws:InstanceId": {
        "Values": [
          "i-02573cafcfEXAMPLE",
          "i-0471e04240EXAMPLE",
          "i-07782c72faEXAMPLE"
        ]
      }
    }
  ]
}
```

```
],  
  "Priority": 1,  
  "MaxConcurrency": "1",  
  "MaxErrors": "3",  
  "Status": "SUCCESS",  
  "StartTime": "2021-08-04T11:45:35.088000-07:00",  
  "EndTime": "2021-08-04T11:53:09.079000-07:00"  
}
```

有关更多信息，请参阅 [S AWS systems Manager 用户指南中的查看有关任务和任务执行 \(AWS CLI\)](#) 的信息。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 `GetMaintenanceWindowExecutionTask`](#) 中的。

## get-maintenance-window-execution

以下代码示例演示了如何使用 `get-maintenance-window-execution`。

### AWS CLI

获取有关维护时段任务执行的信息

以下 `get-maintenance-window-execution` 示例列出有关指定维护时段执行组成部分来执行的任务的信息。

```
aws ssm get-maintenance-window-execution \  
  --window-execution-id "518d5565-5969-4cca-8f0e-da3b2EXAMPLE"
```

输出：

```
{  
  "Status": "SUCCESS",  
  "TaskIds": [  
    "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE"  
  ],  
  "StartTime": 1487692834.595,  
  "EndTime": 1487692835.051,  
  "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2EXAMPLE",  
}
```

有关更多信息，请参阅《[S AWS systems Manager 用户指南](#)》中的“[查看有关任务和任务执行的信息 \(AWS CLI\)](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMaintenanceWindowExecution](#)中的。

## get-maintenance-window-task

以下代码示例演示了如何使用 `get-maintenance-window-task`。

### AWS CLI

获取有关维护时段任务的信息

以下`get-maintenance-window-task`示例检索有关指定维护时段任务的详细信息。

```
aws ssm get-maintenance-window-task \  
  --window-id mw-0c5ed765acEXAMPLE \  
  --window-task-id 0e842a8d-2d44-4886-bb62-af8dcEXAMPLE
```

输出：

```
{  
  "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/  
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",  
  "MaxErrors": "1",  
  "TaskArn": "AWS-RunPowerShellScript",  
  "MaxConcurrency": "1",  
  "WindowTaskId": "0e842a8d-2d44-4886-bb62-af8dcEXAMPLE",  
  "TaskParameters": {},  
  "Priority": 1,  
  "TaskInvocationParameters": {  
    "RunCommand": {  
      "Comment": "",  
      "TimeoutSeconds": 600,  
      "Parameters": {  
        "commands": [  
          "echo Hello"  
        ],  
        "executionTimeout": [  
          "3600"  
        ],  
        "workingDirectory": [  
          ""  
        ]  
      }  
    }  
  }  
}
```



```
    }
  },
  "WindowId": "mw-0c5ed765acEXAMPLE",
  "TaskType": "RUN_COMMAND",
  "Targets": [
    {
      "Values": [
        "84c818da-b619-4d3d-9651-946f3EXAMPLE"
      ],
      "Key": "WindowTargetIds"
    }
  ],
  "Name": "ExampleTask"
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的“[查看有关维护窗口的信息 \(AWS CLI\)](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMaintenanceWindowTask](#)中的。

## get-maintenance-window

以下代码示例演示了如何使用 get-maintenance-window。

### AWS CLI

查看有关维护时段的信息

以下 get-maintenance-window 示例将检索指定维护时段的详细信息。

```
aws ssm get-maintenance-window \
  --window-id "mw-03eb9db428EXAMPLE"
```

输出：

```
{
  "AllowUnassociatedTargets": true,
  "CreateDate": 1515006912.957,
  "Cutoff": 1,
  "Duration": 6,
  "Enabled": true,
  "ModifiedDate": 2020-01-01T10:04:04.099Z,
```

```
"Name": "My-Maintenance-Window",
"Schedule": "rate(3 days)",
"WindowId": "mw-03eb9db428EXAMPLE",
"NextExecutionTime": "2020-02-25T00:08:15.099Z"
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的“[查看有关维护时段 \(AWS CLI\) 的信息](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMaintenanceWindow](#)中的。

## get-ops-item

以下代码示例演示了如何使用 get-ops-item。

### AWS CLI

查看有关某人的信息 OpsItem

以下get-ops-item示例显示了有关指定项的详细信息 OpsItem。

```
aws ssm get-ops-item \
  --ops-item-id oi-0b725EXAMPLE
```

输出：

```
{
  "OpsItem": {
    "CreatedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-CWE-Role/fbf77cbe264a33509569f23e4EXAMPLE",
    "CreatedTime": "2019-12-04T15:52:16.793000-08:00",
    "Description": "CloudWatch Event Rule SSMOpsItems-EC2-instance-terminated was triggered. Your EC2 instance has terminated. See below for more details.",
    "LastModifiedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-CWE-Role/fbf77cbe264a33509569f23e4EXAMPLE",
    "LastModifiedTime": "2019-12-04T15:52:16.793000-08:00",
    "Notifications": [],
    "RelatedOpsItems": [],
    "Status": "Open",
    "OpsItemId": "oi-0b725EXAMPLE",
    "Title": "EC2 instance terminated",
    "Source": "EC2",
```

```

    "OperationalData": {
      "/aws/automations": {
        "Value": "[ { \"automationType\": \"AWS:SSM:Automation\",
        \"automationId\": \"AWS-CreateManagedWindowsInstance\" }, { \"automationType\":
        \"AWS:SSM:Automation\", \"automationId\": \"AWS-CreateManagedLinuxInstance\" } ]",
        "Type": "SearchableString"
      },
      "/aws/dedup": {
        "Value": "{ \"dedupString\": \"SSMOpsItems-EC2-instance-terminated
        \"}",
        "Type": "SearchableString"
      },
      "/aws/resources": {
        "Value": "[{ \"arn\": \"arn:aws:ec2:us-east-2:111222333444:instance/
        i-05adec7e97EXAMPLE\"}]\"",
        "Type": "SearchableString"
      },
      "event-time": {
        "Value": "2019-12-04T23:52:16Z",
        "Type": "String"
      },
      "instance-state": {
        "Value": "terminated",
        "Type": "String"
      }
    },
    "Category": "Availability",
    "Severity": "4"
  }
}

```

有关更多信息，请参阅《S AWS systems Manager 用户指南》[OpsItems](#)中的“使用”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetOpsItem](#)中的。

## get-ops-summary

以下代码示例演示了如何使用 get-ops-summary。

### AWS CLI

查看所有内容的摘要 OpsItems

以下get-ops-summary示例显示了您 AWS 账户 OpsItems 中所有内容的摘要。

```
aws ssm get-ops-summary
```

输出：

```
{
  "Entities": [
    {
      "Id": "oi-4309fEXAMPLE",
      "Data": {
        "AWS:OpsItem": {
          "CaptureTime": "2020-02-26T18:58:32.918Z",
          "Content": [
            {
              "AccountId": "111222333444",
              "Category": "Availability",
              "CreatedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-CWE-Role/fbf77cbe264a33509569f23e4EXAMPLE",
              "CreatedTime": "2020-02-26T19:10:44.149Z",
              "Description": "CloudWatch Event Rule SSM0psItems-EC2-instance-terminated was triggered. Your EC2 instance has terminated. See below for more details.",
              "LastModifiedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-CWE-Role/fbf77cbe264a33509569f23e4EXAMPLE",
              "LastModifiedTime": "2020-02-26T19:10:44.149Z",
              "Notifications": "",
              "OperationalData": "{\"/aws/automations\": {\"type\": \"SearchableString\", \"value\": \"[ { \\\"automationType\\\": \\\"AWS:SSM:Automation\\\", \\\"automationId\\\": \\\"AWS-CreateManagedWindowsInstance\\\" }, { \\\"automationType\\\": \\\"AWS:SSM:Automation\\\", \\\"automationId\\\": \\\"AWS-CreateManagedLinuxInstance\\\" } ]\", \"/aws/resources\": {\"type\": \"SearchableString\", \"value\": \"[{\\\"arn\\\": \\\"arn:aws:ec2:us-east-2:111222333444:instance/i-0acbd0800fEXAMPLE\\\"]\", \"/aws/dedup\": {\"type\": \"SearchableString\", \"value\": \"{\\\"dedupString\\\": \\\"SSM0psItems-EC2-instance-terminated\\\"}\"}}",
              "OpsItemId": "oi-4309fEXAMPLE",
              "RelatedItems": "",
              "Severity": "3",
              "Source": "EC2",
              "Status": "Open",
              "Title": "EC2 instance terminated"
            }
          ]
        }
      }
    }
  ]
}
```

```

    }
  },
  {
    "Id": "oi-bb2a0e6a4541",
    "Data": {
      "AWS:OpsItem": {
        "CaptureTime": "2019-11-26T19:20:06.161Z",
        "Content": [
          {
            "AccountId": "111222333444",
            "Category": "Availability",
            "CreatedBy": "arn:aws:sts::111222333444:assumed-role/
OpsItem-CWE-Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "CreatedTime": "2019-11-26T20:00:07.237Z",
            "Description": "CloudWatch Event Rule SSMOpsItems-SSM-
maintenance-window-execution-failed was triggered. Your SSM Maintenance Window
execution has failed. See below for more details.",
            "LastModifiedBy": "arn:aws:sts::111222333444:assumed-
role/OpsItem-CWE-Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "LastModifiedTime": "2019-11-26T20:00:07.237Z",
            "Notifications": "",
            "OperationalData": "{\"/aws/resources\":{\"type
\": \"SearchableString\", \"value\": \"[\\\"arn\\\": \\\"arn:aws:ssm:us-
east-2:111222333444:maintenancewindow/mw-0e83ba440dEXAMPLE\\\"]\"}, \"/aws/dedup\":
{\"type\": \"SearchableString\", \"value\": \"[\\\"dedupString\\\": \\\"SSMOpsItems-SSM-
maintenance-window-execution-failed\\\"]\"}}",
            "OpsItemId": "oi-bb2a0EXAMPLE",
            "RelatedItems": "",
            "Severity": "3",
            "Source": "SSM",
            "Status": "Open",
            "Title": "SSM Maintenance Window execution failed"
          }
        ]
      }
    }
  }
]
}

```

有关更多信息，请参阅《S AWS systems Manager 用户指南》[OpsItems中的“使用”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetOpsSummary](#)中的。

## get-parameter-history

以下代码示例演示了如何使用 `get-parameter-history`。

### AWS CLI

获取参数的值历史记录

以下 `get-parameter-history` 示例列出指定参数的更改历史记录，包括其值。

```
aws ssm get-parameter-history \  
  --name "MyStringParameter"
```

输出：

```
{  
  "Parameters": [  
    {  
      "Name": "MyStringParameter",  
      "Type": "String",  
      "LastModifiedDate": 1582154711.976,  
      "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",  
      "Description": "This is the first version of my String parameter",  
      "Value": "Veni",  
      "Version": 1,  
      "Labels": [],  
      "Tier": "Standard",  
      "Policies": []  
    },  
    {  
      "Name": "MyStringParameter",  
      "Type": "String",  
      "LastModifiedDate": 1582156093.471,  
      "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",  
      "Description": "This is the second version of my String parameter",  
      "Value": "Vidi",  
      "Version": 2,  
      "Labels": [],  
      "Tier": "Standard",  
      "Policies": []  
    },  
    {  
      "Name": "MyStringParameter",
```

```
        "Type": "String",
        "LastModifiedDate": 1582156117.545,
        "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
        "Description": "This is the third version of my String parameter",
        "Value": "Vici",
        "Version": 3,
        "Labels": [],
        "Tier": "Standard",
        "Policies": []
    }
]
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用参数版本](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetParameterHistory](#)中的。

## get-parameter

以下代码示例演示了如何使用 get-parameter。

### AWS CLI

#### 示例 1：显示参数的值

以下 get-parameter 示例列出了指定单个参数的值。

```
aws ssm get-parameter \
  --name "MyStringParameter"
```

输出：

```
{
  "Parameter": {
    "Name": "MyStringParameter",
    "Type": "String",
    "Value": "Veni",
    "Version": 1,
    "LastModifiedDate": 1530018761.888,
    "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/MyStringParameter"
    "DataType": "text"
  }
}
```

```
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Parameter Store](#)。

### 示例 2：解密参数的值 SecureString

以下 `get-parameter` 示例解密指定参数 `SecureString` 的值。

```
aws ssm get-parameter \  
  --name "MySecureStringParameter" \  
  --with-decryption
```

输出：

```
{  
  "Parameter": {  
    "Name": "MySecureStringParameter",  
    "Type": "SecureString",  
    "Value": "16679b88-310b-4895-a943-e0764EXAMPLE",  
    "Version": 2,  
    "LastModifiedDate": 1582155479.205,  
    "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/  
MySecureStringParameter"  
    "DataType": "text"  
  }  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Parameter Store](#)。

### 示例 3：使用标签显示参数的值

以下 `get-parameter` 示例列出具有指定标签的指定单个参数的值。

```
aws ssm get-parameter \  
  --name "MyParameter:label"
```

输出：

```
{  
  "Parameter": {
```



```
    "Name": "MyParameter",
    "Type": "String",
    "Value": "parameter version 2",
    "Version": 2,
    "Selector": ":label",
    "LastModifiedDate": "2021-07-12T09:49:15.865000-07:00",
    "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/MyParameter",
    "DataType": "text"
  }
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用参数标签](#)。

示例 4：使用版本显示参数的值

以下`get-parameter`示例列出了指定单参数版本的值。

```
aws ssm get-parameter \
  --name "MyParameter:2"
```

输出：

```
{
  "Parameter": {
    "Name": "MyParameter",
    "Type": "String",
    "Value": "parameter version 2",
    "Version": 2,
    "Selector": ":2",
    "LastModifiedDate": "2021-07-12T09:49:15.865000-07:00",
    "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/MyParameter",
    "DataType": "text"
  }
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用参数标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetParameter](#)中的。

## get-parameters-by-path

以下代码示例演示了如何使用 `get-parameters-by-path`。

## AWS CLI

列出特定路径中的参数

以下`get-parameters-by-path`示例列出了指定层次结构中的参数。

```
aws ssm get-parameters-by-path \  
  --path "/site/newyork/department/"
```

输出：

```
{  
  "Parameters": [  
    {  
      "Name": "/site/newyork/department/marketing",  
      "Type": "String",  
      "Value": "Floor 2",  
      "Version": 1,  
      "LastModifiedDate": 1530018761.888,  
      "ARN": "arn:aws:ssm:us-east-1:111222333444:parameter/site/newyork/  
department/marketing"  
    },  
    {  
      "Name": "/site/newyork/department/infotech",  
      "Type": "String",  
      "Value": "Floor 3",  
      "Version": 1,  
      "LastModifiedDate": 1530018823.429,  
      "ARN": "arn:aws:ssm:us-east-1:111222333444:parameter/site/newyork/  
department/infotech"  
    },  
    ...  
  ]  
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[使用参数层次结构](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetParametersByPath](#)中的。

### `get-parameters`

以下代码示例演示了如何使用 `get-parameters`。

## AWS CLI

### 示例 1：列出参数的值

以下 `get-parameters` 示例列出三个指定参数的值。

```
aws ssm get-parameters \  
  --names "MyStringParameter" "MyStringListParameter" "MyInvalidParameterName"
```

输出：

```
{  
  "Parameters": [  
    {  
      "Name": "MyStringListParameter",  
      "Type": "StringList",  
      "Value": "alpha,beta,gamma",  
      "Version": 1,  
      "LastModifiedDate": 1582154764.222,  
      "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/  
MyStringListParameter"  
      "DataType": "text"  
    },  
    {  
      "Name": "MyStringParameter",  
      "Type": "String",  
      "Value": "Vici",  
      "Version": 3,  
      "LastModifiedDate": 1582156117.545,  
      "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/MyStringParameter"  
      "DataType": "text"  
    }  
  ],  
  "InvalidParameters": [  
    "MyInvalidParameterName"  
  ]  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Parameter Store](#)。

### 示例 2：使用 ``--query`` 选项列出多个参数的名称和值

以下 `get-parameters` 示例列出指定参数的名称和值。

```
aws ssm get-parameters \  
  --names MyStringParameter MyStringListParameter \  
  --query "Parameters[*].{Name:Name,Value:Value}"
```

输出：

```
[  
  {  
    "Name": "MyStringListParameter",  
    "Value": "alpha,beta,gamma"  
  },  
  {  
    "Name": "MyStringParameter",  
    "Value": "Vidi"  
  }  
]
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Parameter Store](#)。

示例 3：使用标签显示参数的值

以下 `get-parameter` 示例列出具有指定标签的指定单个参数的值。

```
aws ssm get-parameter \  
  --name "MyParameter:label"
```

输出：

```
{  
  "Parameters": [  
    {  
      "Name": "MyLabelParameter",  
      "Type": "String",  
      "Value": "parameter by label",  
      "Version": 1,  
      "Selector": ":label",  
      "LastModifiedDate": "2021-07-12T09:49:15.865000-07:00",  
      "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/MyParameter",  
      "DataType": "text"  
    },  
    {  
      "Name": "MyVersionParameter",
```

```
        "Type": "String",
        "Value": "parameter by version",
        "Version": 2,
        "Selector": ":2",
        "LastModifiedDate": "2021-03-24T16:20:28.236000-07:00",
        "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/unlabel-param",
        "DataType": "text"
    }
],
  "InvalidParameters": []
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用参数标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetParameters](#)中的。

## get-patch-baseline-for-patch-group

以下代码示例演示了如何使用 `get-patch-baseline-for-patch-group`。

### AWS CLI

显示补丁组的补丁基准

以下 `get-patch-baseline-for-patch-group` 示例检索有关指定补丁组补丁基准的详细信息。

```
aws ssm get-patch-baseline-for-patch-group \
  --patch-group "DEV"
```

输出：

```
{
  "PatchGroup": "DEV",
  "BaselineId": "pb-0123456789abcdef0",
  "OperatingSystem": "WINDOWS"
}
```

有关更多信息，请参阅 Systems Manager AWS CLI 用户指南中的创建补丁组 < <https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-group-tagging.html> > 和[向补丁基准添加补丁组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPatchBaselineForPatchGroup](#)中的。

## get-patch-baseline

以下代码示例演示了如何使用 get-patch-baseline。

### AWS CLI

#### 显示补丁基准

以下 get-patch-baseline 示例检索指定补丁基准的详细信息。

```
aws ssm get-patch-baseline \  
  --baseline-id "pb-0123456789abcdef0"
```

输出：

```
{  
  "BaselineId": "pb-0123456789abcdef0",  
  "Name": "WindowsPatching",  
  "OperatingSystem": "WINDOWS",  
  "GlobalFilters": {  
    "PatchFilters": []  
  },  
  "ApprovalRules": {  
    "PatchRules": [  
      {  
        "PatchFilterGroup": {  
          "PatchFilters": [  
            {  
              "Key": "PRODUCT",  
              "Values": [  
                "WindowsServer2016"  
              ]  
            }  
          ]  
        },  
        "ComplianceLevel": "CRITICAL",  
        "ApproveAfterDays": 0,  
        "EnableNonSecurity": false  
      }  
    ]  
  },  
}
```

```
"ApprovedPatches": [],
"ApprovedPatchesComplianceLevel": "UNSPECIFIED",
"ApprovedPatchesEnableNonSecurity": false,
"RejectedPatches": [],
"RejectedPatchesAction": "ALLOW_AS_DEPENDENCY",
"PatchGroups": [
  "QA",
  "DEV"
],
"CreateDate": 1550244180.465,
"ModifiedDate": 1550244180.465,
"Description": "Patches for Windows Servers",
"Sources": []
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[关于补丁基准](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetPatchBaseline](#)中的。

## get-service-setting

以下代码示例演示了如何使用 get-service-setting。

### AWS CLI

检索参数存储吞吐量的服务设置

以下 get-service-setting 示例检索指定区域中参数存储吞吐量的当前服务设置。

```
aws ssm get-service-setting \
  --setting-id arn:aws:ssm:us-east-1:123456789012:servicesetting/ssm/parameter-
  store/high-throughput-enabled
```

输出：

```
{
  "ServiceSetting": {
    "SettingId": "/ssm/parameter-store/high-throughput-enabled",
    "SettingValue": "false",
    "LastModifiedDate": 1555532818.578,
    "LastModifiedUser": "System",
    "ARN": "arn:aws:ssm:us-east-1:123456789012:servicesetting/ssm/parameter-
    store/high-throughput-enabled",
```

```
    "Status": "Default"
  }
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[“提高参数存储吞吐量”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetServiceSetting](#)中的。

## label-parameter-version

以下代码示例演示了如何使用 label-parameter-version。

### AWS CLI

示例 1：为最新版本的参数添加标签

以下label-parameter-version示例为指定参数的最新版本添加标签。

```
aws ssm label-parameter-version \
  --name "MyStringParameter" \
  --labels "ProductionReady"
```

输出：

```
{
  "InvalidLabels": [],
  "ParameterVersion": 3
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用参数标签](#)。

示例 2：为参数的特定版本添加标签

以下label-parameter-version示例为参数的指定版本添加标签。

```
aws ssm label-parameter-version \
  --name "MyStringParameter" \
  --labels "ProductionReady" \
  --parameter-version "2" --labels "DevelopmentReady"
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用参数标签](#)。



- 有关 API 的详细信息，请参阅AWS CLI 命令参考[LabelParameterVersion](#)中的。

## list-association-versions

以下代码示例演示了如何使用 list-association-versions。

### AWS CLI

列出特定关联 ID 的关联的所有版本

以下 list-association-versions 示例列出指定关联的所有版本。

```
aws ssm list-association-versions \  
  --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

输出：

```
{  
  "AssociationVersions": [  
    {  
      "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",  
      "AssociationVersion": "1",  
      "CreateDate": 1550505536.726,  
      "Name": "AWS-UpdateSSMAgent",  
      "Parameters": {  
        "allowDowngrade": [  
          "false"  
        ],  
        "version": [  
          ""  
        ]  
      },  
      "Targets": [  
        {  
          "Key": "InstanceIds",  
          "Values": [  
            "i-1234567890abcdef0"  
          ]  
        }  
      ],  
      "ScheduleExpression": "cron(0 00 12 ? * SUN *)",  
      "AssociationName": "UpdateSSMAgent"  
    }  
  ]  
}
```

```
]
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[在 Systems Manager 中使用关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListAssociationVersions](#)中的。

## list-associations

以下代码示例演示了如何使用 list-associations。

### AWS CLI

示例 1：列出特定实例的关联

以下列表关联示例列出了与 AssociationName updatesmAgent 的所有关联。

```
aws ssm list-associations /
  --association-filter-list "key=AssociationName,value=UpdateSSMAgent"
```

输出：

```
{
  "Associations": [
    {
      "Name": "AWS-UpdateSSMAgent",
      "InstanceId": "i-1234567890abcdef0",
      "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
      "AssociationVersion": "1",
      "Targets": [
        {
          "Key": "InstanceIds",
          "Values": [
            "i-016648b75dd622dab"
          ]
        }
      ],
      "Overview": {
        "Status": "Pending",
        "DetailedStatus": "Associated",
        "AssociationStatusAggregatedCount": {
          "Pending": 1
        }
      }
    }
  ]
}
```

```

    }
  },
  "ScheduleExpression": "cron(0 00 12 ? * SUN *)",
  "AssociationName": "UpdateSSMAgent"
}
]
}

```

有关更多信息，请参阅《Systems Manager 用户指南》中的[在 Systems Manager 中使用关联](#)。

## 示例 2：列出特定文档的关联

以下 list-associations 示例列出指定文档的所有关联。

```

aws ssm list-associations /
  --association-filter-list "key=Name,value=AWS-UpdateSSMAgent"

```

输出：

```

{
  "Associations": [
    {
      "Name": "AWS-UpdateSSMAgent",
      "InstanceId": "i-1234567890abcdef0",
      "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
      "AssociationVersion": "1",
      "Targets": [
        {
          "Key": "InstanceIds",
          "Values": [
            "i-1234567890abcdef0"
          ]
        }
      ],
      "LastExecutionDate": 1550505828.548,
      "Overview": {
        "Status": "Success",
        "DetailedStatus": "Success",
        "AssociationStatusAggregatedCount": {
          "Success": 1
        }
      },
      "ScheduleExpression": "cron(0 00 12 ? * SUN *)",

```

```
    "AssociationName": "UpdateSSMAgent"
  },
  {
    "Name": "AWS-UpdateSSMAgent",
    "InstanceId": "i-9876543210abcdef0",
    "AssociationId": "fbc07ef7-b985-4684-b82b-0123456789ab",
    "AssociationVersion": "1",
    "Targets": [
      {
        "Key": "InstanceIds",
        "Values": [
          "i-9876543210abcdef0"
        ]
      }
    ],
    "LastExecutionDate": 1550507531.0,
    "Overview": {
      "Status": "Success",
      "AssociationStatusAggregatedCount": {
        "Success": 1
      }
    }
  }
]
}
```

有关更多信息，请参阅《Systems Manager 用户指南》中的[在 Systems Manager 中使用关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListAssociations](#)中的。

## list-command-invocations

以下代码示例演示了如何使用 list-command-invocations。

### AWS CLI

列出特定命令的调用

以下 list-command-invocations 示例列出命令的所有调用。

```
aws ssm list-command-invocations \
  --command-id "ef7fdfd8-9b57-4151-a15c-db9a12345678" \
  --details
```

输出：

```
{
  "CommandInvocations": [
    {
      "CommandId": "ef7fd8-9b57-4151-a15c-db9a12345678",
      "InstanceId": "i-02573cafcfEXAMPLE",
      "InstanceName": "",
      "Comment": "b48291dd-ba76-43e0-
b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-d6ce8EXAMPLE",
      "DocumentName": "AWS-UpdateSSMAgent",
      "DocumentVersion": "",
      "RequestedDateTime": 1582136283.089,
      "Status": "Success",
      "StatusDetails": "Success",
      "StandardOutputUrl": "",
      "StandardErrorUrl": "",
      "CommandPlugins": [
        {
          "Name": "aws:updateSsmAgent",
          "Status": "Success",
          "StatusDetails": "Success",
          "ResponseCode": 0,
          "ResponseStartDateTime": 1582136283.419,
          "ResponseFinishDateTime": 1582136283.51,
          "Output": "\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/
ssm-agent-manifest.json\namazon-ssm-agent 2.3.842.0 has already been installed,
update skipped\n",
          "StandardOutputUrl": "",
          "StandardErrorUrl": "",
          "OutputS3Region": "us-east-2",
          "OutputS3BucketName": "",
          "OutputS3KeyPrefix": ""
        }
      ],
      "ServiceRole": "",
      "NotificationConfig": {
        "NotificationArn": "",
        "NotificationEvents": [],
        "NotificationType": ""
      },
      "CloudWatchOutputConfig": {
        "CloudWatchLogGroupName": "",

```

```

        "CloudWatchOutputEnabled": false
    }
},
{
    "CommandId": "ef7fd8-9b57-4151-a15c-db9a12345678",
    "InstanceId": "i-0471e04240EXAMPLE",
    "InstanceName": "",
    "Comment": "b48291dd-ba76-43e0-
b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-d6ce8EXAMPLE",
    "DocumentName": "AWS-UpdateSSMAgent",
    "DocumentVersion": "",
    "RequestedDateTime": 1582136283.02,
    "Status": "Success",
    "StatusDetails": "Success",
    "StandardOutputUrl": "",
    "StandardErrorUrl": "",
    "CommandPlugins": [
        {
            "Name": "aws:updateSsmAgent",
            "Status": "Success",
            "StatusDetails": "Success",
            "ResponseCode": 0,
            "ResponseStartDateTime": 1582136283.812,
            "ResponseFinishDateTime": 1582136295.031,
            "Output": "Updating amazon-ssm-agent from 2.3.672.0 to latest
\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/
ssm-agent-manifest.json\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/
amazon-ssm-us-east-2/amazon-ssm-agent-updater/2.3.842.0/amazon-ssm-agent-updater-
snap-amd64.tar.gz\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/
amazon-ssm-us-east-2/amazon-ssm-agent/2.3.672.0/amazon-ssm-agent-snap-amd64.tar.gz
\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/
amazon-ssm-agent/2.3.842.0/amazon-ssm-agent-snap-amd64.tar.gz\nInitiating amazon-
ssm-agent update to 2.3.842.0\namazon-ssm-agent updated successfully to 2.3.842.0",
            "StandardOutputUrl": "",
            "StandardErrorUrl": "",
            "OutputS3Region": "us-east-2",
            "OutputS3BucketName": "",
            "OutputS3KeyPrefix": "8bee3135-398c-4d31-99b6-e42d2EXAMPLE/
i-0471e04240EXAMPLE/awsupdateSsmAgent"
        }
    ],
    "ServiceRole": "",
    "NotificationConfig": {
        "NotificationArn": "",

```

```
        "NotificationEvents": [],
        "NotificationType": ""
    },
    "CloudWatchOutputConfig": {
        "CloudWatchLogGroupName": "",
        "CloudWatchOutputEnabled": false
    }
}
]
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[了解命令状态](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListCommandInvocations](#) 中的。

## list-commands

以下代码示例演示了如何使用 `list-commands`。

### AWS CLI

示例 1：获取特定命令的状态

以下 `list-commands` 示例检索并显示指定命令的状态。

```
aws ssm list-commands \  
  --command-id "0831e1a8-a1ac-4257-a1fd-c831bEXAMPLE"
```

示例 2：获取特定日期之后请求的命令的状态

以下 `list-commands` 示例检索在指定日期之后请求的命令的详细信息。

```
aws ssm list-commands \  
  --filter "key=InvokedAfter,value=2020-02-01T00:00:00Z"
```

示例 3：列出 AWS 账户中请求的所有命令

以下 `list-commands` 示例列出了当前 AWS 账户和区域中的用户请求的所有命令。

```
aws ssm list-commands
```

输出：

```
{
  "Commands": [
    {
      "CommandId": "8bee3135-398c-4d31-99b6-e42d2EXAMPLE",
      "DocumentName": "AWS-UpdateSSMAgent",
      "DocumentVersion": "",
      "Comment": "b48291dd-ba76-43e0-
b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-d6ce8EXAMPLE",
      "ExpiresAfter": "2020-02-19T11:28:02.500000-08:00",
      "Parameters": {},
      "InstanceIds": [
        "i-028ea792daEXAMPLE",
        "i-02feef8c46EXAMPLE",
        "i-038613f3f0EXAMPLE",
        "i-03a530a2d4EXAMPLE",
        "i-083b678d37EXAMPLE",
        "i-0dee81debaEXAMPLE"
      ],
      "Targets": [],
      "RequestedDateTime": "2020-02-19T10:18:02.500000-08:00",
      "Status": "Success",
      "StatusDetails": "Success",
      "OutputS3BucketName": "",
      "OutputS3KeyPrefix": "",
      "MaxConcurrency": "50",
      "MaxErrors": "100%",
      "TargetCount": 6,
      "CompletedCount": 6,
      "ErrorCount": 0,
      "DeliveryTimedOutCount": 0,
      "ServiceRole": "",
      "NotificationConfig": {
        "NotificationArn": "",
        "NotificationEvents": [],
        "NotificationType": ""
      },
      "CloudWatchOutputConfig": {
        "CloudWatchLogGroupName": "",
        "CloudWatchOutputEnabled": false
      }
    }
  ]
}
```



```
"CommandId": "e9ade581-c03d-476b-9b07-26667EXAMPLE",
"DocumentName": "AWS-FindWindowsUpdates",
"DocumentVersion": "1",
"Comment": "",
"ExpiresAfter": "2020-01-24T12:37:31.874000-08:00",
"Parameters": {
  "KbArticleIds": [
    ""
  ],
  "UpdateLevel": [
    "All"
  ]
},
"InstanceIds": [],
"Targets": [
  {
    "Key": "InstanceIds",
    "Values": [
      "i-00ec29b21eEXAMPLE",
      "i-09911ddd90EXAMPLE"
    ]
  }
],
"RequestedDateTime": "2020-01-24T11:27:31.874000-08:00",
"Status": "Success",
"StatusDetails": "Success",
"OutputS3BucketName": "my-us-east-2-bucket",
"OutputS3KeyPrefix": "my-rc-output",
"MaxConcurrency": "50",
"MaxErrors": "0",
"TargetCount": 2,
"CompletedCount": 2,
"ErrorCount": 0,
"DeliveryTimedOutCount": 0,
"ServiceRole": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
"NotificationConfig": {
  "NotificationArn": "arn:aws:sns:us-east-2:111222333444:my-us-east-2-
notification-arn",
  "NotificationEvents": [
    "All"
  ],
  "NotificationType": "Invocation"
},
}
```

```
    "CloudWatchOutputConfig": {
      "CloudWatchLogGroupName": "",
      "CloudWatchOutputEnabled": false
    }
  }
  {
    "CommandId": "d539b6c3-70e8-4853-80e5-0ce4fEXAMPLE",
    "DocumentName": "AWS-RunPatchBaseline",
    "DocumentVersion": "1",
    "Comment": "",
    "ExpiresAfter": "2020-01-24T12:21:04.350000-08:00",
    "Parameters": {
      "InstallOverrideList": [
        ""
      ],
      "Operation": [
        "Install"
      ],
      "RebootOption": [
        "RebootIfNeeded"
      ],
      "SnapshotId": [
        ""
      ]
    },
    "InstanceIds": [],
    "Targets": [
      {
        "Key": "InstanceIds",
        "Values": [
          "i-00ec29b21eEXAMPLE",
          "i-09911ddd90EXAMPLE"
        ]
      }
    ]
  },
  "RequestedDateTime": "2020-01-24T11:11:04.350000-08:00",
  "Status": "Success",
  "StatusDetails": "Success",
  "OutputS3BucketName": "my-us-east-2-bucket",
  "OutputS3KeyPrefix": "my-rc-output",
  "MaxConcurrency": "50",
  "MaxErrors": "0",
  "TargetCount": 2,
  "CompletedCount": 2,
```

```
    "ErrorCount": 0,
    "DeliveryTimedOutCount": 0,
    "ServiceRole": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
    "NotificationConfig": {
      "NotificationArn": "arn:aws:sns:us-east-2:111222333444:my-us-east-2-
notification-arn",
      "NotificationEvents": [
        "All"
      ],
      "NotificationType": "Invocation"
    },
    "CloudWatchOutputConfig": {
      "CloudWatchLogGroupName": "",
      "CloudWatchOutputEnabled": false
    }
  }
]
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Systems Manager Run Command 运行命令](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListCommands](#)中的。

## list-compliance-items

以下代码示例演示了如何使用 list-compliance-items。

### AWS CLI

列出特定实例的合规性项目

此示例列出指定实例的所有合规性项目。

命令:

```
aws ssm list-compliance-items --resource-ids "i-1234567890abcdef0" --resource-types
"ManagedInstance"
```

输出:

```
{
```

```
"ComplianceItems": [  
  {  
    "ComplianceType": "Association",  
    "ResourceType": "ManagedInstance",  
    "ResourceId": "i-1234567890abcdef0",  
    "Id": "8dfe3659-4309-493a-8755-0123456789ab",  
    "Title": "",  
    "Status": "COMPLIANT",  
    "Severity": "UNSPECIFIED",  
    "ExecutionSummary": {  
      "ExecutionTime": 1550408470.0  
    },  
    "Details": {  
      "DocumentName": "AWS-GatherSoftwareInventory",  
      "DocumentVersion": "1"  
    }  
  },  
  {  
    "ComplianceType": "Association",  
    "ResourceType": "ManagedInstance",  
    "ResourceId": "i-1234567890abcdef0",  
    "Id": "e4c2ed6d-516f-41aa-aa2a-0123456789ab",  
    "Title": "",  
    "Status": "COMPLIANT",  
    "Severity": "UNSPECIFIED",  
    "ExecutionSummary": {  
      "ExecutionTime": 1550508475.0  
    },  
    "Details": {  
      "DocumentName": "AWS-UpdateSSMAgent",  
      "DocumentVersion": "1"  
    }  
  },  
  ...  
],  
"NextToken": "--token string truncated--"  
}
```

列出特定实例和关联 ID 的合规性项目

此示例列出指定实例和关联 ID 的所有合规性项目。

命令:

```
aws ssm list-compliance-items --resource-ids "i-1234567890abcdef0" --resource-types
"ManagedInstance" --filters "Key=ComplianceType,Values=Association,Type=EQUAL"
"Key=Id,Values=e4c2ed6d-516f-41aa-aa2a-0123456789ab,Type=EQUAL"
```

列出特定日期和时间之后实例的合规性项目

此示例列出指定日期和时间之后实例的所有合规性项目。

命令:

```
aws ssm list-compliance-items --resource-ids "i-1234567890abcdef0" --resource-types
"ManagedInstance" --filters
"Key=ExecutionTime,Values=2019-02-18T16:00:00Z,Type=GREATER_THAN"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListComplianceItems](#)中的。

## list-compliance-summaries

以下代码示例演示了如何使用 list-compliance-summaries。

### AWS CLI

列出所有合规性类型的合规性摘要

此示例列出您账户中所有合规性类型的合规性摘要。

命令:

```
aws ssm list-compliance-summaries
```

输出:

```
{
  "ComplianceSummaryItems": [
    {
      "ComplianceType": "Association",
      "CompliantSummary": {
        "CompliantCount": 2,
        "SeveritySummary": {
          "CriticalCount": 0,
          "HighCount": 0,
          "MediumCount": 0,

```

```
        "LowCount": 0,
        "InformationalCount": 0,
        "UnspecifiedCount": 2
    }
},
"NonCompliantSummary": {
    "NonCompliantCount": 0,
    "SeveritySummary": {
        "CriticalCount": 0,
        "HighCount": 0,
        "MediumCount": 0,
        "LowCount": 0,
        "InformationalCount": 0,
        "UnspecifiedCount": 0
    }
}
},
{
    "ComplianceType": "Patch",
    "CompliantSummary": {
        "CompliantCount": 1,
        "SeveritySummary": {
            "CriticalCount": 0,
            "HighCount": 0,
            "MediumCount": 0,
            "LowCount": 0,
            "InformationalCount": 0,
            "UnspecifiedCount": 1
        }
    },
    "NonCompliantSummary": {
        "NonCompliantCount": 1,
        "SeveritySummary": {
            "CriticalCount": 1,
            "HighCount": 0,
            "MediumCount": 0,
            "LowCount": 0,
            "InformationalCount": 0,
            "UnspecifiedCount": 0
        }
    }
},
...
],
```

```
"NextToken": "eyJ0ZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ=="
}
```

列出特定合规性类型的合规性摘要

此示例列出补丁合规性类型的合规性摘要。

命令:

```
aws ssm list-compliance-summaries --filters
"Key=ComplianceType,Values=Patch,Type=EQUAL"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListComplianceSummaries](#)中的。

## list-document-metadata-history

以下代码示例演示了如何使用 list-document-metadata-history。

AWS CLI

示例：查看更改模板的批准历史记录和状态

以下list-document-metadata-history示例返回指定 Change Manager 变更模板的批准历史记录。

```
aws ssm list-document-metadata-history \
--name MyChangeManageTemplate \
--metadata DocumentReviews
```

输出：

```
{
  "Name": "MyChangeManagerTemplate",
  "DocumentVersion": "1",
  "Author": "arn:aws:iam::111222333444::user/JohnDoe",
  "Metadata": {
    "ReviewerResponse": [
      {
        "CreateTime": "2021-07-30T11:58:28.025000-07:00",
        "UpdateTime": "2021-07-30T12:01:19.274000-07:00",
        "ReviewStatus": "APPROVED",
```

```

        "Comment": [
            {
                "Type": "COMMENT",
                "Content": "I approve this template version"
            }
        ],
        "Reviewer": "arn:aws:iam::111222333444:user/ShirleyRodriguez"
    },
    {
        "CreateTime": "2021-07-30T11:58:28.025000-07:00",
        "UpdateTime": "2021-07-30T11:58:28.025000-07:00",
        "ReviewStatus": "PENDING"
    }
]
}

```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[审阅、批准或拒绝变更模板](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListDocumentMetadataHistory](#)中的。

## list-document-versions

以下代码示例演示了如何使用 list-document-versions。

### AWS CLI

列出文档版本

以下 list-document-versions 示例列出 Systems Manager 文档的所有版本。

```
aws ssm list-document-versions \
  --name "Example"
```

输出：

```
{
  "DocumentVersions": [
    {
      "Name": "Example",
      "DocumentVersion": "1",
      "CreateDate": 1583257938.266,
```



```
        "IsDefaultVersion": true,  
        "DocumentFormat": "YAML",  
        "Status": "Active"  
      }  
    ]  
  }  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[发送使用文档版本参数的命令](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDocumentVersions](#) 中的。

## list-documents

以下代码示例演示了如何使用 list-documents。

### AWS CLI

#### 示例 1：列出文档

以下 list-documents 示例列出标有自定义标签的请求账户拥有的文档。

```
aws ssm list-documents \  
  --filters Key=Owner,Values=Self Key=tag:DocUse,Values=Testing
```

输出：

```
{  
  "DocumentIdentifiers": [  
    {  
      "Name": "Example",  
      "Owner": "29884EXAMPLE",  
      "PlatformTypes": [  
        "Windows",  
        "Linux"  
      ],  
      "DocumentVersion": "1",  
      "DocumentType": "Automation",  
      "SchemaVersion": "0.3",  
      "DocumentFormat": "YAML",  
      "Tags": [  
        {  
          "Key": "DocUse",  
          "Value": "Testing"  
        }  
      ]  
    }  
  ]  
}
```

```
    }
  ]
}
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的 [AWS Systems Manager 文档](#)。

## 示例 2：列出共享文档

以下 `list-documents` 示例列出了共享文档，包括不属于的私有共享文档 AWS。

```
aws ssm list-documents \
  --filters Key=Name,Values=sharedDocNamePrefix Key=Owner,Values=Private
```

输出：

```
{
  "DocumentIdentifiers": [
    {
      "Name": "Example",
      "Owner": "12345EXAMPLE",
      "PlatformTypes": [
        "Windows",
        "Linux"
      ],
      "DocumentVersion": "1",
      "DocumentType": "Command",
      "SchemaVersion": "0.3",
      "DocumentFormat": "YAML",
      "Tags": []
    }
  ]
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的 [AWS Systems Manager 文档](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListDocuments](#) 中的。

## list-inventory-entries

以下代码示例演示了如何使用 `list-inventory-entries`。

## AWS CLI

### 示例 1：查看实例的特定清单类型条目

以下 `list-inventory-entries` 示例列出了特定实例上:Application AWS清单类型的清单条目。

```
aws ssm list-inventory-entries \  
  --instance-id "i-1234567890abcdef0" \  
  --type-name "AWS:Application"
```

输出：

```
{  
  "TypeName": "AWS:Application",  
  "InstanceId": "i-1234567890abcdef0",  
  "SchemaVersion": "1.1",  
  "CaptureTime": "2019-02-15T12:17:55Z",  
  "Entries": [  
    {  
      "Architecture": "i386",  
      "Name": "Amazon SSM Agent",  
      "PackageId": "{88a60be2-89a1-4df8-812a-80863c2a2b68}",  
      "Publisher": "Amazon Web Services",  
      "Version": "2.3.274.0"  
    },  
    {  
      "Architecture": "x86_64",  
      "InstalledTime": "2018-05-03T13:42:34Z",  
      "Name": "AmazonCloudWatchAgent",  
      "Publisher": "",  
      "Version": "1.200442.0"  
    }  
  ]  
}
```

### 示例 2：查看分配给实例的自定义清单条目

以下 `list-inventory-entries` 示例列出分配给实例的自定义清单条目。

```
aws ssm list-inventory-entries \  
  --instance-id "i-1234567890abcdef0" \  
  --type-name "Custom:RackInfo"
```

输出：

```
{
  "TypeName": "Custom:RackInfo",
  "InstanceId": "i-1234567890abcdef0",
  "SchemaVersion": "1.0",
  "CaptureTime": "2021-05-22T10:01:01Z",
  "Entries": [
    {
      "RackLocation": "Bay B/Row C/Rack D/Shelf E"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListInventoryEntries](#)中的。

## list-ops-item-related-items

以下代码示例演示了如何使用 list-ops-item-related-items。

AWS CLI

列出相关物品的资源 OpsItem

以下list-ops-item-related-items示例列出了的相关项目资源。 OpsItem

```
aws ssm list-ops-item-related-items \
  --ops-item-id "oi-f99f2EXAMPLE"
```

输出：

```
{
  "Summaries": [
    {
      "OpsItemId": "oi-f99f2EXAMPLE",
      "AssociationId": "e2036148-cccb-490e-ac2a-390e5EXAMPLE",
      "ResourceType": "AWS::SSMIncidents::IncidentRecord",
      "AssociationType": "IsParentOf",
      "ResourceUri": "arn:aws:ssm-incidents::111122223333:incident-record/example-response/64bd9b45-1d0e-2622-840d-03a87a1451fa",
      "CreatedBy": {
```

```

        "Arn": "arn:aws:sts::111122223333:assumed-role/
AWSServiceRoleForIncidentManager/IncidentResponse"
      },
      "CreatedTime": "2021-08-11T18:47:14.994000+00:00",
      "LastModifiedBy": {
        "Arn": "arn:aws:sts::111122223333:assumed-role/
AWSServiceRoleForIncidentManager/IncidentResponse"
      },
      "LastModifiedTime": "2021-08-11T18:47:14.994000+00:00"
    }
  ]
}

```

有关更多信息，请参阅《S AWS system [s Manager 用户指南](#)》OpsCenter中的“[处理事件管理器事件](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListOpsItemRelatedItems](#)中的。

## list-resource-compliance-summaries

以下代码示例演示了如何使用 `list-resource-compliance-summaries`。

### AWS CLI

列出资源级合规性摘要计数

此示例列出资源级合规性摘要计数。

命令:

```
aws ssm list-resource-compliance-summaries
```

输出:

```

{
  "ResourceComplianceSummaryItems": [
    {
      "ComplianceType": "Association",
      "ResourceType": "ManagedInstance",
      "ResourceId": "i-1234567890abcdef0",
      "Status": "COMPLIANT",
      "OverallSeverity": "UNSPECIFIED",
      "ExecutionSummary": {

```

```
        "ExecutionTime": 1550509273.0
    },
    "CompliantSummary": {
        "CompliantCount": 2,
        "SeveritySummary": {
            "CriticalCount": 0,
            "HighCount": 0,
            "MediumCount": 0,
            "LowCount": 0,
            "InformationalCount": 0,
            "UnspecifiedCount": 2
        }
    },
    "NonCompliantSummary": {
        "NonCompliantCount": 0,
        "SeveritySummary": {
            "CriticalCount": 0,
            "HighCount": 0,
            "MediumCount": 0,
            "LowCount": 0,
            "InformationalCount": 0,
            "UnspecifiedCount": 0
        }
    }
},
{
    "ComplianceType": "Patch",
    "ResourceType": "ManagedInstance",
    "ResourceId": "i-9876543210abcdef0",
    "Status": "COMPLIANT",
    "OverallSeverity": "UNSPECIFIED",
    "ExecutionSummary": {
        "ExecutionTime": 1550248550.0,
        "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
        "ExecutionType": "Command"
    },
    "CompliantSummary": {
        "CompliantCount": 397,
        "SeveritySummary": {
            "CriticalCount": 0,
            "HighCount": 0,
            "MediumCount": 0,
            "LowCount": 0,
            "InformationalCount": 0,
```

```
        "UnspecifiedCount": 397
      }
    },
    "NonCompliantSummary": {
      "NonCompliantCount": 0,
      "SeveritySummary": {
        "CriticalCount": 0,
        "HighCount": 0,
        "MediumCount": 0,
        "LowCount": 0,
        "InformationalCount": 0,
        "UnspecifiedCount": 0
      }
    }
  }
],
"NextToken": "--token string truncated--"
}
```

列出特定合规性规类型的资源级合规性摘要

此示例列出补丁合规性类型的资源级合规性摘要。

命令:

```
aws ssm list-resource-compliance-summaries --filters
"Key=ComplianceType,Values=Patch,Type=EQUAL"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResourceComplianceSummaries](#)中的。

## list-resource-data-sync

以下代码示例演示了如何使用 list-resource-data-sync。

AWS CLI

列出您的资源数据同步配置

此示例检索有关您的资源数据同步配置的信息。

```
aws ssm list-resource-data-sync
```

输出：

```
{
  "ResourceDataSyncItems": [
    {
      "SyncName": "MyResourceDataSync",
      "S3Destination": {
        "BucketName": "ssm-resource-data-sync",
        "SyncFormat": "JsonSerDe",
        "Region": "us-east-1"
      },
      "LastSyncTime": 1550261472.003,
      "LastSuccessfulSyncTime": 1550261472.003,
      "LastStatus": "Successful",
      "SyncCreatedTime": 1543235736.72,
      "LastSyncStatusMessage": "The sync was successfully completed"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResourceDataSync](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

AWS CLI

列出应用于补丁基准的标签

以下 `list-tags-for-resource` 示例列出了补丁基准的标签。

```
aws ssm list-tags-for-resource \
  --resource-type "PatchBaseline" \
  --resource-id "pb-0123456789abcdef0"
```

输出：

```
{
  "TagList": [
    {
```



```
        "Key": "Environment",
        "Value": "Production"
    },
    {
        "Key": "Region",
        "Value": "EMEA"
    }
]
}
```

有关更多信息，请参阅《AWS 一般参考》中的“为[AWS 资源添加标签](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## modify-document-permission

以下代码示例演示了如何使用 `modify-document-permission`。

### AWS CLI

#### 修改文档权限

以下 `modify-document-permission` 示例公开共享一个 Systems Manager 文档。

```
aws ssm modify-document-permission \
  --name "Example" \
  --permission-type "Share" \
  --account-ids-to-add "All"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[共享 Systems Manager 文档](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyDocumentPermission](#) 中的。

## put-compliance-items

以下代码示例演示了如何使用 `put-compliance-items`。

### AWS CLI

向指定实例注册合规性类型和合规性详细信息

此示例将合规性类型 `Custom:AVCheck` 注册到指定的托管式实例。如果此命令成功，则无任何输出。

命令:

```
aws ssm put-compliance-items --resource-id "i-1234567890abcdef0" --
resource-type "ManagedInstance" --compliance-type "Custom:AVCheck"
--execution-summary "ExecutionTime=2019-02-18T16:00:00Z" --items
"Id=Version2.0,Title=ScanHost,Severity=CRITICAL,Status=COMPLIANT"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutComplianceItems](#)中的。

## put-inventory

以下代码示例演示了如何使用 `put-inventory`。

### AWS CLI

将客户元数据分配给实例

此示例将机架位置信息分配给某个实例。如果此命令成功，则无任何输出。

命令 ( Linux ) :

```
aws ssm put-inventory --instance-id "i-016648b75dd622dab" --items
' [{"TypeName": "Custom:RackInfo", "SchemaVersion": "1.0", "CaptureTime":
"2019-01-22T10:01:01Z", "Content": [{"RackLocation": "Bay B/Row C/Rack D/Shelf
E"}]} ]'
```

命令 ( Windows ) :

```
aws ssm put-inventory --instance-id "i-016648b75dd622dab" --items
"TypeName=Custom:RackInfo,SchemaVersion=1.0,CaptureTime=2019-01-22T10:01:01Z,Content=[{Rack
B/Row C/Rack D/Shelf F'}]"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutInventory](#)中的。

## put-parameter

以下代码示例演示了如何使用 `put-parameter`。

## AWS CLI

### 示例 1：更改参数值

以下 `put-parameter` 示例更改了指定参数的值。

```
aws ssm put-parameter \  
  --name "MyStringParameter" \  
  --type "String" \  
  --value "Vici" \  
  --overwrite
```

输出：

```
{  
  "Version": 2,  
  "Tier": "Standard"  
}
```

有关更多信息，请参阅《[Systems Manager 用户指南](#)》中的[创建 Systems Manager 参数 \(AWS CLI\)](#)、“[管理参数层 < https://docs.aws.amazon.com/systems-manager/latest/userguide/parameter-store-advanced-parameters .html>](https://docs.aws.amazon.com/systems-manager/latest/userguide/parameter-store-advanced-parameters.html)”和“[使用参数策略](#)”。AWS

### 示例 2：创建高级参数

以下 `put-parameter` 示例将创建高级参数。

```
aws ssm put-parameter \  
  --name "MyAdvancedParameter" \  
  --description "This is an advanced parameter" \  
  --value "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod  
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,  
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat  
[truncated]" \  
  --type "String" \  
  --tier Advanced
```

输出：

```
{  
  "Version": 1,
```

```
"Tier": "Advanced"
}
```

有关更多信息，请参阅《[Systems Manager 用户指南](#)》中的[创建 Systems Manager 参数 \(AWS CLI\)](#)、“[管理参数层 < https://docs.aws.amazon.com/systems-manager/latest/userguide/parameter-store-advanced-parameters.html>](#)”和“[使用参数策略](#)”。AWS

示例 3：将标准参数转换为高级参数

以下 `put-parameter` 示例将现有标准参数转换为高级参数。

```
aws ssm put-parameter \
  --name "MyConvertedParameter" \
  --value "abc123" \
  --type "String" \
  --tier Advanced \
  --overwrite
```

输出：

```
{
  "Version": 2,
  "Tier": "Advanced"
}
```

有关更多信息，请参阅《[Systems Manager 用户指南](#)》中的[创建 Systems Manager 参数 \(AWS CLI\)](#)、“[管理参数层 < https://docs.aws.amazon.com/systems-manager/latest/userguide/parameter-store-advanced-parameters.html>](#)”和“[使用参数策略](#)”。AWS

示例 4：创建附加有策略的参数

以下 `put-parameter` 示例创建了一个附加参数策略的高级参数。

```
aws ssm put-parameter \
  --name "/Finance/Payroll/q2accesskey" \
  --value "P@sSw)rd" \
  --type "SecureString" \
  --tier Advanced \
  --policies "[{"Type": "Expiration", "Version": "1.0", "Attributes": {"Timestamp": "2020-06-30T00:00:00.000Z"}}, {"Type": "ExpirationNotification", "Version": "1.0", "Attributes": {"Before": "5", "Unit": "Days"}}, {"Type":
```

```
\"NoChangeNotification\", \"Version\": \"1.0\", \"Attributes\": {\"After\": \"60\", \"Unit\": \"Days\"}]"]
```

输出：

```
{
  "Version": 1,
  "Tier": "Advanced"
}
```

有关更多信息，请参阅《[Systems Manager 用户指南](#)》中的[创建 Systems Manager 参数 \(AWS CLI\)](#)、“[管理参数层 < https://docs.aws.amazon.com/systems-manager/latest/userguide/parameter-store-advanced-parameters.html>`\\_`](#)”和“[使用参数策略](#)”。AWS

示例 5：向现有参数添加策略

以下 `put-parameter` 示例将策略附加到现有高级参数。

```
aws ssm put-parameter \
  --name "/Finance/Payroll/q2accesskey" \
  --value "N3wP@sSwW)rd" \
  --type "SecureString" \
  --tier Advanced \
  --policies "[{\"Type\": \"Expiration\", \"Version\": \"1.0\", \"Attributes\": {\"Timestamp\": \"2020-06-30T00:00:00.000Z\"}}, {\"Type\": \"ExpirationNotification\", \"Version\": \"1.0\", \"Attributes\": {\"Before\": \"5\", \"Unit\": \"Days\"}}, {\"Type\": \"NoChangeNotification\", \"Version\": \"1.0\", \"Attributes\": {\"After\": \"60\", \"Unit\": \"Days\"}}]"
  --overwrite
```

输出：

```
{
  "Version": 2,
  "Tier": "Advanced"
}
```

有关更多信息，请参阅《[Systems Manager 用户指南](#)》中的[创建 Systems Manager 参数 \(AWS CLI\)](#)、“[管理参数层 < https://docs.aws.amazon.com/systems-manager/latest/userguide/parameter-store-advanced-parameters.html>`\\_`](#)”和“[使用参数策略](#)”。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutParameter](#) 中的。

## register-default-patch-baseline

以下代码示例演示了如何使用 register-default-patch-baseline。

### AWS CLI

#### 设置默认补丁基准

以下 register-default-patch-baseline 示例将指定的自定义补丁基准注册为其支持的操作系统类型的默认补丁基准。

```
aws ssm register-default-patch-baseline \  
  --baseline-id "pb-abc123cf9bEXAMPLE"
```

输出：

```
{  
  "BaselineId": "pb-abc123cf9bEXAMPLE"  
}
```

以下 register-default-patch-baseline 示例将 CentOS 提供的默认补丁基准注册 AWS 为默认补丁基准。

```
aws ssm register-default-patch-baseline \  
  --baseline-id "arn:aws:ssm:us-east-2:733109147000:patchbaseline/  
pb-0574b43a65ea646ed"
```

输出：

```
{  
  "BaselineId": "pb-abc123cf9bEXAMPLE"  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[关于预定义和自定义补丁基准](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterDefaultPatchBaseline](#) 中的。

## register-patch-baseline-for-patch-group

以下代码示例演示了如何使用 register-patch-baseline-for-patch-group。

## AWS CLI

### 为补丁组注册补丁基准

以下 `register-patch-baseline-for-patch-group` 示例为补丁组注册补丁基准。

```
aws ssm register-patch-baseline-for-patch-group \  
  --baseline-id "pb-045f10b4f382baeda" \  
  --patch-group "Production"
```

输出：

```
{  
  "BaselineId": "pb-045f10b4f382baeda",  
  "PatchGroup": "Production"  
}
```

有关更多信息，请参阅 Systems Manager AWS 用户指南中的创建补丁组 < <https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-group-tagging.html> > 和 [向补丁基准添加补丁组](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterPatchBaselineForPatchGroup](#) 中的。

## **register-target-with-maintenance-window**

以下代码示例演示了如何使用 `register-target-with-maintenance-window`。

### AWS CLI

示例 1：向维护时段注册单个目标

以下 `register-target-with-maintenance-window` 示例向维护时段注册实例。

```
aws ssm register-target-with-maintenance-window \  
  --window-id "mw-ab12cd34ef56gh78" \  
  --target "Key=InstanceIds,Values=i-0000293ffd8c57862" \  
  --owner-information "Single instance" \  
  --resource-type "INSTANCE"
```

输出：

```
{
```

```
    "WindowTargetId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
  }
```

### 示例 2：使用实例 ID 向维护时段注册多个目标

以下 `register-target-with-maintenance-window` 示例通过指定其实例 ID 向维护时段注册两个实例。

```
aws ssm register-target-with-maintenance-window \
  --window-id "mw-ab12cd34ef56gh78" \
  --target "Key=InstanceIds,Values=i-0000293ffd8c57862,i-0cb2b964d3e14fd9f" \
  --owner-information "Two instances in a list" \
  --resource-type "INSTANCE"
```

输出：

```
{
  "WindowTargetId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

### 示例 3：使用资源标签向维护时段注册目标

以下 `register-target-with-maintenance-window` 示例通过指定已应用于实例的资源标签，向维护时段注册实例。

```
aws ssm register-target-with-maintenance-window \
  --window-id "mw-06cf17cbefcb4bf4f" \
  --targets "Key=tag:Environment,Values=Prod" "Key=Role,Values=Web" \
  --owner-information "Production Web Servers" \
  --resource-type "INSTANCE"
```

输出：

```
{
  "WindowTargetId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

### 示例 4：使用一组标签键注册目标

以下 `register-target-with-maintenance-window` 示例注册所有被分配了一个或多个标签键的实例（不考虑其键值）。



```
aws ssm register-target-with-maintenance-window \  
  --window-id "mw-0c50858d01EXAMPLE" \  
  --resource-type "INSTANCE" \  
  --target "Key=tag-key,Values=Name,Instance-Type,CostCenter"
```

输出：

```
{  
  "WindowTargetId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"  
}
```

示例 5：使用资源组名称注册目标

以下 `register-target-with-maintenance-window` 示例注册指定的资源组，无论其包含的资源类型如何。

```
aws ssm register-target-with-maintenance-window \  
  --window-id "mw-0c50858d01EXAMPLE" \  
  --resource-type "RESOURCE_GROUP" \  
  --target "Key=resource-groups:Name,Values=MyResourceGroup"
```

输出：

```
{  
  "WindowTargetId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"  
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[在维护时段 \(AWS CLI\) 注册目标实例](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterTargetWithMaintenanceWindow](#)中的。

## **register-task-with-maintenance-window**

以下代码示例演示了如何使用 `register-task-with-maintenance-window`。

AWS CLI

示例 1：向维护时段注册 Automation 任务

以下 `register-task-with-maintenance-window` 示例向针对实例的维护时段注册 Automation 任务。

```
aws ssm register-task-with-maintenance-window \
  --window-id "mw-082dcd7649EXAMPLE" \
  --targets Key=InstanceIds,Values=i-1234520122EXAMPLE \
  --task-arn AWS-RestartEC2Instance \
  --service-role-arn arn:aws:iam::111222333444:role/SSM --task-type AUTOMATION \
  --task-invocation-parameters "{\"Automation\":{\"DocumentVersion\":\"\$LATEST\",
  \Parameters\":{\"InstanceId\":[\"{{RESOURCE_ID}}\"]}}\" \
  --priority 0 \
  --max-concurrency 1 \
  --max-errors 1 \
  --name "AutomationExample" \
  --description "Restarting EC2 Instance for maintenance"
```

输出：

```
{
  "WindowTaskId":"11144444-5555-6666-7777-88888888"
}
```

有关更多信息，请参阅 [《S AWS systems Manager 用户指南》](#) 中的在维护时段 (AWS CLI) 注册任务。

示例 2：向维护时段注册 Lambda 任务

以下 `register-task-with-maintenance-window` 示例向针对实例的维护时段注册 Lambda 任务。

```
aws ssm register-task-with-maintenance-window \
  --window-id "mw-082dcd7649dee04e4" \
  --targets Key=InstanceIds,Values=i-12344d305eEXAMPLE \
  --task-arn arn:aws:lambda:us-east-1:111222333444:function:SSMTestLAMBDA \
  --service-role-arn arn:aws:iam::111222333444:role/SSM \
  --task-type LAMBDA \
  --task-invocation-parameters '{"Lambda":{"Payload":{"InstanceId":
  \{{RESOURCE_ID}}\","targetType\":"{{TARGET_TYPE}}\"},"Qualifier":"$LATEST"}}' \
  --priority 0 \
  --max-concurrency 10 \
  --max-errors 5 \
  --name "Lambda_Example" \
```

```
--description "My Lambda Example"
```

输出：

```
{
  "WindowTaskId": "22244444-5555-6666-7777-88888888"
}
```

有关更多信息，请参阅 [《S AWS systems Manager 用户指南》](#) 中的 [在维护时段 \(AWS CLI\) 注册任务](#)。

示例 3：向维护时段注册 Run Command 任务

以下 `register-task-with-maintenance-window` 示例向针对实例的维护时段注册 Run Command 任务。

```
aws ssm register-task-with-maintenance-window \
  --window-id "mw-082dcd7649dee04e4" \
  --targets "Key=InstanceIds,Values=i-12344d305eEXAMPLE" \
  --service-role-arn "arn:aws:iam::111222333444:role/SSM" \
  --task-type "RUN_COMMAND" \
  --name "SSMInstallPowerShellModule" \
  --task-arn "AWS-InstallPowerShellModule" \
  --task-invocation-parameters "{\"RunCommand\":{\"Comment\":"\"\",
  \"OutputS3BucketName\":"\"runcommandlogs\"\", \"Parameters\":{\"commands\":[\"Get-
  Module -ListAvailable\"], \"executionTimeout\":[\"3600\"], \"source\":[\"https://
  gallery.technet.microsoft.com/EZ0ut-33ae0fb7/file/110351/1/EZ0ut.zip\"],
  \"workingDirectory\":[\"\\\\\\\\\\\\\\\\\"]}, \"TimeoutSeconds\":600}}" \
  --max-concurrency 1 \
  --max-errors 1 \
  --priority 10
```

输出：

```
{
  "WindowTaskId": "33344444-5555-6666-7777-88888888"
}
```

有关更多信息，请参阅 [《S AWS systems Manager 用户指南》](#) 中的 [在维护时段 \(AWS CLI\) 注册任务](#)。

示例 4：向维护时段注册 Step Functions 任务

以下 `register-task-with-maintenance-window` 示例向针对实例的维护时段注册 Step Functions 任务。

```
aws ssm register-task-with-maintenance-window \
  --window-id "mw-1234d787d6EXAMPLE" \
  --targets Key=WindowTargetIds,Values=12347414-69c3-49f8-95b8-ed2dcEXAMPLE \
  --task-arn arn:aws:states:us-
east-1:111222333444:stateMachine:SSMTestStateMachine \
  --service-role-arn arn:aws:iam::111222333444:role/MaintenanceWindows \
  --task-type STEP_FUNCTIONS \
  --task-invocation-parameters '{"StepFunctions":{"Input":{"\"InstanceId\":
\"{{RESOURCE_ID}}\"}}}' \
  --priority 0 \
  --max-concurrency 10 \
  --max-errors 5 \
  --name "Step_Functions_Example" \
  --description "My Step Functions Example"
```

输出：

```
{
  "WindowTaskId":"444444444-5555-6666-7777-88888888"
}
```

有关更多信息，请参阅 [《S AWS systems Manager 用户指南》](#) 中的在维护时段 (AWS CLI) 注册任务。

示例 5：使用维护时段目标 ID 注册任务

以下 `register-task-with-maintenance-window` 示例使用维护时段目标 ID 注册任务。维护时段目标 ID 位于 `aws ssm register-target-with-maintenance-window` 命令的输出中。您也可以从 `aws ssm describe-maintenance-window-targets` 命令输出中进行检索。

```
aws ssm register-task-with-maintenance-window \
  --targets "Key=WindowTargetIds,Values=350d44e6-28cc-44e2-951f-4b2c9EXAMPLE" \
  --task-arn "AWS-RunShellScript" \
  --service-role-arn "arn:aws:iam::111222333444:role/MaintenanceWindowsRole" \
  --window-id "mw-ab12cd34eEXAMPLE" \
  --task-type "RUN_COMMAND" \
  --task-parameters "{\"commands\":{\"Values\":[\"df\"]}}" \
  --max-concurrency 1 \
  --max-errors 1 \
```

```
--priority 10
```

输出：

```
{
  "WindowTaskId": "33344444-5555-6666-7777-88888888"
}
```

有关更多信息，请参阅 [《S AWS systems Manager 用户指南》](#) 中的 [在维护时段 \(AWS CLI\) 注册任务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RegisterTaskWithMaintenanceWindow](#) 中的。

## remove-tags-from-resource

以下代码示例演示了如何使用 `remove-tags-from-resource`。

AWS CLI

从补丁基准删除标签

以下 `remove-tags-from-resource` 示例将从补丁基准中删除标签。

```
aws ssm remove-tags-from-resource \
  --resource-type "PatchBaseline" \
  --resource-id "pb-0123456789abcdef0" \
  --tag-keys "Region"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS 一般参考》中的“为 [AWS 资源添加标签](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [RemoveTagsFromResource](#) 中的。

## reset-service-setting

以下代码示例演示了如何使用 `reset-service-setting`。

AWS CLI

重置参数存储吞吐量的服务设置

以下 `reset-service-setting` 示例将指定区域中参数存储吞吐量的服务设置重置为不再使用增加的吞吐量。

```
aws ssm reset-service-setting \  
  --setting-id arn:aws:ssm:us-east-1:123456789012:servicesetting/ssm/parameter-  
store/high-throughput-enabled
```

输出：

```
{  
  "ServiceSetting": {  
    "SettingId": "/ssm/parameter-store/high-throughput-enabled",  
    "SettingValue": "false",  
    "LastModifiedDate": 1555532818.578,  
    "LastModifiedUser": "System",  
    "ARN": "arn:aws:ssm:us-east-1:123456789012:servicesetting/ssm/parameter-  
store/high-throughput-enabled",  
    "Status": "Default"  
  }  
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的“[提高参数存储吞吐量](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ResetServiceSetting](#) 中的。

## resume-session

以下代码示例演示了如何使用 `resume-session`。

### AWS CLI

#### 恢复会话管理器会话

此 `resume-session` 示例在实例断开连接后恢复与该实例的会话管理器会话。请注意，此交互式命令要求在发出呼叫的客户端计算机上安装会话管理器插件。

```
aws ssm resume-session \  
  --session-id Mary-Major-07a16060613c408b5
```

输出：

```
{
  "SessionId": "Mary-Major-07a16060613c408b5",
  "TokenValue":
    "AAEAAVbTGsa0nyvcUoNGqifbv5r/8lgxuQljCuY8qVcv0noBAAAAFxtD3jIXAFUUXGTJ7zF/
    AWJpWdvi0lF5p3dlAgrqVIV06IEXhkHLz0/1gXKRKEME71E6TL0p1LDJAMZ
    +kREejkZu4c5AxMkrQjMF+gtHP1bYJKTwTHQd1wju1PLex08SH17g5R/
    wekrj6WsDUpnEegFBfGftpAIz2GXQVfTJXKfkc5qepQ11C11D0IT2doz0qXgHwfQHfAKLErM5dWDZqKwyT1Z3iw7unQd
    +ihfGa6MEJJ97Jmat/a2TspEn0jNn9Mvu5iwXIW2yCvWZrGUj+/
    QI5Xr7s1XJBEsKR54o4fN0GV9RWl0RZsZm1m1ki0JJtiwwgZ",
  "StreamUrl": "wss://ssmmessages.us-east-2.amazonaws.com/v1/data-channel/Mary-
    Major-07a16060613c408b5?role=publish_subscribe"
}
```

有关更多信息，请参阅 [《Syst AWS ems Manager 用户指南》](#) 中的 [AWS CLI 安装会话管理器插件](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 ResumeSession](#) 中的。

## send-automation-signal

以下代码示例演示了如何使用 send-automation-signal。

### AWS CLI

向自动化执行发送信号

以下 send-automation-signal 示例向自动化执行发送批准信号。

```
aws ssm send-automation-signal \
  --automation-execution-id 73c8eef8-f4ee-4a05-820c-e354fEXAMPLE \
  --signal-type "Approve"
```

此命令不生成任何输出。

有关更多信息，请参阅 [S AWS systems Manager 用户指南](#) 中的 [使用批准者运行自动化工作流程](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 SendAutomationSignal](#) 中的。

## send-command

以下代码示例演示了如何使用 send-command。

## AWS CLI

示例 1：在一个或多个远程实例上运行命令

以下 `send-command` 示例在目标实例上运行 `echo` 命令。

```
aws ssm send-command \  
  --document-name "AWS-RunShellScript" \  
  --parameters 'commands=["echo HelloWorld"]' \  
  --targets "Key=instanceids,Values=i-1234567890abcdef0" \  
  --comment "echo HelloWorld"
```

输出：

```
{  
  "Command": {  
    "CommandId": "92853adf-ba41-4cd6-9a88-142d1EXAMPLE",  
    "DocumentName": "AWS-RunShellScript",  
    "DocumentVersion": "",  
    "Comment": "echo HelloWorld",  
    "ExpiresAfter": 1550181014.717,  
    "Parameters": {  
      "commands": [  
        "echo HelloWorld"  
      ]  
    },  
    "InstanceIds": [  
      "i-0f00f008a2dcbef2"  
    ],  
    "Targets": [],  
    "RequestedDateTime": 1550173814.717,  
    "Status": "Pending",  
    "StatusDetails": "Pending",  
    "OutputS3BucketName": "",  
    "OutputS3KeyPrefix": "",  
    "MaxConcurrency": "50",  
    "MaxErrors": "0",  
    "TargetCount": 1,  
    "CompletedCount": 0,  
    "ErrorCount": 0,  
    "DeliveryTimedOutCount": 0,  
    "ServiceRole": "",  
    "NotificationConfig": {
```



```
        "NotificationArn": "",
        "NotificationEvents": [],
        "NotificationType": ""
    },
    "CloudWatchOutputConfig": {
        "CloudWatchLogGroupName": "",
        "CloudWatchOutputEnabled": false
    }
}
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Systems Manager Run Command 运行命令](#)。

示例 2：获取有关实例的 IP 信息

以下 send-command 示例检索关于实例的 IP 信息。

```
aws ssm send-command \
  --instance-ids "i-1234567890abcdef0" \
  --document-name "AWS-RunShellScript" \
  --comment "IP config" \
  --parameters "commands=ifconfig"
```

有关输出示例，请参阅示例 1。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Systems Manager Run Command 运行命令](#)。

示例 3：在具有特定标签的实例上运行命令

以下 send-command 示例在标签键为“ENV”且值为“Dev”的实例上运行命令。

```
aws ssm send-command \
  --targets "Key=tag:ENV,Values=Dev" \
  --document-name "AWS-RunShellScript" \
  --parameters "commands=ifconfig"
```

有关输出示例，请参阅示例 1。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Systems Manager Run Command 运行命令](#)。

#### 示例 4：运行发送 SNS 通知的命令

以下 send-command 示例运行一条命令，发送所有通知事件和 Command 通知类型的 SNS 通知。

```
aws ssm send-command \  
  --instance-ids "i-1234567890abcdef0" \  
  --document-name "AWS-RunShellScript" \  
  --comment "IP config" \  
  --parameters "commands=ifconfig" \  
  --service-role-arn "arn:aws:iam::123456789012:role/SNS_Role" \  
  --notification-config "NotificationArn=arn:aws:sns:us-  
east-1:123456789012:SNSTopicName,NotificationEvents=All,NotificationType=Command"
```

有关输出示例，请参阅示例 1。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Systems Manager Run Command 运行命令](#)。

#### 示例 5：运行输出到 S3 的命令和 CloudWatch

以下 send-command 示例运行一个命令，该命令将命令详细信息输出到 S3 存储桶和 CloudWatch 日志组。

```
aws ssm send-command \  
  --instance-ids "i-1234567890abcdef0" \  
  --document-name "AWS-RunShellScript" \  
  --comment "IP config" \  
  --parameters "commands=ifconfig" \  
  --output-s3-bucket-name "s3-bucket-name" \  
  --output-s3-key-prefix "runcommand" \  
  --cloud-watch-output-config  
  "CloudWatchOutputEnabled=true,CloudWatchLogGroupName=CWLGroupName"
```

有关输出示例，请参阅示例 1。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Systems Manager Run Command 运行命令](#)。

#### 示例 6：在具有不同标签的多个实例上运行命令

以下 send-command 示例对具有两个不同标签键和值的实例运行命令。

```
aws ssm send-command \  
  --document-name "AWS-RunPowerShellScript" \  
  --parameters commands=["echo helloWorld"] \  
  --targets Key=tag:Env,Values=Dev Key=tag:Role,Values=WebServers
```

有关输出示例，请参阅示例 1。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Systems Manager Run Command 运行命令](#)。

示例 7：将具有相同标签键的多个实例设为目标

以下 send-command 示例在具有相同标签键但不同值的实例上运行命令。

```
aws ssm send-command \  
  --document-name "AWS-RunPowerShellScript" \  
  --parameters commands=["echo helloWorld"] \  
  --targets Key=tag:Env,Values=Dev,Test
```

有关输出示例，请参阅示例 1。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用 Systems Manager Run Command 运行命令](#)。

示例 8：运行使用共享文档的命令

以下 send-command 示例在目标实例上运行共享文档。

```
aws ssm send-command \  
  --document-name "arn:aws:ssm:us-east-1:123456789012:document/ExampleDocument" \  
  --targets "Key=instanceids,Values=i-1234567890abcdef0"
```

有关输出示例，请参阅示例 1。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用共享 SSM 文档](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[SendCommand](#)中的。

## start-associations-once

以下代码示例演示了如何使用 start-associations-once。

## AWS CLI

立即运行一个协会，并且只运行一次

以下 `start-associations-once` 示例立即运行指定的关联，且仅运行一次。如果此命令成功，则无任何输出。

```
aws ssm start-associations-once \  
  --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[查看关联历史记录](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartAssociationsOnce](#) 中的。

## start-automation-execution

以下代码示例演示了如何使用 `start-automation-execution`。

### AWS CLI

示例 1：执行自动化文档

以下 `start-automation-execution` 示例运行自动化文档。

```
aws ssm start-automation-execution \  
  --document-name "AWS-UpdateLinuxAmi" \  
  --parameters "AutomationAssumeRole=arn:aws:iam::123456789012:role/  
SSMAutomationRole,SourceAmiId=ami-EXAMPLE,IamInstanceProfileName=EC2InstanceRole"
```

输出：

```
{  
  "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0a1b2EXAMPLE"  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[手动运行自动化工作流程](#)。

示例 2：运行共享自动化文档

以下 `start-automation-execution` 示例运行一个共享的自动化文档。

```
aws ssm start-automation-execution \  
  --document-name "arn:aws:ssm:us-east-1:123456789012:document/ExampleDocument"
```

输出：

```
{  
  "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0a1b2EXAMPLE"  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[使用共享 SSM 文档](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[StartAutomationExecution](#)中的。

## start-change-request-execution

以下代码示例演示了如何使用 `start-change-request-execution`。

### AWS CLI

示例 1：启动变更请求

以下 `start-change-request-execution` 示例使用最少的指定选项启动变更请求。

```
aws ssm start-change-request-execution \  
  --change-request-name MyChangeRequest \  
  --document-name AWS-HelloWorldChangeTemplate \  
  --runbooks '[{"DocumentName": "AWS-HelloWorld", "Parameters":  
  {"AutomationAssumeRole": ["arn:aws:iam:us-east-2:1112223233444:role/  
MyChangeManagerAssumeRole"]}]'] \  
  --parameters  
  Approver="JohnDoe", ApproverType="IamUser", ApproverSnsTopicArn="arn:aws:sns:us-  
east-2:1112223233444:MyNotificationTopic"
```

输出：

```
{  
  "AutomationExecutionId": "9d32a4fc-f944-11e6-4105-0a1b2EXAMPLE"  
}
```

## 示例 2：使用外部 JSON 文件启动变更请求

以下 `start-automation-execution` 示例使用在 JSON 文件中指定的多个选项启动变更请求。

```
aws ssm start-change-request-execution \  
  --cli-input-json file://MyChangeRequest.json
```

`MyChangeRequest.json` 的内容：

```
{  
  "ChangeRequestName": "MyChangeRequest",  
  "DocumentName": "AWS-HelloWorldChangeTemplate",  
  "DocumentVersion": "$DEFAULT",  
  "ScheduledTime": "2021-12-30T03:00:00",  
  "ScheduledEndTime": "2021-12-30T03:05:00",  
  "Tags": [  
    {  
      "Key": "Purpose",  
      "Value": "Testing"  
    }  
  ],  
  "Parameters": {  
    "Approver": [  
      "JohnDoe"  
    ],  
    "ApproverType": [  
      "IamUser"  
    ],  
    "ApproverSnsTopicArn": [  
      "arn:aws:sns:us-east-2:111222333444:MyNotificationTopic"  
    ]  
  },  
  "Runbooks": [  
    {  
      "DocumentName": "AWS-HelloWorld",  
      "DocumentVersion": "1",  
      "MaxConcurrency": "1",  
      "MaxErrors": "1",  
      "Parameters": {  
        "AutomationAssumeRole": [  
          "arn:aws:iam::111222333444:role/MyChangeManagerAssumeRole"  
        ]  
      }  
    }  
  ]  
}
```

```
    }
  ],
  "ChangeDetails": "### Document Name: HelloWorldChangeTemplate\n\n## What does this document do?\nThis change template demonstrates the feature set available for creating change templates for Change Manager. This template starts a Runbook workflow for the Automation document called AWS-HelloWorld.\n\n## Input Parameters\n\n* ApproverSnsTopicArn: (Required) Amazon Simple Notification Service ARN for approvers.\n* Approver: (Required) The name of the approver to send this request to.\n* ApproverType: (Required) The type of reviewer.\n  * Allowed Values: IamUser, IamGroup, IamRole, SSOGroup, SSOUser\n\n\n## Output Parameters\n\nThis document has no outputs \n"
```

输出：

```
{
  "AutomationExecutionId": "9d32a4fc-f944-11e6-4105-0a1b2EXAMPLE"
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[创建变更请求](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartChangeRequestExecution](#)中的。

## start-session

以下代码示例演示了如何使用 start-session。

### AWS CLI

#### 示例 1：启动会话管理器会话

此start-session示例与会话管理器会话的实例建立连接。请注意，此交互式命令要求在进行呼叫的客户端计算机上安装会话管理器插件。

```
aws ssm start-session \
  --target "i-1234567890abcdef0"
```

输出：

```
Starting session with SessionId: Jane-Roe-07a16060613c408b5
```

## 示例 2：使用 SSH 启动会话管理器会话

此 `start-session` 示例使用 SSH 与会话管理器会话的实例建立连接。请注意，此交互式命令要求在调用客户端计算机上安装会话管理器插件，并且该命令在实例上使用默认用户，`ec2-user` 例如 Linux 的 EC2 实例。

```
ssh -i /path/my-key-pair.pem ec2-user@i-02573cafcfEXAMPLE
```

输出：

```
Starting session with SessionId: ec2-user-07a16060613c408b5
```

有关更多信息，请参阅 [System Manager 用户指南中的启动会话](#) 并安装 AWS CLI 的会话管理器 [插件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StartSession](#) 中的。

## stop-automation-execution

以下代码示例演示了如何使用 `stop-automation-execution`。

### AWS CLI

#### 停止自动化执行

以下 `stop-automation-execution` 示例停止自动化文档。

```
aws ssm stop-automation-execution
  --automation-execution-id "4105a4fc-f944-11e6-9d32-0a1b2EXAMPLE"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的 [手动运行自动化工作流程](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [StopAutomationExecution](#) 中的。

## terminate-session

以下代码示例演示了如何使用 `terminate-session`。



## AWS CLI

### 结束会话管理器会话

此 `terminate-session` 示例永久结束用户 “Shirley-Rodriguez” 创建的会话，并关闭会话管理器客户端与实例上的 SSM 代理之间的数据连接。

```
aws ssm terminate-session \  
  --session-id "Shirley-Rodriguez-07a16060613c408b5"
```

输出：

```
{  
  "SessionId": "Shirley-Rodriguez-07a16060613c408b5"  
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[终止会话](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TerminateSession](#) 中的。

## unlabel-parameter-version

以下代码示例演示了如何使用 `unlabel-parameter-version`。

### AWS CLI

#### 删除参数标签

以下 `unlabel-parameter-version` 示例从给定参数版本中删除指定的标签。

```
aws ssm unlabel-parameter-version \  
  --name "parameterName" \  
  --parameter-version "version" \  
  --labels "label_1" "label_2" "label_3"
```

输出：

```
{  
  "RemovedLabels": [  
    "label_1"  
    "label_2"  
  ]  
}
```

```
    "label_3"  
  ],  
  "InvalidLabels": []  
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[删除参数标签 \(AWS CLI\)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UnlabelParameterVersion](#)中的。

## update-association-status

以下代码示例演示了如何使用 update-association-status。

### AWS CLI

#### 更新关联状态

以下 update-association-status 示例更新了实例和文档之间关联的关联状态。

```
aws ssm update-association-status \  
  --name "AWS-UpdateSSMAgent" \  
  --instance-id "i-1234567890abcdef0" \  
  --association-status  
  "Date=1424421071.939,Name=Pending,Message=temp_status_change,AdditionalInfo=Additional-  
  Config-Needed"
```

#### 输出：

```
{  
  "AssociationDescription": {  
    "Name": "AWS-UpdateSSMAgent",  
    "InstanceId": "i-1234567890abcdef0",  
    "AssociationVersion": "1",  
    "Date": 1550507529.604,  
    "LastUpdateAssociationDate": 1550507806.974,  
    "Status": {  
      "Date": 1424421071.0,  
      "Name": "Pending",  
      "Message": "temp_status_change",  
      "AdditionalInfo": "Additional-Config-Needed"  
    },  
    "Overview": {
```

```
    "Status": "Success",
    "AssociationStatusAggregatedCount": {
      "Success": 1
    }
  },
  "DocumentVersion": "$DEFAULT",
  "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
  "Targets": [
    {
      "Key": "InstanceIds",
      "Values": [
        "i-1234567890abcdef0"
      ]
    }
  ],
  "LastExecutionDate": 1550507808.0,
  "LastSuccessfulExecutionDate": 1550507808.0
}
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[在 Systems Manager 中使用关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateAssociationStatus](#)中的。

## update-association

以下代码示例演示了如何使用 update-association。

### AWS CLI

#### 示例 1：更新文档关联

以下 update-association 示例使用新文档版本更新关联。

```
aws ssm update-association \  
  --association-id "8dfe3659-4309-493a-8755-0123456789ab" \  
  --document-version "\$LATEST"
```

输出：

```
{
```

```

"AssociationDescription": {
  "Name": "AWS-UpdateSSMAgent",
  "AssociationVersion": "2",
  "Date": 1550508093.293,
  "LastUpdateAssociationDate": 1550508106.596,
  "Overview": {
    "Status": "Pending",
    "DetailedStatus": "Creating"
  },
  "DocumentVersion": "$LATEST",
  "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
  "Targets": [
    {
      "Key": "tag:Name",
      "Values": [
        "Linux"
      ]
    }
  ],
  "LastExecutionDate": 1550508094.879,
  "LastSuccessfulExecutionDate": 1550508094.879
}
}

```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[编辑和创建关联的新版本](#)。

## 示例 2：更新关联的计划表达式

以下 `update-association` 示例更新了指定关联的计划表达式。

```

aws ssm update-association \
  --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
  --schedule-expression "cron(0 0 0/4 1/1 * ? *)"

```

输出：

```

{
  "AssociationDescription": {
    "Name": "AWS-HelloWorld",
    "AssociationVersion": "2",
    "Date": "2021-02-08T13:54:19.203000-08:00",
    "LastUpdateAssociationDate": "2021-06-29T11:51:07.933000-07:00",

```

```

    "Overview": {
      "Status": "Pending",
      "DetailedStatus": "Creating"
    },
    "DocumentVersion": "$DEFAULT",
    "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
    "Targets": [
      {
        "Key": "aws:NoOpAutomationTag",
        "Values": [
          "AWS-NoOpAutomationTarget-Value"
        ]
      }
    ],
    "ScheduleExpression": "cron(0 0 0/4 1/1 * ? *)",
    "LastExecutionDate": "2021-06-26T19:00:48.110000-07:00",
    "ApplyOnlyAtCronInterval": false
  }
}

```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[编辑和创建关联的新版本](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateAssociation](#) 中的。

## update-document-default-version

以下代码示例演示了如何使用 `update-document-default-version`。

### AWS CLI

#### 更新文档的默认版本

以下 `update-document-default-version` 示例更新了 Systems Manager 文档的默认版本。

```

aws ssm update-document-default-version \
  --name "Example" \
  --document-version "2"

```

#### 输出：

```

{
  "Description": {

```

```
    "Name": "Example",  
    "DefaultVersion": "2"  
  }  
}
```

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[编写 SSM 文档内容](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateDocumentDefaultVersion](#) 中的。

## update-document-metadata

以下代码示例演示了如何使用 update-document-metadata。

### AWS CLI

示例：批准最新版本的更改模板

以下内容 update-document-metadata 提供了对已提交审核的最新版本变更模板的批准。

```
aws ssm update-document-metadata \  
  --name MyChangeManagerTemplate \  
  --document-reviews 'Action=Approve,Comment=[{Type=Comment,Content=Approved!}]'
```

此命令不生成任何输出。

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[审阅、批准或拒绝变更模板](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateDocumentMetadata](#) 中的。

## update-document

以下代码示例演示了如何使用 update-document。

### AWS CLI

创建文档的新版本

以下 update-document 示例在 Windows 计算机上运行时创建文档的新版本。--document 指定的文档必须采用 JSON 格式。请注意，必须先引用 file://，后跟内容文件的路径。由于 --document-version 参数的开头有 \$，因此在 Windows 上，必须用双引号将该值括起来。在 Linux、macOS 上或 PowerShell 出现提示时，必须用单引号将值括起来。

## Windows 版本：

```
aws ssm update-document \  
  --name "RunShellScript" \  
  --content "file://RunShellScript.json" \  
  --document-version "$LATEST"
```

## Linux/Mac 版本：

```
aws ssm update-document \  
  --name "RunShellScript" \  
  --content "file://RunShellScript.json" \  
  --document-version '$LATEST'
```

## 输出：

```
{  
  "DocumentDescription": {  
    "Status": "Updating",  
    "Hash": "f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b",  
    "Name": "RunShellScript",  
    "Parameters": [  
      {  
        "Type": "StringList",  
        "Name": "commands",  
        "Description": "(Required) Specify a shell script or a command to  
run."  
      }  
    ],  
    "DocumentType": "Command",  
    "PlatformTypes": [  
      "Linux"  
    ],  
    "DocumentVersion": "2",  
    "HashType": "Sha256",  
    "CreateDate": 1487899655.152,  
    "Owner": "809632081692",  
    "SchemaVersion": "2.0",  
    "DefaultVersion": "1",  
    "LatestVersion": "2",  
    "Description": "Run an updated script"  
  }  
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDocument](#)中的。

## update-maintenance-window-target

以下代码示例演示了如何使用 update-maintenance-window-target。

### AWS CLI

#### 更新维护时段目标

以下update-maintenance-window-target示例仅更新维护时段目标的名称。

```
aws ssm update-maintenance-window-target \  
  --window-id "mw-0c5ed765acEXAMPLE" \  
  --window-target-id "57e8344e-fe64-4023-8191-6bf05EXAMPLE" \  
  --name "NewName" \  
  --no-replace
```

输出：

```
{  
  "Description": "",  
  "OwnerInformation": "",  
  "WindowTargetId": "57e8344e-fe64-4023-8191-6bf05EXAMPLE",  
  "WindowId": "mw-0c5ed765acEXAMPLE",  
  "Targets": [  
    {  
      "Values": [  
        "i-1234567890EXAMPLE"  
      ],  
      "Key": "InstanceIds"  
    }  
  ],  
  "Name": "NewName"  
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[“更新维护窗口 \(AWS CLI\)”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateMaintenanceWindowTarget](#)中的。



## update-maintenance-window-task

以下代码示例演示了如何使用 update-maintenance-window-task。

### AWS CLI

#### 更新维护时段任务

以下update-maintenance-window-task示例更新了维护时段任务的服务角色。

```
aws ssm update-maintenance-window-task \  
  --window-id "mw-0c5ed765acEXAMPLE" \  
  --window-task-id "23d3809e-9fbe-4ddf-b41a-b49d7EXAMPLE" \  
  --service-role-arn "arn:aws:iam::111222333444:role/aws-service-role/  
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM"
```

输出：

```
{  
  "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/  
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",  
  "MaxErrors": "1",  
  "TaskArn": "AWS-UpdateEC2Config",  
  "MaxConcurrency": "1",  
  "WindowTaskId": "23d3809e-9fbe-4ddf-b41a-b49d7EXAMPLE",  
  "TaskParameters": {},  
  "Priority": 1,  
  "TaskInvocationParameters": {  
    "RunCommand": {  
      "TimeoutSeconds": 600,  
      "Parameters": {  
        "allowDowngrade": [  
          "false"  
        ]  
      }  
    }  
  },  
  "WindowId": "mw-0c5ed765acEXAMPLE",  
  "Description": "UpdateEC2Config",  
  "Targets": [  
    {  
      "Values": [  
        "57e8344e-fe64-4023-8191-6bf05EXAMPLE"  
      ]  
    }  
  ]  
}
```

```
    ],
    "Key": "WindowTargetIds"
  }
],
"Name": "UpdateEC2Config"
}
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的“[更新维护窗口 \(AWS CLI\)](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateMaintenanceWindowTask](#)中的。

## update-maintenance-window

以下代码示例演示了如何使用 update-maintenance-window。

### AWS CLI

#### 示例 1：更新维护时段

以下 update-maintenance-window 示例更新了维护时段的名称。

```
aws ssm update-maintenance-window \
  --window-id "mw-1a2b3c4d5e6f7g8h9" \
  --name "My-Renamed-MW"
```

输出：

```
{
  "Cutoff": 1,
  "Name": "My-Renamed-MW",
  "Schedule": "cron(0 16 ? * TUE *)",
  "Enabled": true,
  "AllowUnassociatedTargets": true,
  "WindowId": "mw-1a2b3c4d5e6f7g8h9",
  "Duration": 4
}
```

#### 示例 2：禁用维护时段

以下 update-maintenance-window 示例禁用维护时段。

```
aws ssm update-maintenance-window \
```

```
--window-id "mw-1a2b3c4d5e6f7g8h9" \  
--no-enabled
```

### 示例 3：启用维护时段

以下 `update-maintenance-window` 示例启用维护时段。

```
aws ssm update-maintenance-window \  
  --window-id "mw-1a2b3c4d5e6f7g8h9" \  
  --enabled
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》中的[“更新维护窗口 \(AWS CLI\)”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateMaintenanceWindow](#)中的。

## update-managed-instance-role

以下代码示例演示了如何使用 `update-managed-instance-role`。

### AWS CLI

更新托管式实例的 IAM 角色

以下 `update-managed-instance-role` 示例更新了托管式实例的 IAM 实例配置文件。

```
aws ssm update-managed-instance-role \  
  --instance-id "mi-08ab247cdfEXAMPLE" \  
  --iam-role "ExampleRole"
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS Systems Manager 用户指南》中的[步骤 4：为 Systems Manager 创建 IAM 实例配置文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateManagedInstanceRole](#)中的。

## update-ops-item

以下代码示例演示了如何使用 `update-ops-item`。

### AWS CLI

要更新 OpsItem

以下 `update-ops-item` 示例更新了描述、优先级和类别 `OpsItem`。此外，该命令还指定一个 SNS 主题，当编辑或更改该主题时，通知将发送到 `OpsItem` 该主题。

```
aws ssm update-ops-item \  
  --ops-item-id "oi-287b5EXAMPLE" \  
  --description "Primary OpsItem for failover event 2020-01-01-fh398yf" \  
  --priority 2 \  
  --category "Security" \  
  --notifications "Arn=arn:aws:sns:us-east-2:111222333444:my-us-east-2-topic"
```

输出：

```
This command produces no output.
```

有关更多信息，请参阅《S AWS systems Manager 用户指南》[OpsItems](#)中的“使用”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateOpsItem](#)中的。

## update-patch-baseline

以下代码示例演示了如何使用 `update-patch-baseline`。

### AWS CLI

#### 示例 1：更新补丁基准

以下 `update-patch-baseline` 示例将指定的两个补丁（作为已拒绝的补丁）和一个补丁（作为已批准的补丁）添加到指定的补丁基准。

```
aws ssm update-patch-baseline \  
  --baseline-id "pb-0123456789abcdef0" \  
  --rejected-patches "KB2032276" "MS10-048" \  
  --approved-patches "KB2124261"
```

输出：

```
{  
  "BaselineId": "pb-0123456789abcdef0",  
  "Name": "WindowsPatching",  
  "OperatingSystem": "WINDOWS",  
  "GlobalFilters": {  
    "PatchFilters": []  
  }  
}
```

```

},
"ApprovalRules": {
  "PatchRules": [
    {
      "PatchFilterGroup": {
        "PatchFilters": [
          {
            "Key": "PRODUCT",
            "Values": [
              "WindowsServer2016"
            ]
          }
        ]
      },
      "ComplianceLevel": "CRITICAL",
      "ApproveAfterDays": 0,
      "EnableNonSecurity": false
    }
  ]
},
"ApprovedPatches": [
  "KB2124261"
],
"ApprovedPatchesComplianceLevel": "UNSPECIFIED",
"ApprovedPatchesEnableNonSecurity": false,
"RejectedPatches": [
  "KB2032276",
  "MS10-048"
],
"RejectedPatchesAction": "ALLOW_AS_DEPENDENCY",
"CreateDate": 1550244180.465,
"ModifiedDate": 1550244180.465,
"Description": "Patches for Windows Servers",
"Sources": []
}

```

## 示例 2：重命名补丁基准

以下 `update-patch-baseline` 示例重命名指定的补丁基准。

```

aws ssm update-patch-baseline \
  --baseline-id "pb-0713accee01234567" \
  --name "Windows-Server-2012-R2-Important-and-Critical-Security-Updates"

```

有关更多信息，请参阅 Systems Manager 用户指南中的更新或删除补丁基准 < <https://docs.aws.amazon.com/systems-manager/latest/userguide/patch-baseline-update-or-delete.html>>`\_。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePatchBaseline](#)中的。

## update-resource-data-sync

以下代码示例演示了如何使用 update-resource-data-sync。

### AWS CLI

#### 更新资源数据同步

以下update-resource-data-sync示例更新 SyncFromSource 资源数据同步。

```
aws ssm update-resource-data-sync \  
  --sync-name exampleSync \  
  --sync-type SyncFromSource \  
  --sync-source '{"SourceType":"SingleAccountMultiRegions", "SourceRegions":["us-east-1", "us-west-2"]}'
```

此命令不生成任何输出。

有关更多信息，请参阅《[Systems Manager 用户指南](#)》中的“[将 Systems Manager Explorer 设置为显示来自多个账户和地区的数据](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateResourceDataSync](#)中的。

## update-service-setting

以下代码示例演示了如何使用 update-service-setting。

### AWS CLI

#### 更新参数存储吞吐量的服务设置

以下update-service-setting示例更新了指定区域中参数存储吞吐量的当前服务设置，以使用更高的吞吐量。

```
aws ssm update-service-setting \  
  --service-setting-name ParameterStorageThroughputLimit
```

```
--setting-id arn:aws:ssm:us-east-1:123456789012:servicesetting/ssm/parameter-  
store/high-throughput-enabled \  
--setting-value true
```

此命令不生成任何输出。

有关更多信息，请参阅《S AWS ystems Manager 用户指南》中的“[提高参数存储吞吐量](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateServiceSetting](#) 中的。

## 使用 Amazon Textract 的示例 AWS CLI

以下代码示例向您展示了如何在 Amazon Textract 中 AWS Command Line Interface 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **analyze-document**

以下代码示例演示了如何使用 analyze-document。

AWS CLI

分析文档中的文本

以下 analyze-document 示例演示如何分析文档中的文本。

Linux/macOS :

```
aws textract analyze-document \  

```

```
--document '{"S3Object":{"Bucket":"bucket","Name":"document"}}' \  
--feature-types ['TABLES','FORMS']
```

## Windows:

```
aws textract analyze-document \  
--document "{\"S3Object\":{\"Bucket\":\"bucket\",\"Name\":\"document\"}}\" \  
--feature-types [\"TABLES\",\"FORMS\"] \  
--region region-name
```

## 输出:

```
{  
  "Blocks": [  
    {  
      "Geometry": {  
        "BoundingBox": {  
          "Width": 1.0,  
          "Top": 0.0,  
          "Left": 0.0,  
          "Height": 1.0  
        },  
        "Polygon": [  
          {  
            "Y": 0.0,  
            "X": 0.0  
          },  
          {  
            "Y": 0.0,  
            "X": 1.0  
          },  
          {  
            "Y": 1.0,  
            "X": 1.0  
          },  
          {  
            "Y": 1.0,  
            "X": 0.0  
          }  
        ]  
      },  
      "Relationships": [  
        {
```



```
        "Type": "CHILD",
        "Ids": [
            "87586964-d50d-43e2-ace5-8a890657b9a0",
            "a1e72126-21d9-44f4-a8d6-5c385f9002ba",
            "e889d012-8a6b-4d2e-b7cd-7a8b327d876a"
        ]
    },
    ],
    "BlockType": "PAGE",
    "Id": "c2227f12-b25d-4e1f-baea-1ee180d926b2"
}
],
"DocumentMetadata": {
    "Pages": 1
}
}
```

有关更多信息，请参阅《Amazon Textract 开发人员指南》中的“使用 Amazon Textract 分析文档文本”

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AnalyzeDocument](#)中的。

## detect-document-text

以下代码示例演示了如何使用 detect-document-text。

### AWS CLI

检测文档中的文本

以下 detect-document-text 示例演示了如何检测文档中的文本。

Linux/macOS :

```
aws textract detect-document-text \
  --document '{"S3Object":{"Bucket":"bucket","Name":"document"}}'
```

Windows:

```
aws textract detect-document-text \
  --document '{"S3Object":{"Bucket":"bucket","Name":"document"}}' \
  --region region-name
```

输出：

```
{
  "Blocks": [
    {
      "Geometry": {
        "BoundingBox": {
          "Width": 1.0,
          "Top": 0.0,
          "Left": 0.0,
          "Height": 1.0
        },
        "Polygon": [
          {
            "Y": 0.0,
            "X": 0.0
          },
          {
            "Y": 0.0,
            "X": 1.0
          },
          {
            "Y": 1.0,
            "X": 1.0
          },
          {
            "Y": 1.0,
            "X": 0.0
          }
        ]
      },
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "896a9f10-9e70-4412-81ce-49ead73ed881",
            "0da18623-dc4c-463d-a3d1-9ac050e9e720",
            "167338d7-d38c-4760-91f1-79a8ec457bb2"
          ]
        }
      ],
      "BlockType": "PAGE",
      "Id": "21f0535e-60d5-4bc7-adf2-c05dd851fa25"
    },
  ],
}
```

```
{
  "Relationships": [
    {
      "Type": "CHILD",
      "Ids": [
        "62490c26-37ea-49fa-8034-7a9ff9369c9c",
        "1e4f3f21-05bd-4da9-ba10-15d01e66604c"
      ]
    }
  ],
  "Confidence": 89.11581420898438,
  "Geometry": {
    "BoundingBox": {
      "Width": 0.33642634749412537,
      "Top": 0.17169663310050964,
      "Left": 0.13885067403316498,
      "Height": 0.49159330129623413
    },
    "Polygon": [
      {
        "Y": 0.17169663310050964,
        "X": 0.13885067403316498
      },
      {
        "Y": 0.17169663310050964,
        "X": 0.47527703642845154
      },
      {
        "Y": 0.6632899641990662,
        "X": 0.47527703642845154
      },
      {
        "Y": 0.6632899641990662,
        "X": 0.13885067403316498
      }
    ]
  },
  "Text": "Hello,",
  "BlockType": "LINE",
  "Id": "896a9f10-9e70-4412-81ce-49ead73ed881"
},
{
  "Relationships": [
    {
```

```
        "Type": "CHILD",
        "Ids": [
            "19b28058-9516-4352-b929-64d7cef29daf"
        ]
    },
],
"Confidence": 85.5694351196289,
"Geometry": {
    "BoundingBox": {
        "Width": 0.33182239532470703,
        "Top": 0.23131252825260162,
        "Left": 0.5091826915740967,
        "Height": 0.3766750991344452
    },
    "Polygon": [
        {
            "Y": 0.23131252825260162,
            "X": 0.5091826915740967
        },
        {
            "Y": 0.23131252825260162,
            "X": 0.8410050868988037
        },
        {
            "Y": 0.607987642288208,
            "X": 0.8410050868988037
        },
        {
            "Y": 0.607987642288208,
            "X": 0.5091826915740967
        }
    ]
},
"Text": "worlc",
"BlockType": "LINE",
"Id": "0da18623-dc4c-463d-a3d1-9ac050e9e720"
}
],
"DocumentMetadata": {
    "Pages": 1
}
}
```

有关更多信息，请参阅《Amazon Textract 开发人员指南》中的“使用 Amazon Textract 检测文档文本”

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetectDocumentText](#)中的。

## get-document-analysis

以下代码示例演示了如何使用 get-document-analysis。

### AWS CLI

获取对多页文档进行异步文本分析的结果

以下 get-document-analysis 示例演示了如何获取对多页文档进行异步文本分析的结果。

```
aws textract get-document-analysis \  
  --job-id df7cf32ebbd2a5de113535fcf4d921926a701b09b4e7d089f3aebadb41e0712b \  
  --max-results 1000
```

输出：

```
{  
  "Blocks": [  
    {  
      "Geometry": {  
        "BoundingBox": {  
          "Width": 1.0,  
          "Top": 0.0,  
          "Left": 0.0,  
          "Height": 1.0  
        },  
        "Polygon": [  
          {  
            "Y": 0.0,  
            "X": 0.0  
          },  
          {  
            "Y": 0.0,  
            "X": 1.0  
          },  
          {  
            "Y": 1.0,  
            "X": 1.0  
          }  
        ]  
      }  
    ]  
  }
```

```

        },
        {
            "Y": 1.0,
            "X": 0.0
        }
    ]
},
"Relationships": [
    {
        "Type": "CHILD",
        "Ids": [
            "75966e64-81c2-4540-9649-d66ec341cd8f",
            "bb099c24-8282-464c-a179-8a9fa0a057f0",
            "5ebf522d-f9e4-4dc7-bfae-a288dc094595"
        ]
    }
],
"BlockType": "PAGE",
"Id": "247c28ee-b63d-4aeb-9af0-5f7ea8ba109e",
"Page": 1
}
],
"NextToken": "cY1W3eTFvoB0cH7YrKVudI4Gb0H8J0xAYLo8xI/JunCIPWCthaKQ+07n/
ElyutsSy0+1V0ImoTRmP1zw4P0RFtaeV9Bzhnfedpx1YqwB4xaGDA==",
"DocumentMetadata": {
    "Pages": 1
},
"JobStatus": "SUCCEEDED"
}

```

有关更多信息，请参阅《Amazon Textract 开发人员指南》中的“检测和分析多页文档中的文本”

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDocumentAnalysis](#) 中的。

## get-document-text-detection

以下代码示例演示了如何使用 `get-document-text-detection`。

### AWS CLI

在文档中获取异步文本检测的结果

以下 `get-document-text-detection` 示例说明如何在多页文档中获取异步文本检测的结果。

```
aws textract get-document-text-detection \  
  --job-id 57849a3dc627d4df74123dca269d69f7b89329c870c65bb16c9fd63409d200b9 \  
  --max-results 1000
```

## 输出

```
{  
  "Blocks": [  
    {  
      "Geometry": {  
        "BoundingBox": {  
          "Width": 1.0,  
          "Top": 0.0,  
          "Left": 0.0,  
          "Height": 1.0  
        },  
        "Polygon": [  
          {  
            "Y": 0.0,  
            "X": 0.0  
          },  
          {  
            "Y": 0.0,  
            "X": 1.0  
          },  
          {  
            "Y": 1.0,  
            "X": 1.0  
          },  
          {  
            "Y": 1.0,  
            "X": 0.0  
          }  
        ]  
      },  
      "Relationships": [  
        {  
          "Type": "CHILD",  
          "Ids": [  
            "1b926a34-0357-407b-ac8f-ec473160c6a9",  
            "0c35dc17-3605-4c9d-af1a-d9451059df51",  
            "dea3db8a-52c2-41c0-b50c-81f66f4aa758"  
          ]  
        }  
      ]  
    }  
  ]  
}
```

```

        }
      ],
      "BlockType": "PAGE",
      "Id": "84671a5e-8c99-43be-a9d1-6838965da33e",
      "Page": 1
    }
  ],
  "NextToken": "GcqyoAJuZwuj0T35EN4LCI3EUzMtiLq3nKyFFHvU5q1SaIdEBcSty+njNgoWwuMP/
muqc96S4o5NzDqehhXvhkodMyV050JGyms51srCxibWJw==",
  "DocumentMetadata": {
    "Pages": 1
  },
  "JobStatus": "SUCCEEDED"
}

```

有关更多信息，请参阅《Amazon Textract 开发人员指南》中的“检测和分析多页文档中的文本”

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetDocumentTextDetection](#) 中的。

## start-document-analysis

以下代码示例演示了如何使用 start-document-analysis。

### AWS CLI

开始分析多页文档中的文本

以下 start-document-analysis 示例演示如何开始异步分析多页文档中的文本。

Linux/macOS :

```

aws textract start-document-analysis \
  --document-location '{"S3Object":{"Bucket":"bucket","Name":"document"}}' \
  --feature-types ["TABLES","FORMS"] \
  --notification-channel "SNSTopicArn=arn:snsTopic,RoleArn=roleArn"

```

Windows:

```

aws textract start-document-analysis \
  --document-location "{\"S3Object\":{\"Bucket\":\"bucket\",\"Name\":\"document\
  \}}\" \
  --feature-types "[\"TABLES\", \"FORMS\"]" \

```



```
--region region-name \  
--notification-channel "SNSTopicArn=arn:snsTopic,RoleArn=roleArn"
```

输出：

```
{  
  "JobId": "df7cf32ebbd2a5de113535fcf4d921926a701b09b4e7d089f3aebadb41e0712b"  
}
```

有关更多信息，请参阅《Amazon Textract 开发人员指南》中的“检测和分析多页文档中的文本”

• 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartDocumentAnalysis](#)中的。

## start-document-text-detection

以下代码示例演示了如何使用 start-document-text-detection。

### AWS CLI

开始检测多页文档中的文本

以下 start-document-text-detection 示例演示如何开始异步检测多页文档中的文本。

Linux/macOS：

```
aws textract start-document-text-detection \  
  --document-location '{"S3Object":{"Bucket":"bucket","Name":"document"}}' \  
  --notification-channel "SNSTopicArn=arn:snsTopic,RoleArn=roleARN"
```

Windows:

```
aws textract start-document-text-detection \  
  --document-location "{\"S3Object\":{\"Bucket\":\"bucket\",\"Name\":\"document  
  \\\"}\" \  
  --region region-name \  
  --notification-channel "SNSTopicArn=arn:snsTopic,RoleArn=roleArn"
```

输出：

```
{  
  "JobId": "57849a3dc627d4df74123dca269d69f7b89329c870c65bb16c9fd63409d200b9"
```

```
}
```

有关更多信息，请参阅《Amazon Textract 开发人员指南》中的“检测和分析多页文档中的文本”

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartDocumentTextDetection](#)中的。

## 使用 Amazon Transcribe 示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon Transcribe 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **create-language-model**

以下代码示例演示了如何使用 create-language-model。

#### AWS CLI

示例 1：使用训练和调整数据创建自定义语言模型。

以下create-language-model示例创建了一个自定义语言模型。您可以使用自定义语言模型来提高法律、酒店、金融和保险等领域的转录性能。对于语言代码，请输入有效的语言代码。对于 base-model-name，请指定最适合您要使用自定义语言模型转录的音频采样率的基本模型。在 model-name 中，指定要调用自定义语言模型的名称。

```
aws transcribe create-language-model \  
  --language-code language-code \  
  --base-model-name base-model-name \  
  --model-name cli-clm-example \  

```

```
--input-data-config S3Uri="s3://DOC-EXAMPLE-BUCKET/Amazon-S3-Prefix-for-training-data",TuningDataS3Uri="s3://DOC-EXAMPLE-BUCKET/Amazon-S3-Prefix-for-tuning-data",DataAccessRoleArn="arn:aws:iam::AWS-account-number:role/IAM-role-with-permissions-to-create-a-custom-language-model"
```

输出：

```
{
  "LanguageCode": "language-code",
  "BaseModelName": "base-model-name",
  "ModelName": "cli-clm-example",
  "InputDataConfig": {
    "S3Uri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-Prefix/",
    "TuningDataS3Uri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-Prefix/",
    "DataAccessRoleArn": "arn:aws:iam::AWS-account-number:role/IAM-role-with-permissions-create-a-custom-language-model"
  },
  "ModelStatus": "IN_PROGRESS"
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[使用自定义语言模型提高特定领域的转录准确性](#)。

示例 2：仅使用训练数据创建自定义语言模型。

以下 `create-language-model` 示例转录音频文件。您可以使用自定义语言模型来提高法律、酒店、金融和保险等领域的转录性能。对于语言代码，请输入有效的语言代码。对于 `base-model-name`，请指定最适合您要使用自定义语言模型转录的音频采样率的基本模型。在 `model-name` 中，指定要调用自定义语言模型的名称。

```
aws transcribe create-language-model \
  --language-code en-US \
  --base-model-name base-model-name \
  --model-name cli-clm-example \
  --input-data-config S3Uri="s3://DOC-EXAMPLE-BUCKET/Amazon-S3-Prefix-For-Training-Data",DataAccessRoleArn="arn:aws:iam::AWS-account-number:role/IAM-role-with-permissions-to-create-a-custom-language-model"
```

输出：

```
{
```

```
"LanguageCode": "en-US",
"BaseModelName": "base-model-name",
"ModelName": "cli-clm-example",
"InputDataConfig": {
  "S3Uri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-Prefix-For-Training-Data/",
  "DataAccessRoleArn": "arn:aws:iam::your-AWS-account-number:role/IAM-role-
with-permissions-to-create-a-custom-language-model"
},
"ModelStatus": "IN_PROGRESS"
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[使用自定义语言模型提高特定领域的转录准确性](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateLanguageModel](#) 中的。

## create-medical-vocabulary

以下代码示例演示了如何使用 create-medical-vocabulary。

### AWS CLI

#### 创建医学自定义词汇表

以下 create-medical-vocabulary 示例创建一个自定义词汇表。要创建自定义词汇表，您必须创建一个文本文件，其中包含要更准确地进行转录的所有术语。对于 vocabulary-file-uri，请指定该文本文件的亚马逊简单存储服务 (Amazon S3) Service URI。对于 language-code，指定与自定义词汇表的语言对应的语言代码。对于 vocabulary-name，指定所需的自定义词汇表名称。

```
aws transcribe create-medical-vocabulary \
  --vocabulary-name cli-medical-vocab-example \
  --language-code language-code \
  --vocabulary-file-uri https://DOC-EXAMPLE-BUCKET.AWS-Region.amazonaws.com/the-
text-file-for-the-medical-custom-vocabulary.txt
```

输出：

```
{
  "VocabularyName": "cli-medical-vocab-example",
  "LanguageCode": "language-code",
  "VocabularyState": "PENDING"
}
```

```
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[医疗自定义词汇表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateMedicalVocabulary](#)中的。

## create-vocabulary-filter

以下代码示例演示了如何使用 create-vocabulary-filter。

### AWS CLI

#### 创建词汇过滤器

以下create-vocabulary-filter示例创建了一个词汇过滤器，该过滤器使用一个文本文件，其中包含您不想在转录中出现的单词列表。对于语言代码，请指定与词汇过滤器的语言相对应的语言代码。对于 vocabulary-filter-file-uri，请指定文本文件的亚马逊简单存储服务 (Amazon S3) Service URI。对于 vocabulary-filter-name，请指定词汇过滤器的名称。

```
aws transcribe create-vocabulary-filter \  
  --language-code language-code \  
  --vocabulary-filter-file-uri s3://DOC-EXAMPLE-BUCKET/vocabulary-filter.txt \  
  --vocabulary-filter-name cli-vocabulary-filter-example
```

输出：

```
{  
  "VocabularyFilterName": "cli-vocabulary-filter-example",  
  "LanguageCode": "language-code"  
}
```

有关更多信息，请参阅《Amazon Transcribe 开发者指南》中的[筛选不需要的单词](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVocabularyFilter](#)中的。

## create-vocabulary

以下代码示例演示了如何使用 create-vocabulary。

### AWS CLI

#### 创建自定义词汇表

以下 `create-vocabulary` 示例创建一个自定义词汇表。要创建自定义词汇表，您必须创建一个文本文件，其中包含要更准确地进行转录的所有术语。对于 `vocabulary-file-uri`，请指定该文本文件的亚马逊简单存储服务 (Amazon S3) Service URI。对于 `language-code`，指定与自定义词汇表的语言对应的语言代码。对于 `vocabulary-name`，指定所需的自定义词汇表名称。

```
aws transcribe create-vocabulary \  
  --language-code language-code \  
  --vocabulary-name cli-vocab-example \  
  --vocabulary-file-uri s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/the-text-file-  
for-the-custom-vocabulary.txt
```

输出：

```
{  
  "VocabularyName": "cli-vocab-example",  
  "LanguageCode": "language-code",  
  "VocabularyState": "PENDING"  
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[自定义词汇表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateVocabulary](#)中的。

## delete-language-model

以下代码示例演示了如何使用 `delete-language-model`。

### AWS CLI

#### 删除自定义语言模型

以下`delete-language-model`示例删除了自定义语言模型。

```
aws transcribe delete-language-model \  
  --model-name model-name
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[使用自定义语言模型提高特定领域的转录准确性](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLanguageModel](#)中的。

## delete-medical-transcription-job

以下代码示例演示了如何使用 delete-medical-transcription-job。

### AWS CLI

删除医疗转录作业

以下 delete-medical-transcription-job 示例删除一个医疗转录作业。

```
aws transcribe delete-medical-transcription-job \  
  --medical-transcription-job-name medical-transcription-job-name
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon T [DeleteMedicalTranscriptionJob](#)ranscribe 开发者指南》。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMedicalTranscriptionJob](#)中的。

## delete-medical-vocabulary

以下代码示例演示了如何使用 delete-medical-vocabulary。

### AWS CLI

删除医疗自定义词汇

以下delete-medical-vocabulary示例删除了医学自定义词汇表。在“词汇名称”中，指定医学自定义词汇的名称。

```
aws transcribe delete-vocabulary \  
  --vocabulary-name medical-custom-vocabulary-name
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[医疗自定义词汇表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMedicalVocabulary](#)中的。

## delete-transcription-job

以下代码示例演示了如何使用 delete-transcription-job。

### AWS CLI

删除一个转录作业

以下 delete-transcription-job 示例删除一个转录作业。

```
aws transcribe delete-transcription-job \  
  --transcription-job-name your-transcription-job
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon T [DeleteTranscriptionJob](#)ranscribe 开发者指南》。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTranscriptionJob](#)中的。

## delete-vocabulary-filter

以下代码示例演示了如何使用 delete-vocabulary-filter。

### AWS CLI

删除词汇过滤器

以下delete-vocabulary-filter示例删除词汇过滤器。

```
aws transcribe delete-vocabulary-filter \  
  --vocabulary-filter-name vocabulary-filter-name
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Transcribe 开发者指南》中的[筛选不需要的单词](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVocabularyFilter](#)中的。

## delete-vocabulary

以下代码示例演示了如何使用 delete-vocabulary。



## AWS CLI

### 删除自定义词汇表

以下 `delete-vocabulary` 示例删除一个自定义词汇表。

```
aws transcribe delete-vocabulary \  
  --vocabulary-name vocabulary-name
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[自定义词汇表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteVocabulary](#)中的。

## describe-language-model

以下代码示例演示了如何使用 `describe-language-model`。

### AWS CLI

获取有关特定自定义语言模型的信息

以下`describe-language-model`示例获取有关特定自定义语言模型的信息。例如，在下方，`BaseModelName` 您可以看到您的模型是否使用 `NarrowBand` 或 `WideBand` 模型进行训练。带有 `NarrowBand` 基本模型的自定义语言模型可以转录采样率低于 16 kHz 的音频。使用 `WideBand` 基础模型的语言模型可以转录采样率大于 16 kHz 的音频。`S3Uri` 参数表示您用于访问训练数据以创建自定义语言模型的 Amazon S3 前缀。

```
aws transcribe describe-language-model \  
  --model-name cli-clm-example
```

输出：

```
{  
  "LanguageModel": {  
    "ModelName": "cli-clm-example",  
    "CreateTime": "2020-09-25T17:57:38.504000+00:00",  
    "LastModifiedTime": "2020-09-25T17:57:48.585000+00:00",  
    "LanguageCode": "language-code",  
    "BaseModelName": "base-model-name",  
    "ModelStatus": "IN_PROGRESS",
```

```

    "UpgradeAvailability": false,
    "InputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-Prefix/",
      "TuningDataS3Uri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-Prefix/",
      "DataAccessRoleArn": "arn:aws:iam::AWS-account-number:role/IAM-role-
with-permissions-to-create-a-custom-language-model"
    }
  }
}

```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[使用自定义语言模型提高特定领域的转录准确性](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DescribeLanguageModel](#)中的。

## get-medical-transcription-job

以下代码示例演示了如何使用 `get-medical-transcription-job`。

### AWS CLI

获取有关特定医疗转录工作的信息

以下 `get-medical-transcription-job` 示例获取有关特定医疗转录作业的信息。要访问转录结果，请使用 `TranscriptFileUri` 参数。如果您为转录作业启用了其他功能，则可以在“设置”对象中看到这些功能。`Specialty` 参数显示提供者的医疗专业知识。`Type` 参数表示转录作业中的语音是医疗对话还是医疗听写。

```

aws transcribe get-medical-transcription-job \
  --medical-transcription-job-name vocabulary-dictation-medical-transcription-job

```

输出：

```

{
  "MedicalTranscriptionJob": {
    "MedicalTranscriptionJobName": "vocabulary-dictation-medical-transcription-
job",
    "TranscriptionJobStatus": "COMPLETED",
    "LanguageCode": "en-US",
    "MediaSampleRateHertz": 48000,
    "MediaFormat": "mp4",
    "Media": {

```

```
    "MediaFileUri": "s3://Amazon-S3-Prefix/your-audio-file.file-extension"
  },
  "Transcript": {
    "TranscriptFileUri": "https://s3.Region.amazonaws.com/Amazon-S3-Prefix/
vocabulary-dictation-medical-transcription-job.json"
  },
  "StartTime": "2020-09-21T21:17:27.045000+00:00",
  "CreationTime": "2020-09-21T21:17:27.016000+00:00",
  "CompletionTime": "2020-09-21T21:17:59.561000+00:00",
  "Settings": {
    "ChannelIdentification": false,
    "ShowAlternatives": false,
    "VocabularyName": "cli-medical-vocab-example"
  },
  "Specialty": "PRIMARYCARE",
  "Type": "DICTATION"
}
}
```

有关更多信息，请参阅 Amazon [Transcribe 开发者指南中的批量转录](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetMedicalTranscriptionJob](#) 中的。

## get-medical-vocabulary

以下代码示例演示了如何使用 `get-medical-vocabulary`。

### AWS CLI

获取有关医学自定义词汇的信息

以下 `get-medical-vocabulary` 示例获取有关医学自定义词汇的信息。您可以使用 `VocabularyState` 参数来查看词汇表的处理状态。如果准备就绪，则可以在 `StartMedicalTranscriptionJob` 操作中使用它。：

```
aws transcribe get-medical-vocabulary \
  --vocabulary-name medical-vocab-example
```

输出：

```
{
  "VocabularyName": "medical-vocab-example",
```

```
"LanguageCode": "en-US",
"VocabularyState": "READY",
"LastModifiedTime": "2020-09-19T23:59:04.349000+00:00",
"DownloadUri": "https://link-to-download-the-text-file-used-to-create-your-
medical-custom-vocabulary"
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[医疗自定义词汇表](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetMedicalVocabulary](#) 中的。

## get-transcription-job

以下代码示例演示了如何使用 `get-transcription-job`。

### AWS CLI

获取有关特定转录作业的信息

以下 `get-transcription-job` 示例获取有关特定转录作业的信息。要访问转录结果，请使用 `TranscriptFileUri` 参数。使用 `MediaFileUri` 参数查看您使用此作业转录了哪个音频文件。您可以使用 `Settings` 对象查看在转录作业中启用的可选功能。

```
aws transcribe get-transcription-job \
  --transcription-job-name your-transcription-job
```

输出：

```
{
  "TranscriptionJob": {
    "TranscriptionJobName": "your-transcription-job",
    "TranscriptionJobStatus": "COMPLETED",
    "LanguageCode": "language-code",
    "MediaSampleRateHertz": 48000,
    "MediaFormat": "mp4",
    "Media": {
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.file-extension"
    },
    "Transcript": {
      "TranscriptFileUri": "https://Amazon-S3-file-location-of-transcription-
output"
    },
  },
}
```

```
"StartTime": "2020-09-18T22:27:23.970000+00:00",
"CreationTime": "2020-09-18T22:27:23.948000+00:00",
"CompletionTime": "2020-09-18T22:28:21.197000+00:00",
"Settings": {
  "ChannelIdentification": false,
  "ShowAlternatives": false
},
"IdentifyLanguage": true,
"IdentifiedLanguageScore": 0.8672199249267578
}
```

有关更多信息，请参阅《Amazon Transcribe 开发者指南》中的[入门 \( AWS 命令行界面 \)](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetTranscriptionJob](#)中的。

## get-vocabulary-filter

以下代码示例演示了如何使用 `get-vocabulary-filter`。

### AWS CLI

获取有关词汇过滤器的信息

以下`get-vocabulary-filter`示例获取有关词汇过滤器的信息。您可以使用 `DownloadUri` 参数获取用于创建词汇过滤器的单词列表。

```
aws transcribe get-vocabulary-filter \
  --vocabulary-filter-name testFilter
```

输出：

```
{
  "VocabularyFilterName": "testFilter",
  "LanguageCode": "language-code",
  "LastModifiedTime": "2020-05-07T22:39:32.147000+00:00",
  "DownloadUri": "https://Amazon-S3-location-to-download-your-vocabulary-filter"
}
```

有关更多信息，请参阅《Amazon Transcribe 开发者指南》中的[筛选不需要的单词](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVocabularyFilter](#)中的。

## get-vocabulary

以下代码示例演示了如何使用 `get-vocabulary`。

### AWS CLI

获取有关自定义词汇表的信息

以下 `get-vocabulary` 示例获取有关以前创建的自定义词汇表的信息。

```
aws transcribe get-vocabulary \  
  --vocabulary-name cli-vocab-1
```

输出：

```
{  
  "VocabularyName": "cli-vocab-1",  
  "LanguageCode": "language-code",  
  "VocabularyState": "READY",  
  "LastModifiedTime": "2020-09-19T23:22:32.836000+00:00",  
  "DownloadUri": "https://link-to-download-the-text-file-used-to-create-your-custom-vocabulary"  
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[自定义词汇表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetVocabulary](#)中的。

## list-language-models

以下代码示例演示了如何使用 `list-language-models`。

### AWS CLI

列出您的自定义语言模型

以下`list-language-models`示例列出了与您的 AWS 账户和地区关联的自定义语言模型。您可以使用`S3Uri`和`TuningDataS3Uri`参数来查找用作训练数据或调整数据的 Amazon S3 前缀。会 `BaseModelName` 告诉您是否使用了 `NarrowBand`或 `WideBand` 模型来创建自定义语言模型。您可以使用基本模型使用 `NarrowBand` 自定义语言模型转录采样率低于 16 kHz 的音频。您可以使用基本模型使用 `WideBand` 自定义语言模型转录 16 kHz 或更高的音频。该`ModelStatus`参数显示您是

否可以在转录作业中使用自定义语言模型。如果该值为 COMPLETED，则可以在转录作业中使用它。

```
aws transcribe list-language-models
```

输出：

```
{
  "Models": [
    {
      "ModelName": "cli-clm-2",
      "CreateTime": "2020-09-25T17:57:38.504000+00:00",
      "LastModifiedTime": "2020-09-25T17:57:48.585000+00:00",
      "LanguageCode": "language-code",
      "BaseModelName": "WideBand",
      "ModelStatus": "IN_PROGRESS",
      "UpgradeAvailability": false,
      "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/clm-training-data/",
        "TuningDataS3Uri": "s3://DOC-EXAMPLE-BUCKET/clm-tuning-data/",
        "DataAccessRoleArn": "arn:aws:iam::AWS-account-number:role/IAM-role-used-to-create-the-custom-language-model"
      }
    },
    {
      "ModelName": "cli-clm-1",
      "CreateTime": "2020-09-25T17:16:01.835000+00:00",
      "LastModifiedTime": "2020-09-25T17:16:15.555000+00:00",
      "LanguageCode": "language-code",
      "BaseModelName": "WideBand",
      "ModelStatus": "IN_PROGRESS",
      "UpgradeAvailability": false,
      "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET/clm-training-data/",
        "DataAccessRoleArn": "arn:aws:iam::AWS-account-number:role/IAM-role-used-to-create-the-custom-language-model"
      }
    },
    {
      "ModelName": "clm-console-1",
      "CreateTime": "2020-09-24T19:26:28.076000+00:00",
      "LastModifiedTime": "2020-09-25T04:25:22.271000+00:00",
      "LanguageCode": "language-code",
```

```

    "BaseModelName": "NarrowBand",
    "ModelStatus": "COMPLETED",
    "UpgradeAvailability": false,
    "InputDataConfig": {
      "S3Uri": "s3://DOC-EXAMPLE-BUCKET/clm-training-data/",
      "DataAccessRoleArn": "arn:aws:iam::AWS-account-number:role/IAM-role-used-to-create-the-custom-language-model"
    }
  }
]
}

```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[使用自定义语言模型提高特定领域的转录准确性](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListLanguageModels](#)中的。

## list-medical-transcription-jobs

以下代码示例演示了如何使用 `list-medical-transcription-jobs`。

### AWS CLI

#### 列出医疗转录作业

以下`list-medical-transcription-jobs`示例列出了与您的 AWS 账户和地区相关的医疗转录作业。要获取有关特定转录作业的更多信息，请在转录输出中复制 `MedicalTranscriptionJobName` 参数的值，然后为命令的`MedicalTranscriptionJobName`选项指定该值。`get-medical-transcription-job`要查看更多转录作业，请复制 `NextToken` 参数的值，再次运行该`list-medical-transcription-jobs`命令，然后在`--next-token`选项中指定该值。

```
aws transcribe list-medical-transcription-jobs
```

输出：

```

{
  "NextToken": "3/PblzkiGhzjER3KHuQt2fmbPLF7cDYafjFMEoGn440N/
gsuUSTIkGyanvRE6WMXFd/ZTEc2EZj+P9eii/
z102FDYli6RLI0WoRX4RwMisVrh9G0Kie0Y8ikBCdtqLZB10Wa9McC+eb0l
+LaDtZPC4u6ttoHLRlEfzqstHXSgapXg3tEBtm9piIaPB6MOM5BB6t86+qtmocTR/
qrteHZBBudhTfbCwhsxaqujHiiUvFdm3BQbKKWIW06yV9b+4f38oD2lVIan
+vfUs3gBYA15VTdMXXzQPBQ0HPjtwmFI+IWX15nSUjWuN3TUylHgPWzDaYT8qBtu0Z+3UG4V6b

```



```
+K2CC0XszXg5rBq9hYgNzy4XoFh/6s5DoSznzq49Q9xHgHdT2yBADFmvFK7myZBsJ75+2vQZ0SVpWUPy3WT/32zFAcoEL
+mFYfUjtTZ8n/jq7aQEjQ42A
+X/7K6Jg0cdVPtEg8P1Dr5kgYYG3q30mYXX37U3FZuJmnTI63VtIXsNn0U5eGoY0btpk00Nq9UkzgSJxqj84ZD5n
+S0EGy9ZUYBJRRcGeYUM3Q4DbSJfUwSAqcFdLIWZdp8qIREMQIBWy7BLwSdyqsQo2vRrd53hm5aWM7SVf6pPq6X/
IXR5+1eU00D8/coaTT4ES2DerbV6RkV4o0VT1d0SdVX/
MmtkNG8nYj8PqU07w7988quh1ZP6D80veJS1q73tUUR9MjnGernW2tAnvnLNhdefBcD
+sZVfYq3iBMFY7wTy1P1G6NqW9GrYDYox3tTPW1D7phpbVSYKrh/
PdYrps5UxnsGoA1b7L/FfAXDfUoGrGUB4N3JsPYXX9D++g+6gV1qBBs/
Wff934aKqfD6UTggm/zV3GA0WiBpfvAZRvEb924i6yGHYMC7y5401ZAwSBupmI
+FFd13CaP04kN1vJlth6aM5vUPXg4BpyUhtbRhWd/KxCvf9K0tLJGyL1A=="
  "MedicalTranscriptionJobSummaries": [
    {
      "MedicalTranscriptionJobName": "vocabulary-dictation-medical-
transcription-job",
      "CreationTime": "2020-09-21T21:17:27.016000+00:00",
      "StartTime": "2020-09-21T21:17:27.045000+00:00",
      "CompletionTime": "2020-09-21T21:17:59.561000+00:00",
      "LanguageCode": "en-US",
      "TranscriptionJobStatus": "COMPLETED",
      "OutputLocationType": "CUSTOMER_BUCKET",
      "Specialty": "PRIMARYCARE",
      "Type": "DICTATION"
    },
    {
      "MedicalTranscriptionJobName": "alternatives-dictation-medical-
transcription-job",
      "CreationTime": "2020-09-21T21:01:14.569000+00:00",
      "StartTime": "2020-09-21T21:01:14.592000+00:00",
      "CompletionTime": "2020-09-21T21:01:43.606000+00:00",
      "LanguageCode": "en-US",
      "TranscriptionJobStatus": "COMPLETED",
      "OutputLocationType": "CUSTOMER_BUCKET",
      "Specialty": "PRIMARYCARE",
      "Type": "DICTATION"
    },
    {
      "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
      "CreationTime": "2020-09-21T19:09:18.171000+00:00",
      "StartTime": "2020-09-21T19:09:18.199000+00:00",
      "CompletionTime": "2020-09-21T19:10:22.516000+00:00",
      "LanguageCode": "en-US",
      "TranscriptionJobStatus": "COMPLETED",
      "OutputLocationType": "CUSTOMER_BUCKET",
```

```
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION"
  },
  {
    "MedicalTranscriptionJobName": "speaker-id-conversation-medical-
transcription-job",
    "CreationTime": "2020-09-21T18:43:37.157000+00:00",
    "StartTime": "2020-09-21T18:43:37.265000+00:00",
    "CompletionTime": "2020-09-21T18:44:21.192000+00:00",
    "LanguageCode": "en-US",
    "TranscriptionJobStatus": "COMPLETED",
    "OutputLocationType": "CUSTOMER_BUCKET",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION"
  },
  {
    "MedicalTranscriptionJobName": "multichannel-conversation-medical-
transcription-job",
    "CreationTime": "2020-09-20T23:46:44.053000+00:00",
    "StartTime": "2020-09-20T23:46:44.081000+00:00",
    "CompletionTime": "2020-09-20T23:47:35.851000+00:00",
    "LanguageCode": "en-US",
    "TranscriptionJobStatus": "COMPLETED",
    "OutputLocationType": "CUSTOMER_BUCKET",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION"
  }
]
}
```

欲了解更多信息，请参阅《亚马逊转录开发者指南》中的 <https://docs.aws.amazon.com/transcribe/latest/dg/batch-med-transcription.html>。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListMedicalTranscriptionJobs](#)中的。

## list-medical-vocabularies

以下代码示例演示了如何使用 list-medical-vocabularies。

### AWS CLI

列出您的医学自定义词汇表

以下`list-medical-vocabularies`示例列出了与您的 AWS 账户和地区相关的医学自定义词汇表。要获取有关特定转录作业的更多信息，请在转录输出中复制`MedicalTranscriptionJobName`参数的值，然后为命令的`MedicalTranscriptionJobName`选项指定该值。`get-medical-transcription-job`要查看更多转录作业，请复制`NextToken`参数的值，再次运行该`list-medical-transcription-jobs`命令，然后在`--next-token`选项中指定该值。

```
aws transcribe list-medical-vocabularies
```

输出：

```
{
  "Vocabularies": [
    {
      "VocabularyName": "cli-medical-vocab-2",
      "LanguageCode": "en-US",
      "LastModifiedTime": "2020-09-21T21:44:59.521000+00:00",
      "VocabularyState": "READY"
    },
    {
      "VocabularyName": "cli-medical-vocab-1",
      "LanguageCode": "en-US",
      "LastModifiedTime": "2020-09-19T23:59:04.349000+00:00",
      "VocabularyState": "READY"
    }
  ]
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[医疗自定义词汇表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListMedicalVocabularies](#)中的。

## list-transcription-jobs

以下代码示例演示了如何使用 `list-transcription-jobs`。

### AWS CLI

#### 列出转录作业

以下`list-transcription-jobs`示例列出了与您的 AWS 账户和地区相关的转录作业。

```
aws transcribe list-transcription-jobs
```

输出：

```
{
  "NextToken": "NextToken",
  "TranscriptionJobSummaries": [
    {
      "TranscriptionJobName": "speak-id-job-1",
      "CreationTime": "2020-08-17T21:06:15.391000+00:00",
      "StartTime": "2020-08-17T21:06:15.416000+00:00",
      "CompletionTime": "2020-08-17T21:07:05.098000+00:00",
      "LanguageCode": "language-code",
      "TranscriptionJobStatus": "COMPLETED",
      "OutputLocationType": "SERVICE_BUCKET"
    },
    {
      "TranscriptionJobName": "job-1",
      "CreationTime": "2020-08-17T20:50:24.207000+00:00",
      "StartTime": "2020-08-17T20:50:24.230000+00:00",
      "CompletionTime": "2020-08-17T20:52:18.737000+00:00",
      "LanguageCode": "language-code",
      "TranscriptionJobStatus": "COMPLETED",
      "OutputLocationType": "SERVICE_BUCKET"
    },
    {
      "TranscriptionJobName": "sdk-test-job-4",
      "CreationTime": "2020-08-17T20:32:27.917000+00:00",
      "StartTime": "2020-08-17T20:32:27.956000+00:00",
      "CompletionTime": "2020-08-17T20:33:15.126000+00:00",
      "LanguageCode": "language-code",
      "TranscriptionJobStatus": "COMPLETED",
      "OutputLocationType": "SERVICE_BUCKET"
    },
    {
      "TranscriptionJobName": "Diarization-speak-id",
      "CreationTime": "2020-08-10T22:10:09.066000+00:00",
      "StartTime": "2020-08-10T22:10:09.116000+00:00",
      "CompletionTime": "2020-08-10T22:26:48.172000+00:00",
      "LanguageCode": "language-code",
      "TranscriptionJobStatus": "COMPLETED",
      "OutputLocationType": "SERVICE_BUCKET"
    }
  ],
}
```

```
{
  "TranscriptionJobName": "your-transcription-job-name",
  "CreationTime": "2020-07-29T17:45:09.791000+00:00",
  "StartTime": "2020-07-29T17:45:09.826000+00:00",
  "CompletionTime": "2020-07-29T17:46:20.831000+00:00",
  "LanguageCode": "language-code",
  "TranscriptionJobStatus": "COMPLETED",
  "OutputLocationType": "SERVICE_BUCKET"
}
```

有关更多信息，请参阅《Amazon Transcribe 开发者指南》中的[入门 \( AWS 命令行界面 \)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListTranscriptionJobs](#)中的。

## list-vocabularies

以下代码示例演示了如何使用 list-vocabularies。

### AWS CLI

列出自定义词汇表

以下 list-vocabularies 示例列出了与您的 AWS 账户和地区关联的自定义词汇表。

```
aws transcribe list-vocabularies
```

输出：

```
{
  "NextToken": "NextToken",
  "Vocabularies": [
    {
      "VocabularyName": "ards-test-1",
      "LanguageCode": "language-code",
      "LastModifiedTime": "2020-04-27T22:00:27.330000+00:00",
      "VocabularyState": "READY"
    },
    {
      "VocabularyName": "sample-test",
      "LanguageCode": "language-code",

```

```
    "LastModifiedTime": "2020-04-24T23:04:11.044000+00:00",
    "VocabularyState": "READY"
  },
  {
    "VocabularyName": "CRLF-to-LF-test-3-1",
    "LanguageCode": "language-code",
    "LastModifiedTime": "2020-04-24T22:12:22.277000+00:00",
    "VocabularyState": "READY"
  },
  {
    "VocabularyName": "CRLF-to-LF-test-2",
    "LanguageCode": "language-code",
    "LastModifiedTime": "2020-04-24T21:53:50.455000+00:00",
    "VocabularyState": "READY"
  },
  {
    "VocabularyName": "CRLF-to-LF-1-1",
    "LanguageCode": "language-code",
    "LastModifiedTime": "2020-04-24T21:39:33.356000+00:00",
    "VocabularyState": "READY"
  }
]
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[自定义词汇表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListVocabularies](#)中的。

## list-vocabulary-filters

以下代码示例演示了如何使用 list-vocabulary-filters。

### AWS CLI

列出你的词汇过滤器

以下list-vocabulary-filters示例列出了与您的 AWS 账户和地区关联的词汇过滤器。

```
aws transcribe list-vocabulary-filters
```

输出：

```
{
```

```
"NextToken": "NextToken": [  
  {  
    "VocabularyFilterName": "testFilter",  
    "LanguageCode": "language-code",  
    "LastModifiedTime": "2020-05-07T22:39:32.147000+00:00"  
  },  
  {  
    "VocabularyFilterName": "testFilter2",  
    "LanguageCode": "language-code",  
    "LastModifiedTime": "2020-05-21T23:29:35.174000+00:00"  
  },  
  {  
    "VocabularyFilterName": "filter2",  
    "LanguageCode": "language-code",  
    "LastModifiedTime": "2020-05-08T20:18:26.426000+00:00"  
  },  
  {  
    "VocabularyFilterName": "filter-review",  
    "LanguageCode": "language-code",  
    "LastModifiedTime": "2020-06-03T18:52:30.448000+00:00"  
  },  
  {  
    "VocabularyFilterName": "crlf-filt",  
    "LanguageCode": "language-code",  
    "LastModifiedTime": "2020-05-22T19:42:42.737000+00:00"  
  }  
]  
}
```

有关更多信息，请参阅《Amazon Transcribe 开发者指南》中的[筛选不需要的单词](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListVocabularyFilters](#) 中的。

## start-medical-transcription-job

以下代码示例演示了如何使用 start-medical-transcription-job。

### AWS CLI

#### 示例 1：转录存储为音频文件的医疗口述

以下 start-medical-transcription-job 示例转录一个音频文件。您可以在 OutputBucketName 参数中指定转录输出位置。

```
aws transcribe start-medical-transcription-job \  
  --cli-input-json file://myfile.json
```

myfile.json 的内容：

```
{  
  "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-job",  
  "LanguageCode": "language-code",  
  "Specialty": "PRIMARYCARE",  
  "Type": "DICTATION",  
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",  
  "Media": {  
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"  
  }  
}
```

输出：

```
{  
  "MedicalTranscriptionJob": {  
    "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-job",  
    "TranscriptionJobStatus": "IN_PROGRESS",  
    "LanguageCode": "language-code",  
    "Media": {  
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"  
    },  
    "StartTime": "2020-09-20T00:35:22.256000+00:00",  
    "CreationTime": "2020-09-20T00:35:22.218000+00:00",  
    "Specialty": "PRIMARYCARE",  
    "Type": "DICTATION"  
  }  
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[批量转录概述](#)。

示例 2：转录存储为音频文件的临床医生与患者之间的对话

以下 start-medical-transcription-job 示例转录包含有临床医生与患者之间对话的音频文件。您可以在 OutputBucketName 参数中指定转录输出的位置。

```
aws transcribe start-medical-transcription-job \  
  --cli-input-json file://myfile.json
```



```
--cli-input-json file://mysecondfile.json
```

mysecondfile.json 的内容：

```
{
  "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-job",
  "LanguageCode": "language-code",
  "Specialty": "PRIMARYCARE",
  "Type": "CONVERSATION",
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
  }
}
```

输出：

```
{
  "MedicalTranscriptionJob": {
    "MedicalTranscriptionJobName": "simple-conversation-medical-transcription-job",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "LanguageCode": "language-code",
    "Media": {
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
    },
    "StartTime": "2020-09-20T23:19:49.965000+00:00",
    "CreationTime": "2020-09-20T23:19:49.941000+00:00",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION"
  }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[批量转录概述](#)。

示例 3：转录临床医生与患者之间对话的多声道音频文件

以下 start-medical-transcription-job 示例转录音频文件中每个声道的音频，并将每个声道的单独转录合并为一个转录输出。您可以在 OutputBucketName 参数中指定转录输出位置。

```
aws transcribe start-medical-transcription-job \
```

```
--cli-input-json file://mythirdfile.json
```

mythirdfile.json 的内容：

```
{
  "MedicalTranscriptionJobName": "multichannel-conversation-medical-transcription-
job",
  "LanguageCode": "language-code",
  "Specialty": "PRIMARYCARE",
  "Type": "CONVERSATION",
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
  },
  "Settings": {
    "ChannelIdentification": true
  }
}
```

输出：

```
{
  "MedicalTranscriptionJob": {
    "MedicalTranscriptionJobName": "multichannel-conversation-medical-
transcription-job",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "LanguageCode": "language-code",
    "Media": {
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
    },
    "StartTime": "2020-09-20T23:46:44.081000+00:00",
    "CreationTime": "2020-09-20T23:46:44.053000+00:00",
    "Settings": {
      "ChannelIdentification": true
    },
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION"
  }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[声道识别](#)。

示例 4：转录临床医生与患者之间对话的音频文件并在转录输出中识别发言者

以下 `start-medical-transcription-job` 示例转录一个音频文件，并在转录输出中标记每个发言者的语音。您可以在 `OutputBucketName` 参数中指定转录输出位置。

```
aws transcribe start-medical-transcription-job \  
  --cli-input-json file://myfourthfile.json
```

`myfourthfile.json` 的内容：

```
{  
  "MedicalTranscriptionJobName": "speaker-id-conversation-medical-transcription-  
job",  
  "LanguageCode": "language-code",  
  "Specialty": "PRIMARYCARE",  
  "Type": "CONVERSATION",  
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",  
  "Media": {  
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"  
  },  
  "Settings": {  
    "ShowSpeakerLabels": true,  
    "MaxSpeakerLabels": 2  
  }  
}
```

输出：

```
{  
  "MedicalTranscriptionJob": {  
    "MedicalTranscriptionJobName": "speaker-id-conversation-medical-  
transcription-job",  
    "TranscriptionJobStatus": "IN_PROGRESS",  
    "LanguageCode": "language-code",  
    "Media": {  
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"  
    },  
    "StartTime": "2020-09-21T18:43:37.265000+00:00",  
    "CreationTime": "2020-09-21T18:43:37.157000+00:00",  
    "Settings": {  
      "ShowSpeakerLabels": true,  
      "MaxSpeakerLabels": 2  
    },  
    "Specialty": "PRIMARYCARE",  
  }  
}
```

```
    "Type": "CONVERSATION"
  }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[识别发言者](#)。

示例 5：转录存储为音频文件的医疗对话，最多两个备选转录

以下 `start-medical-transcription-job` 示例通过单个音频文件创建最多两个备选转录。每个转录具有关联的置信度。默认情况下，Amazon Transcribe 返回置信度最高的转录。您可以指定 Amazon Transcribe 返回置信度较低的其他转录。您可以在 `OutputBucketName` 参数中指定转录输出位置。

```
aws transcribe start-medical-transcription-job \
  --cli-input-json file://myfifthfile.json
```

`myfifthfile.json` 的内容：

```
{
  "MedicalTranscriptionJobName": "alternatives-conversation-medical-transcription-
job",
  "LanguageCode": "language-code",
  "Specialty": "PRIMARYCARE",
  "Type": "CONVERSATION",
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
  },
  "Settings": {
    "ShowAlternatives": true,
    "MaxAlternatives": 2
  }
}
```

输出：

```
{
  "MedicalTranscriptionJob": {
    "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "LanguageCode": "language-code",
```

```

    "Media": {
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
    },
    "StartTime": "2020-09-21T19:09:18.199000+00:00",
    "CreationTime": "2020-09-21T19:09:18.171000+00:00",
    "Settings": {
      "ShowAlternatives": true,
      "MaxAlternatives": 2
    },
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION"
  }
}

```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[备选转录](#)。

示例 6：转录医疗口述音频文件，最多具有两个备选转录

以下 `start-medical-transcription-job` 示例转录一个音频文件，并使用词汇表过滤器屏蔽任何不需要的单词。您可以在 `OutputBucketName` 参数中指定转录输出的位置。

```

aws transcribe start-medical-transcription-job \
  --cli-input-json file://mysixthfile.json

```

`mysixthfile.json` 的内容：

```

{
  "MedicalTranscriptionJobName": "alternatives-conversation-medical-transcription-
job",
  "LanguageCode": "language-code",
  "Specialty": "PRIMARYCARE",
  "Type": "DICTATION",
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
  },
  "Settings": {
    "ShowAlternatives": true,
    "MaxAlternatives": 2
  }
}

```

输出：

```
{
  "MedicalTranscriptionJob": {
    "MedicalTranscriptionJobName": "alternatives-dictation-medical-
transcription-job",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "LanguageCode": "language-code",
    "Media": {
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
    },
    "StartTime": "2020-09-21T21:01:14.592000+00:00",
    "CreationTime": "2020-09-21T21:01:14.569000+00:00",
    "Settings": {
      "ShowAlternatives": true,
      "MaxAlternatives": 2
    },
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION"
  }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[备选转录](#)。

#### 示例 7：使用自定义词汇表更准确地转录医疗口述音频文件

以下 `start-medical-transcription-job` 示例转录一个音频文件，并使用您以前创建的医学自定义词汇表提高转录准确性。您可以在 `OutputBucketName` 参数中指定转录输出位置。

```
aws transcribe start-transcription-job \
  --cli-input-json file://myseventhfile.json
```

`mysixthfile.json` 的内容：

```
{
  "MedicalTranscriptionJobName": "vocabulary-dictation-medical-transcription-job",
  "LanguageCode": "language-code",
  "Specialty": "PRIMARYCARE",
  "Type": "DICTATION",
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
  },
  "Settings": {
```

```
    "VocabularyName": "cli-medical-vocab-1"
  }
}
```

输出：

```
{
  "MedicalTranscriptionJob": {
    "MedicalTranscriptionJobName": "vocabulary-dictation-medical-transcription-
job",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "LanguageCode": "language-code",
    "Media": {
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
    },
    "StartTime": "2020-09-21T21:17:27.045000+00:00",
    "CreationTime": "2020-09-21T21:17:27.016000+00:00",
    "Settings": {
      "VocabularyName": "cli-medical-vocab-1"
    },
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION"
  }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[医疗自定义词汇表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartMedicalTranscriptionJob](#)中的。

## start-transcription-job

以下代码示例演示了如何使用 start-transcription-job。

### AWS CLI

示例 1：转录音频文件

以下 start-transcription-job 示例转录音频文件。

```
aws transcribe start-transcription-job \
  --cli-input-json file://myfile.json
```

myfile.json 的内容：

```
{
  "TranscriptionJobName": "cli-simple-transcription-job",
  "LanguageCode": "the-language-of-your-transcription-job",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-file-
name.file-extension"
  }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发者指南》中的[入门 \( AWS 命令行界面 \)](#)。

## 示例 2：转录多声道音频文件

以下 `start-transcription-job` 示例转录多声道音频文件。

```
aws transcribe start-transcription-job \
  --cli-input-json file://mysecondfile.json
```

`mysecondfile.json` 的内容：

```
{
  "TranscriptionJobName": "cli-channelid-job",
  "LanguageCode": "the-language-of-your-transcription-job",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-file-
name.file-extension"
  },
  "Settings":{
    "ChannelIdentification":true
  }
}
```

输出：

```
{
  "TranscriptionJob": {
    "TranscriptionJobName": "cli-channelid-job",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
    }
  }
}
```



```
    },
    "StartTime": "2020-09-17T16:07:56.817000+00:00",
    "CreationTime": "2020-09-17T16:07:56.784000+00:00",
    "Settings": {
      "ChannelIdentification": true
    }
  }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[转录多声道音频](#)。

### 示例 3：转录音频文件并识别不同的发言者

以下 `start-transcription-job` 示例转录音频文件，并在转录输出中识别发言者。

```
aws transcribe start-transcription-job \
  --cli-input-json file://mythirdfile.json
```

`mythirdfile.json` 的内容：

```
{
  "TranscriptionJobName": "cli-speakerid-job",
  "LanguageCode": "the-language-of-your-transcription-job",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-file-
name.file-extension"
  },
  "Settings":{
    "ShowSpeakerLabels": true,
    "MaxSpeakerLabels": 2
  }
}
```

输出：

```
{
  "TranscriptionJob": {
    "TranscriptionJobName": "cli-speakerid-job",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
```

```

    },
    "StartTime": "2020-09-17T16:22:59.696000+00:00",
    "CreationTime": "2020-09-17T16:22:59.676000+00:00",
    "Settings": {
      "ShowSpeakerLabels": true,
      "MaxSpeakerLabels": 2
    }
  }
}

```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[识别发言者](#)。

示例 4：转录音频文件并在转录输出中屏蔽任何不需要的单词

以下 `start-transcription-job` 示例转录音频文件，并使用您以前创建的词汇表过滤器屏蔽任何不需要的单词。

```

aws transcribe start-transcription-job \
  --cli-input-json file://myfourthfile.json

```

`myfourthfile.json` 的内容：

```

{
  "TranscriptionJobName": "cli-filter-mask-job",
  "LanguageCode": "the-language-of-your-transcription-job",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-file-
name.file-extension"
  },
  "Settings":{
    "VocabularyFilterName": "your-vocabulary-filter",
    "VocabularyFilterMethod": "mask"
  }
}

```

输出：

```

{
  "TranscriptionJob": {
    "TranscriptionJobName": "cli-filter-mask-job",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {

```

```

        "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
    },
    "StartTime": "2020-09-18T16:36:18.568000+00:00",
    "CreationTime": "2020-09-18T16:36:18.547000+00:00",
    "Settings": {
        "VocabularyFilterName": "your-vocabulary-filter",
        "VocabularyFilterMethod": "mask"
    }
}
}
}

```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[过滤转录](#)。

示例 5：转录音频文件并在转录输出中删除任何不需要的单词

以下 `start-transcription-job` 示例转录音频文件，并使用您以前创建的词汇表过滤器屏蔽任何不需要的单词。

```

aws transcribe start-transcription-job \
  --cli-input-json file://myfifthfile.json

```

`myfifthfile.json` 的内容：

```

{
  "TranscriptionJobName": "cli-filter-remove-job",
  "LanguageCode": "the-language-of-your-transcription-job",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-file-
name.file-extension"
  },
  "Settings":{
    "VocabularyFilterName": "your-vocabulary-filter",
    "VocabularyFilterMethod": "remove"
  }
}

```

输出：

```

{
  "TranscriptionJob": {
    "TranscriptionJobName": "cli-filter-remove-job",
    "TranscriptionJobStatus": "IN_PROGRESS",

```

```

    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "StartTime": "2020-09-18T16:36:18.568000+00:00",
    "CreationTime": "2020-09-18T16:36:18.547000+00:00",
    "Settings": {
      "VocabularyFilterName": "your-vocabulary-filter",
      "VocabularyFilterMethod": "remove"
    }
  }
}

```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[过滤转录](#)。

#### 示例 6：使用自定义词汇表更准确地转录音频文件

以下 `start-transcription-job` 示例转录音频文件，并使用您以前创建的词汇表过滤器屏蔽任何不需要的单词。

```

aws transcribe start-transcription-job \
  --cli-input-json file://mysixthfile.json

```

`mysixthfile.json` 的内容：

```

{
  "TranscriptionJobName": "cli-vocab-job",
  "LanguageCode": "the-language-of-your-transcription-job",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-file-
name.file-extension"
  },
  "Settings":{
    "VocabularyName": "your-vocabulary"
  }
}

```

输出：

```

{
  "TranscriptionJob": {

```

```
"TranscriptionJobName": "cli-vocab-job",
"TranscriptionJobStatus": "IN_PROGRESS",
"LanguageCode": "the-language-of-your-transcription-job",
"Media": {
  "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
},
"StartTime": "2020-09-18T16:36:18.568000+00:00",
"CreationTime": "2020-09-18T16:36:18.547000+00:00",
"Settings": {
  "VocabularyName": "your-vocabulary"
}
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[过滤转录](#)。

#### 示例 7：识别音频文件语言并转录该文件

以下 `start-transcription-job` 示例转录音频文件，并使用您以前创建的词汇表过滤器屏蔽任何不需要的单词。

```
aws transcribe start-transcription-job \
  --cli-input-json file://myseventhfile.json
```

`myseventhfile.json` 的内容：

```
{
  "TranscriptionJobName": "cli-identify-language-transcription-job",
  "IdentifyLanguage": true,
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-file-
name.file-extension"
  }
}
```

输出：

```
{
  "TranscriptionJob": {
    "TranscriptionJobName": "cli-identify-language-transcription-job",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "Media": {
```

```

        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "StartTime": "2020-09-18T22:27:23.970000+00:00",
    "CreationTime": "2020-09-18T22:27:23.948000+00:00",
    "IdentifyLanguage": true
}
}

```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[识别语言](#)。

#### 示例 8：转录音频文件并编辑个人信息

以下 `start-transcription-job` 示例转录音频文件，并在转录输出中编辑任何个人信息。

```

aws transcribe start-transcription-job \
  --cli-input-json file://myeighthfile.json

```

`myeighthfile.json` 的内容：

```

{
  "TranscriptionJobName": "cli-redaction-job",
  "LanguageCode": "language-code",
  "Media": {
    "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
  },
  "ContentRedaction": {
    "RedactionOutput": "redacted",
    "RedactionType": "PII"
  }
}

```

输出：

```

{
  "TranscriptionJob": {
    "TranscriptionJobName": "cli-redaction-job",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "LanguageCode": "language-code",
    "Media": {
      "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
    },
    "StartTime": "2020-09-25T23:49:13.195000+00:00",

```

```

    "CreationTime": "2020-09-25T23:49:13.176000+00:00",
    "ContentRedaction": {
      "RedactionType": "PII",
      "RedactionOutput": "redacted"
    }
  }
}

```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[自动内容编辑](#)。

示例 9：生成个人信息 (PII) 经过编辑和未经过编辑的转录

以下 `start-transcription-job` 示例生成两个音频文件转录，一个转录中的个人信息经过编辑，另一个转录没有进行任何编辑。

```

aws transcribe start-transcription-job \
  --cli-input-json file://myninthfile.json

```

`myninthfile.json` 的内容：

```

{
  "TranscriptionJobName": "cli-redaction-job-with-unredacted-transcript",
  "LanguageCode": "language-code",
  "Media": {
    "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
  },
  "ContentRedaction": {
    "RedactionOutput": "redacted_and_unredacted",
    "RedactionType": "PII"
  }
}

```

输出：

```

{
  "TranscriptionJob": {
    "TranscriptionJobName": "cli-redaction-job-with-unredacted-transcript",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "LanguageCode": "language-code",
    "Media": {
      "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
    },
    "StartTime": "2020-09-25T23:59:47.677000+00:00",

```

```

    "CreationTime": "2020-09-25T23:59:47.653000+00:00",
    "ContentRedaction": {
      "RedactionType": "PII",
      "RedactionOutput": "redacted_and_unredacted"
    }
  }
}

```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[自动内容编辑](#)。

示例 10：使用您以前创建的自定义语言模型转录音频文件

以下 `start-transcription-job` 示例使用您以前创建的自定义语言模型转录音频文件。

```

aws transcribe start-transcription-job \
  --cli-input-json file://mytenthfile.json

```

`mytenthfile.json` 的内容：

```

{
  "TranscriptionJobName": "cli-clm-2-job-1",
  "LanguageCode": "language-code",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.file-extension"
  },
  "ModelSettings": {
    "LanguageModelName": "cli-clm-2"
  }
}

```

输出：

```

{
  "TranscriptionJob": {
    "TranscriptionJobName": "cli-clm-2-job-1",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "LanguageCode": "language-code",
    "Media": {
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.file-extension"
    },
    "StartTime": "2020-09-28T17:56:01.835000+00:00",
    "CreationTime": "2020-09-28T17:56:01.801000+00:00",
    "ModelSettings": {

```



```
        "LanguageModelName": "cli-clm-2"
      }
    }
  }
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[使用自定义语言模型提高特定领域的转录准确性](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartTranscriptionJob](#)中的。

## update-medical-vocabulary

以下代码示例演示了如何使用 update-medical-vocabulary。

### AWS CLI

使用新术语更新医学自定义词汇。

以下update-medical-vocabulary示例用新的术语替换了医学自定义词汇表中使用的术语。先决条件：要替换医学自定义词汇表中的术语，您需要一个包含新术语的文件。

```
aws transcribe update-medical-vocabulary \
  --vocabulary-file-uri s3://DOC-EXAMPLE-BUCKET/Amazon-S3-Prefix/medical-custom-
vocabulary.txt \
  --vocabulary-name medical-custom-vocabulary \
  --language-code language
```

输出：

```
{
  "VocabularyName": "medical-custom-vocabulary",
  "LanguageCode": "en-US",
  "VocabularyState": "PENDING"
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[医疗自定义词汇表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateMedicalVocabulary](#)中的。

## update-vocabulary-filter

以下代码示例演示了如何使用 update-vocabulary-filter。

## AWS CLI

### 替换词汇表筛选器中的单词

以下 `update-vocabulary-filter` 示例将词汇过滤器中的单词替换为新单词。先决条件：要使用新单词更新词汇过滤器，必须将这些单词另存为文本文件。

```
aws transcribe update-vocabulary-filter \  
  --vocabulary-filter-file-uri s3://DOC-EXAMPLE-BUCKET/Amazon-S3-Prefix/your-text-  
file-to-update-your-vocabulary-filter.txt \  
  --vocabulary-filter-name vocabulary-filter-name
```

输出：

```
{  
  "VocabularyFilterName": "vocabulary-filter-name",  
  "LanguageCode": "language-code",  
  "LastModifiedTime": "2020-09-23T18:40:35.139000+00:00"  
}
```

有关更多信息，请参阅《Amazon Transcribe 开发者指南》中的[筛选不需要的单词](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateVocabularyFilter](#) 中的。

## update-vocabulary

以下代码示例演示了如何使用 `update-vocabulary`。

### AWS CLI

使用新术语更新自定义词汇表。

以下 `update-vocabulary` 示例使用您提供的新术语来覆盖用于创建自定义词汇表的术语。先决条件：要替换自定义词汇表中的术语，您需要使用一个包含新术语的文件。

```
aws transcribe update-vocabulary \  
  --vocabulary-file-uri s3://DOC-EXAMPLE-BUCKET/Amazon-S3-Prefix/custom-  
vocabulary.txt \  
  --vocabulary-name custom-vocabulary \  
  --language-code language-code
```

输出：

```
{
  "VocabularyName": "custom-vocabulary",
  "LanguageCode": "language",
  "VocabularyState": "PENDING"
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的[自定义词汇表](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateVocabulary](#)中的。

## Amazon Translate 示例使用 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 Amazon Translate 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **import-terminology**

以下代码示例演示了如何使用 import-terminology。

#### AWS CLI

##### 从文件导入自定义术语

以下import-terminology示例创建了一个MyTestTerminology从test-terminology.csv文件中调用的术语：

```
aws translate import-terminology \
  --name MyTestTerminology \
```

```
--description "Creating a test terminology in AWS Translate" \  
--merge-strategy OVERWRITE \  
--data-file fileb://test-terminology.csv \  
--terminology-data Format=CSV
```

test-terminology.csv 的内容：

en、fr、es、zh Hello world！，你好 tout le monde！，Hola Mundo！，？？？ 亚马逊、亚马逊、亚马逊、亚马逊、亚马逊

输出：

```
{  
  "TerminologyProperties": {  
    "SourceLanguageCode": "en",  
    "Name": "MyTestTerminology",  
    "TargetLanguageCodes": [  
      "fr",  
      "es",  
      "zh"  
    ],  
    "SizeBytes": 97,  
    "LastUpdatedAt": 1571089500.851,  
    "CreatedAt": 1571089500.851,  
    "TermCount": 6,  
    "Arn": "arn:aws:translate:us-west-2:123456789012:terminology/  
MyTestTerminology/LATEST",  
    "Description": "Creating a test terminology in AWS Translate"  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ImportTerminology](#)中的。

## Trusted Advisor 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 Trusted Advisor。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### get-organization-recommendation

以下代码示例演示了如何使用 `get-organization-recommendation`。

#### AWS CLI

##### 获取组织推荐

以下 `get-organization-recommendation` 示例通过组织标识符获取组织推荐。

```
aws trustedadvisor get-organization-recommendation \
  --organization-recommendation-identifier arn:aws:trustedadvisor::organization-
  recommendation/9534ec9b-bf3a-44e8-8213-2ed68b39d9d5
```

##### 输出：

```
{
  "organizationRecommendation": {
    "arn": "arn:aws:trustedadvisor::organization-recommendation/9534ec9b-
    bf3a-44e8-8213-2ed68b39d9d5",
    "name": "Lambda Runtime Deprecation Warning",
    "description": "One or more lambdas are using a deprecated runtime",
    "awsServices": [
      "lambda"
    ],
    "checkArn": "arn:aws:trustedadvisor::check/L4dfs2Q4C5",
    "id": "9534ec9b-bf3a-44e8-8213-2ed68b39d9d5",
    "lifecycleStage": "resolved",
    "pillars": [
      "security"
    ],
    "resourcesAggregates": {
      "errorCount": 0,
      "okCount": 0,
    }
  }
}
```

```
        "warningCount": 0
      },
      "source": "ta_check",
      "status": "warning",
      "type": "priority"
    }
  }
}
```

有关更多信息，请参阅《[Trusted Advisor 用户指南](#)》中的 [Tru AWS sted Advisor API 入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetOrganizationRecommendation](#)中的。

## get-recommendation

以下代码示例演示了如何使用 get-recommendation。

### AWS CLI

要获得推荐

以下get-recommendation示例通过其标识符获取推荐。

```
aws trustedadvisor get-recommendation \
  --recommendation-identifier
  arn:aws:trustedadvisor::000000000000:recommendation/55fa4d2e-
  bbb7-491a-833b-5773e9589578
```

输出：

```
{
  "recommendation": {
    "arn": "arn:aws:trustedadvisor::000000000000:recommendation/55fa4d2e-
    bbb7-491a-833b-5773e9589578",
    "name": "MFA Recommendation",
    "description": "Enable multi-factor authentication",
    "awsServices": [
      "iam"
    ],
    "checkArn": "arn:aws:trustedadvisor:::check/7DAFEemoDos",
    "id": "55fa4d2e-bbb7-491a-833b-5773e9589578",
    "lastUpdatedAt": "2023-11-01T15:57:58.673Z",
    "pillarSpecificAggregates": {
      "costOptimizing": {
```

```

        "estimatedMonthlySavings": 0.0,
        "estimatedPercentMonthlySavings": 0.0
    }
},
"pillars": [
    "security"
],
"resourcesAggregates": {
    "errorCount": 1,
    "okCount": 0,
    "warningCount": 0
},
"source": "ta_check",
"status": "error",
"type": "standard"
}
}

```

有关更多信息，请参阅《[Trusted Advisor 用户指南](#)》中的 [Tru AWS sted Advisor API 入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetRecommendation](#) 中的。

## list-checks

以下代码示例演示了如何使用 list-checks。

### AWS CLI

列出 Trusted Advisor 支票

以下 list-checks 示例列出了所有 Trusted Advisor 支票。

```
aws trustedadvisor list-checks
```

输出：

```

{
  "checkSummaries": [
    {
      "arn": "arn:aws:trustedadvisor:::check/1iG5NDGVre",
      "awsServices": [
        "EC2"
      ],

```

```
    "description": "Checks security groups for rules that allow unrestricted
access to a resource. Unrestricted access increases opportunities for malicious
activity (hacking, denial-of-service attacks, loss of data)",
    "id": "1iG5NDGVre",
    "metadata": {
      "0": "Region",
      "1": "Security Group Name",
      "2": "Security Group ID",
      "3": "Protocol",
      "4": "Port",
      "5": "Status",
      "6": "IP Range"
    },
    "name": "Security Groups - Unrestricted Access",
    "pillars": [
      "security"
    ],
    "source": "ta_check"
  },
  {
    "arn": "arn:aws:trustedadvisor:::check/1qazXsw23e",
    "awsServices": [
      "RDS"
    ],
    "description": "Checks your usage of RDS and provides recommendations
on purchase of Reserved Instances to help reduce costs incurred from using RDS
On-Demand. AWS generates these recommendations by analyzing your On-Demand usage
for the past 30 days. We then simulate every combination of reservations in the
generated category of usage in order to identify the best number of each type
of Reserved Instance to purchase to maximize your savings. This check covers
recommendations based on partial upfront payment option with 1-year or 3-year
commitment. This check is not available to accounts linked in Consolidated Billing.
Recommendations are only available for the Paying Account.",
    "id": "1qazXsw23e",
    "metadata": {
      "0": "Region",
      "1": "Family",
      "2": "Instance Type",
      "3": "License Model",
      "4": "Database Edition",
      "5": "Database Engine",
      "6": "Deployment Option",
      "7": "Recommended number of Reserved Instances to purchase",
      "8": "Expected Average Reserved Instance Utilization",
```



```

    "9": "Estimated Savings with Recommendation (monthly)",
    "10": "Upfront Cost of Reserved Instances",
    "11": "Estimated cost of Reserved Instances (monthly)",
    "12": "Estimated On-Demand Cost Post Recommended Reserved Instance
Purchase (monthly)",
    "13": "Estimated Break Even (months)",
    "14": "Lookback Period (days)",
    "15": "Term (years)"
  },
  "name": "Amazon Relational Database Service (RDS) Reserved Instance
Optimization",
  "pillars": [
    "cost_optimizing"
  ],
  "source": "ta_check"
},
{
  "arn": "arn:aws:trustedadvisor:::check/1qw23er45t",
  "awsServices": [
    "Redshift"
  ],
  "description": "Checks your usage of Redshift and provides
recommendations on purchase of Reserved Nodes to help reduce costs incurred from
using Redshift On-Demand. AWS generates these recommendations by analyzing your
On-Demand usage for the past 30 days. We then simulate every combination of
reservations in the generated category of usage in order to identify the best
number of each type of Reserved Nodes to purchase to maximize your savings. This
check covers recommendations based on partial upfront payment option with 1-year or
3-year commitment. This check is not available to accounts linked in Consolidated
Billing. Recommendations are only available for the Paying Account.",
  "id": "1qw23er45t",
  "metadata": {
    "0": "Region",
    "1": "Family",
    "2": "Node Type",
    "3": "Recommended number of Reserved Nodes to purchase",
    "4": "Expected Average Reserved Node Utilization",
    "5": "Estimated Savings with Recommendation (monthly)",
    "6": "Upfront Cost of Reserved Nodes",
    "7": "Estimated cost of Reserved Nodes (monthly)",
    "8": "Estimated On-Demand Cost Post Recommended Reserved Nodes
Purchase (monthly)",
    "9": "Estimated Break Even (months)",
    "10": "Lookback Period (days)",

```

```

        "11": "Term (years)",
      },
      "name": "Amazon Redshift Reserved Node Optimization",
      "pillars": [
        "cost_optimizing"
      ],
      "source": "ta_check"
    },
  ],
  "nextToken": "REDACTED"
}

```

有关更多信息，请参阅《[Trusted Advisor 用户指南](#)》中的 [Tru AWS sted Advisor API 入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListChecks](#) 中的。

## list-organization-recommendation-accounts

以下代码示例演示了如何使用 `list-organization-recommendation-accounts`。

### AWS CLI

#### 列出组织推荐账号

以下 `list-organization-recommendation-accounts` 示例按组织推荐的标识符列出了组织推荐的所有账户推荐摘要。

```

aws trustedadvisor list-organization-recommendation-accounts \
  --organization-recommendation-identifier arn:aws:trustedadvisor:::organization-
  recommendation/9534ec9b-bf3a-44e8-8213-2ed68b39d9d5

```

#### 输出：

```

{
  "accountRecommendationLifecycleSummaries": [{
    "accountId": "000000000000",
    "accountRecommendationArn":
    "arn:aws:trustedadvisor:::000000000000:recommendation/9534ec9b-
    bf3a-44e8-8213-2ed68b39d9d5",
    "lifecycleStage": "resolved",
    "updateReason": "Resolved issue",
    "updateReasonCode": "valid_business_case",
    "lastUpdatedAt": "2023-01-17T18:25:44.552Z"
  }
]

```

```
  ]],
  "nextToken": "REDACTED"
}
```

有关更多信息，请参阅《[Trusted Advisor 用户指南](#)》中的 [Tru AWS sted Advisor API 入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListOrganizationRecommendationAccounts](#)中的。

## list-organization-recommendation-resources

以下代码示例演示了如何使用 list-organization-recommendation-resources。

### AWS CLI

列出组织推荐资源

以下list-organization-recommendation-resources示例按标识符列出了组织推荐的所有资源。

```
aws trustedadvisor list-organization-recommendation-resources \
  --organization-recommendation-identifier arn:aws:trustedadvisor::organization-
  recommendation/5a694939-2e54-45a2-ae72-730598fa89d0
```

输出：

```
{
  "organizationRecommendationResourceSummaries": [
    {
      "arn": "arn:aws:trustedadvisor::000000000000:recommendation-
      resource/5a694939-2e54-45a2-ae72-730598fa89d0/
      bb38affc0ce0681d9a6cd13f30238ba03a8f63dfe7a379dc403c619119d86af",
      "awsResourceId": "database-1-instance-1",
      "id":
      "bb38affc0ce0681d9a6cd13f302383ba03a8f63dfe7a379dc403c619119d86af",
      "lastUpdatedAt": "2023-11-01T15:09:51.891Z",
      "metadata": {
        "0": "14",
        "1": "208.79999999999998",
        "2": "database-1-instance-1",
        "3": "db.r5.large",
        "4": "false",
        "5": "us-west-2",
```

```
        "6": "arn:aws:rds:us-west-2:000000000000:db:database-1-instance-1",
        "7": "1"
    },
    "recommendationArn": "arn:aws:trustedadvisor:::organization-
recommendation/5a694939-2e54-45a2-ae72-730598fa89d0",
    "regionCode": "us-west-2",
    "status": "warning"
},
{
    "arn": "arn:aws:trustedadvisor:::000000000000:recommendation-
resource/5a694939-2e54-45a2-
ae72-730598fa89d0/51fded4d7a3278818df9cfe344ff5762cec46c095a6763d1ba1ba53bd0e1b0e6",
    "awsResourceId": "database-1",
    "id":
"51fded4d7a3278818df9cfe344ff5762cec46c095a6763d1ba1ba53bd0e1b0e6",
    "lastUpdatedAt": "2023-11-01T15:09:51.891Z",
    "metadata": {
        "0": "14",
        "1": "31.679999999999996",
        "2": "database-1",
        "3": "db.t3.small",
        "4": "false",
        "5": "us-west-2",
        "6": "arn:aws:rds:us-west-2:000000000000:db:database-1",
        "7": "20"
    },
    "recommendationArn": "arn:aws:trustedadvisor:::organization-
recommendation/5a694939-2e54-45a2-ae72-730598fa89d0",
    "regionCode": "us-west-2",
    "status": "warning"
},
{
    "arn": "arn:aws:trustedadvisor:::000000000000:recommendation-
resource/5a694939-2e54-45a2-ae72-730598fa89d0/
f4d01bd20f4cd5372062aaafc8786c489e48f0ead7cdab121463bf9f89e40a36b",
    "awsResourceId": "database-2-instance-1-us-west-2a",
    "id":
"f4d01bd20f4cd5372062aaafc8786c489e48f0ead7cdab121463bf9f89e40a36b",
    "lastUpdatedAt": "2023-11-01T15:09:51.891Z",
    "metadata": {
        "0": "14",
        "1": "187.200000000000002",
        "2": "database-2-instance-1-us-west-2a",
        "3": "db.r6g.large",
```

```

        "4": "true",
        "5": "us-west-2",
        "6": "arn:aws:rds:us-west-2:000000000000:db:database-2-instance-1-
us-west-2a",
        "7": "1"
    },
    "recommendationArn": "arn:aws:trustedadvisor:::organization-
recommendation/5a694939-2e54-45a2-ae72-730598fa89d0",
    "regionCode": "us-west-2",
    "status": "warning"
},
],
"nextToken": "REDACTED"
}

```

有关更多信息，请参阅《[Trusted Advisor 用户指南](#)》中的 [Tru AWS sted Advisor API 入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListOrganizationRecommendationResources](#) 中的。

## list-organization-recommendations

以下代码示例演示了如何使用 `list-organization-recommendations`。

### AWS CLI

示例 1：列出组织推荐

以下 `list-organization-recommendations` 示例列出了所有组织建议，但不包括筛选条件。

```
aws trustedadvisor list-organization-recommendations
```

输出：

```

{
  "organizationRecommendationSummaries": [
    {
      "arn": "arn:aws:trustedadvisor:::organization-recommendation/9534ec9b-
bf3a-44e8-8213-2ed68b39d9d5",
      "name": "Lambda Runtime Deprecation Warning",
      "awsServices": [
        "lambda"
      ],

```

```

    "checkArn": "arn:aws:trustedadvisor:::check/L4dfs2Q4C5",
    "id": "9534ec9b-bf3a-44e8-8213-2ed68b39d9d5",
    "lifecycleStage": "resolved",
    "pillars": [
      "security"
    ],
    "resourcesAggregates": {
      "errorCount": 0,
      "okCount": 0,
      "warningCount": 0
    },
    "source": "ta_check",
    "status": "warning",
    "type": "priority"
  },
  {
    "arn": "arn:aws:trustedadvisor:::organization-
recommendation/4ecff4d4-1bc1-4c99-a5b8-0fff9ee500d6",
    "name": "Lambda Runtime Deprecation Warning",
    "awsServices": [
      "lambda"
    ],
    "checkArn": "arn:aws:trustedadvisor:::check/L4dfs2Q4C5",
    "id": "4ecff4d4-1bc1-4c99-a5b8-0fff9ee500d6",
    "lifecycleStage": "resolved",
    "pillars": [
      "security"
    ],
    "resourcesAggregates": {
      "errorCount": 0,
      "okCount": 0,
      "warningCount": 0
    },
    "source": "ta_check",
    "status": "warning",
    "type": "priority"
  },
],
"nextToken": "REDACTED"
}

```

有关更多信息，请参阅 [《Trusted Advisor 用户指南》中的 Tru AWS sted Advisor API 入门](#)。

示例 2：使用筛选器列出组织推荐

以下`list-organization-recommendations`示例筛选并返回最多一项属于“安全”支柱的组织建议。

```
aws trustedadvisor list-organization-recommendations \  
  --pillar security \  
  --max-items 100
```

输出：

```
{  
  "organizationRecommendationSummaries": [{  
    "arn": "arn:aws:trustedadvisor::organization-recommendation/9534ec9b-  
bf3a-44e8-8213-2ed68b39d9d5",  
    "name": "Lambda Runtime Deprecation Warning",  
    "awsServices": [  
      "lambda"  
    ],  
    "checkArn": "arn:aws:trustedadvisor::check/L4dfs2Q4C5",  
    "id": "9534ec9b-bf3a-44e8-8213-2ed68b39d9d5",  
    "lifecycleStage": "resolved",  
    "pillars": [  
      "security"  
    ],  
    "resourcesAggregates": {  
      "errorCount": 0,  
      "okCount": 0,  
      "warningCount": 0  
    },  
    "source": "ta_check",  
    "status": "warning",  
    "type": "priority"  
  }],  
  "nextToken": "REDACTED"  
}
```

有关更多信息，请参阅 [《Trusted Advisor 用户指南》中的 Tru AWS sted Advisor API 入门](#)。

示例 3：使用分页令牌列出组织推荐

以下`list-organization-recommendations`示例使用从上一个请求返回的“nextToken”来获取下一页的组织推荐。

```
aws trustedadvisor list-organization-recommendations \  
  --next-token REDACTED
```

```
--pillar security \  
--max-items 100 \  
--starting-token <next-token>
```

输出：

```
{  
  "organizationRecommendationSummaries": [{  
    "arn": "arn:aws:trustedadvisor::organization-  
recommendation/4ecff4d4-1bc1-4c99-a5b8-0fff9ee500d6",  
    "name": "Lambda Runtime Deprecation Warning",  
    "awsServices": [  
      "lambda"  
    ],  
    "checkArn": "arn:aws:trustedadvisor::check/L4dfs2Q4C5",  
    "id": "4ecff4d4-1bc1-4c99-a5b8-0fff9ee500d6",  
    "lifecycleStage": "resolved",  
    "pillars": [  
      "security"  
    ],  
    "resourcesAggregates": {  
      "errorCount": 0,  
      "okCount": 0,  
      "warningCount": 0  
    },  
    "source": "ta_check",  
    "status": "warning",  
    "type": "priority"  
  }]  
}
```

有关更多信息，请参阅 [《Trusted Advisor 用户指南》中的 Tru AWS sted Advisor API 入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListOrganizationRecommendations](#) 中的。

## list-recommendation-resources

以下代码示例演示了如何使用 list-recommendation-resources。

### AWS CLI

#### 列出推荐资源



以下 `list-recommendation-resources` 示例按标识符列出了推荐的所有资源。

```
aws trustedadvisor list-recommendation-resources \
  --recommendation-identifier
  arn:aws:trustedadvisor::000000000000:recommendation/55fa4d2e-
  bbb7-491a-833b-5773e9589578
```

输出：

```
{
  "recommendationResourceSummaries": [
    {
      "arn": "arn:aws:trustedadvisor::000000000000:recommendation-
      resource/55fa4d2e-
      bbb7-491a-833b-5773e9589578/18959a1f1973cff8e706e9d9bde28bba36cd602a6b2cb86c8b61252835236010",
      "id":
      "18959a1f1973cff8e706e9d9bde28bba36cd602a6b2cb86c8b61252835236010",
      "awsResourceId": "webcms-dev-01",
      "lastUpdatedAt": "2023-11-01T15:09:51.891Z",
      "metadata": {
        "0": "14",
        "1": "123.12000000000002",
        "2": "webcms-dev-01",
        "3": "db.m6i.large",
        "4": "false",
        "5": "us-east-1",
        "6": "arn:aws:rds:us-east-1:000000000000:db:webcms-dev-01",
        "7": "20"
      },
      "recommendationArn":
      "arn:aws:trustedadvisor::000000000000:recommendation/55fa4d2e-
      bbb7-491a-833b-5773e9589578",
      "regionCode": "us-east-1",
      "status": "warning"
    },
    {
      "arn": "arn:aws:trustedadvisor::000000000000:recommendation-
      resource/55fa4d2e-bbb7-491a-833b-5773e9589578/
      e6367ff500ac90db8e4adeb4892e39ee9c36bbf812dcbce4b9e4fefcec9eb63e",
      "id":
      "e6367ff500ac90db8e4adeb4892e39ee9c36bbf812dcbce4b9e4fefcec9eb63e",
      "awsResourceId": "aws-dev-db-stack-instance-1",
      "lastUpdatedAt": "2023-11-01T15:09:51.891Z",
```

```

      "metadata": {
        "0": "14",
        "1": "29.52",
        "2": "aws-dev-db-stack-instance-1",
        "3": "db.t2.small",
        "4": "false",
        "5": "us-east-1",
        "6": "arn:aws:rds:us-east-1:000000000000:db:aws-dev-db-stack-
instance-1",
        "7": "1"
      },
      "recommendationArn":
"arn:aws:trustedadvisor::000000000000:recommendation/55fa4d2e-
bbb7-491a-833b-5773e9589578",
      "regionCode": "us-east-1",
      "status": "warning"
    },
    {
      "arn": "arn:aws:trustedadvisor::000000000000:recommendation-
resource/55fa4d2e-
bbb7-491a-833b-5773e9589578/31aa78ba050a5015d2d38cca7f5f1ce88f70857c4e1c3ad03f8f9fd95dad7459",
      "id":
"31aa78ba050a5015d2d38cca7f5f1ce88f70857c4e1c3ad03f8f9fd95dad7459",
      "awsResourceId": "aws-awesome-apps-stack-db",
      "lastUpdatedAt": "2023-11-01T15:09:51.891Z",
      "metadata": {
        "0": "14",
        "1": "114.48000000000002",
        "2": "aws-awesome-apps-stack-db",
        "3": "db.m6g.large",
        "4": "false",
        "5": "us-east-1",
        "6": "arn:aws:rds:us-east-1:000000000000:db:aws-awesome-apps-stack-
db",
        "7": "100"
      },
      "recommendationArn":
"arn:aws:trustedadvisor::000000000000:recommendation/55fa4d2e-
bbb7-491a-833b-5773e9589578",
      "regionCode": "us-east-1",
      "status": "warning"
    }
  ],
  "nextToken": "REDACTED"

```

```
}
```

有关更多信息，请参阅《[Trusted Advisor 用户指南](#)》中的 [Tru AWS sted Advisor API 入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListRecommendationResources](#) 中的。

## list-recommendations

以下代码示例演示了如何使用 list-recommendations。

### AWS CLI

示例 1：列出推荐

以下 list-recommendations 示例列出了所有建议，但不包括筛选条件。

```
aws trustedadvisor list-recommendations
```

输出：

```
{
  "recommendationSummaries": [
    {
      "arn": "arn:aws:trustedadvisor::000000000000:recommendation/55fa4d2e-
bbb7-491a-833b-5773e9589578",
      "name": "MFA Recommendation",
      "awsServices": [
        "iam"
      ],
      "checkArn": "arn:aws:trustedadvisor:::check/7DAFEemoDos",
      "id": "55fa4d2e-bbb7-491a-833b-5773e9589578",
      "lastUpdatedAt": "2023-11-01T15:57:58.673Z",
      "pillarSpecificAggregates": {
        "costOptimizing": {
          "estimatedMonthlySavings": 0.0,
          "estimatedPercentMonthlySavings": 0.0
        }
      },
      "pillars": [
        "security"
      ],
      "resourcesAggregates": {
```

```

        "errorCount": 1,
        "okCount": 0,
        "warningCount": 0
    },
    "source": "ta_check",
    "status": "error",
    "type": "standard"
},
{
    "arn":
"arn:aws:trustedadvisor::000000000000:recommendation/8b602b6f-452d-4cb2-8a9e-
c7650955d9cd",
    "name": "RDS clusters quota warning",
    "awsServices": [
        "rds"
    ],
    "checkArn": "arn:aws:trustedadvisor:::check/gjqMBn6pjz",
    "id": "8b602b6f-452d-4cb2-8a9e-c7650955d9cd",
    "lastUpdatedAt": "2023-11-01T15:58:17.397Z",
    "pillarSpecificAggregates": {
        "costOptimizing": {
            "estimatedMonthlySavings": 0.0,
            "estimatedPercentMonthlySavings": 0.0
        }
    },
    "pillars": [
        "service_limits"
    ],
    "resourcesAggregates": {
        "errorCount": 0,
        "okCount": 3,
        "warningCount": 6
    },
    "source": "ta_check",
    "status": "warning",
    "type": "standard"
}
],
"nextToken": "REDACTED"
}

```

有关更多信息，请参阅 [《Trusted Advisor 用户指南》中的 Tru AWS sted Advisor API 入门](#)。

示例 2：使用筛选器列出推荐

以下 `list-recommendations` 示例列出了推荐并包含一个筛选条件。

```
aws trustedadvisor list-recommendations \  
  --aws-service iam \  
  --max-items 100
```

输出：

```
{  
  "recommendationSummaries": [{  
    "arn": "arn:aws:trustedadvisor::000000000000:recommendation/55fa4d2e-  
bbb7-491a-833b-5773e9589578",  
    "name": "MFA Recommendation",  
    "awsServices": [  
      "iam"  
    ],  
    "checkArn": "arn:aws:trustedadvisor:::check/7DAFEemoDos",  
    "id": "55fa4d2e-bbb7-491a-833b-5773e9589578",  
    "lastUpdatedAt": "2023-11-01T15:57:58.673Z",  
    "pillarSpecificAggregates": {  
      "costOptimizing": {  
        "estimatedMonthlySavings": 0.0,  
        "estimatedPercentMonthlySavings": 0.0  
      }  
    },  
    "pillars": [  
      "security"  
    ],  
    "resourcesAggregates": {  
      "errorCount": 1,  
      "okCount": 0,  
      "warningCount": 0  
    },  
    "source": "ta_check",  
    "status": "error",  
    "type": "standard"  
  }],  
  "nextToken": "REDACTED"  
}
```

有关更多信息，请参阅 [《Trusted Advisor 用户指南》中的 Tru AWS sted Advisor API 入门](#)。

示例 3：使用分页令牌列出推荐

以下list-recommendations示例使用从上一个请求返回的“nextToken”来获取下一页经过筛选的推荐。

```
aws trustedadvisor list-recommendations \  
  --aws-service rds \  
  --max-items 100 \  
  --starting-token <next-token>
```

输出：

```
{  
  "recommendationSummaries": [{  
    "arn":  
    "arn:aws:trustedadvisor::000000000000:recommendation/8b602b6f-452d-4cb2-8a9e-c7650955d9cd",  
    "name": "RDS clusters quota warning",  
    "awsServices": [  
      "rds"  
    ],  
    "checkArn": "arn:aws:trustedadvisor:::check/gjqMBn6pjz",  
    "id": "8b602b6f-452d-4cb2-8a9e-c7650955d9cd",  
    "lastUpdatedAt": "2023-11-01T15:58:17.397Z",  
    "pillarSpecificAggregates": {  
      "costOptimizing": {  
        "estimatedMonthlySavings": 0.0,  
        "estimatedPercentMonthlySavings": 0.0  
      }  
    },  
    "pillars": [  
      "service_limits"  
    ],  
    "resourcesAggregates": {  
      "errorCount": 0,  
      "okCount": 3,  
      "warningCount": 6  
    },  
    "source": "ta_check",  
    "status": "warning",  
    "type": "standard"  
  }]  
}
```

有关更多信息，请参阅 [《Trusted Advisor 用户指南》](#) 中的 [Tru AWS sted Advisor API 入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRecommendations](#)中的。

## update-organization-recommendation-lifecycle

以下代码示例演示了如何使用 update-organization-recommendation-lifecycle。

### AWS CLI

#### 更新组织推荐生命周期

以下update-organization-recommendation-lifecycle示例按标识符更新组织推荐的生命周期。

```
aws trustedadvisor update-organization-recommendation-lifecycle \  
  --organization-recommendation-identifier arn:aws:trustedadvisor:::organization-  
recommendation/96b5e5ca-7930-444c-90c6-06d386128100 \  
  --lifecycle-stage dismissed \  
  --update-reason-code not_applicable
```

此命令不生成任何输出。

有关更多信息，请参阅《[Trusted Advisor 用户指南](#)》中的 [Tru AWS sted Advisor API 入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateOrganizationRecommendationLifecycle](#)中的。

## update-recommendation-lifecycle

以下代码示例演示了如何使用 update-recommendation-lifecycle。

### AWS CLI

#### 更新推荐生命周期

以下update-recommendation-lifecycle示例按标识符更新推荐的生命周期。

```
aws trustedadvisor update-recommendation-lifecycle \  
  --recommendation-identifier  
arn:aws:trustedadvisor::000000000000:recommendation/861c9c6e-  
f169-405a-8b59-537a8cacc7a \  
  --lifecycle-stage resolved \  

```

```
--update-reason-code valid_business_case
```

此命令不生成任何输出。

有关更多信息，请参阅《[Trusted Advisor 用户指南](#)》中的 [Tru AWS sted Advisor API 入门](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRecommendationLifecycle](#)中的。

## 使用验证权限示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 具有已验证权限的，来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **create-identity-source**

以下代码示例演示了如何使用 create-identity-source。

AWS CLI

创建身份源

以下create-identity-source示例创建了一个身份源，允许您引用存储在指定 Amazon Cognito 用户池中的身份。这些身份作为实体类型在“已验证权限”中可用User。

```
aws verifiedpermissions create-identity-source \  
  --configuration file://config.txt \  
  --principal-entity-type "User" \  
  --policy-store-id PSEXAMPLEabcdefg111111
```



config.txt 的内容：

```
{
  "cognitoUserPoolConfiguration": {
    "userPoolArn": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-
west-2_1a2b3c4d5",
    "clientIds":["a1b2c3d4e5f6g7h8i9j0kalbmc"]
  }
}
```

输出：

```
{
  "createdDate": "2023-05-19T20:30:28.214829+00:00",
  "identitySourceId": "ISEXAMPLEabcdefgh111111",
  "lastUpdatedDate": "2023-05-19T20:30:28.214829+00:00",
  "policyStoreId": "PSEXAMPLEabcdefgh111111"
}
```

有关身份源的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[对身份提供商使用亚马逊验证权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateIdentitySource](#)中的。

## create-policy-store

以下代码示例演示了如何使用 create-policy-store。

AWS CLI

创建策略存储

以下create-policy-store示例在当前 AWS 区域中创建策略存储。

```
aws verifiedpermissions create-policy-store \
  --validation-settings "mode=STRICT"
```

输出：

```
{
```

```
"arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEEabcdefg111111",
  "createdDate": "2023-05-16T17:41:29.103459+00:00",
  "lastUpdatedDate": "2023-05-16T17:41:29.103459+00:00",
  "policyStoreId": "PSEXAMPLEEabcdefg111111"
}
```

有关策略存储的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[亚马逊验证权限策略存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePolicyStore](#)中的。

## create-policy-template

以下代码示例演示了如何使用 create-policy-template。

### AWS CLI

#### 示例 1：创建策略模板

以下create-policy-template示例使用包含委托人占位符的语句创建策略模板。

```
aws verifiedpermissions create-policy-template \
  --definition file://template1.txt \
  --policy-store-id PSEXAMPLEEabcdefg111111
```

template1.txt 文件的内容：

```
permit(
  principal in ?principal,
  action == Action::"view",
  resource == Photo::"VacationPhoto94.jpg"
);
```

输出：

```
{
  "createdDate": "2023-06-12T20:47:42.804511+00:00",
  "lastUpdatedDate": "2023-06-12T20:47:42.804511+00:00",
  "policyStoreId": "PSEXAMPLEEabcdefg111111",
  "policyTemplateId": "PTEXAMPLEEabcdefg111111"
}
```

有关策略模板的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[亚马逊验证权限策略模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePolicyTemplate](#)中的。

## create-policy

以下代码示例演示了如何使用 create-policy。

### AWS CLI

#### 示例 1：创建静态策略

以下create-policy示例创建了一个静态策略，其策略范围既指定了委托人，又指定了资源。

```
aws verifiedpermissions create-policy \  
  --definition file://definition1.txt \  
  --policy-store-id PSEXAMPLEEabcdefg111111
```

definition1.txt 文件的内容：

```
{  
  "static": {  
    "description": "Grant everyone of janeFriends UserGroup access to the  
vacationFolder Album",  
    "statement": "permit(principal in UserGroup::\\"janeFriends\\", action,  
resource in Album::\\"vacationFolder\" );"  
  }  
}
```

输出：

```
{  
  "createdDate": "2023-06-12T20:33:37.382907+00:00",  
  "lastUpdatedDate": "2023-06-12T20:33:37.382907+00:00",  
  "policyId": "SPEXAMPLEEabcdefg111111",  
  "policyStoreId": "PSEXAMPLEEabcdefg111111",  
  "policyType": "STATIC",  
  "principal": {  
    "entityId": "janeFriends",  
    "entityType": "UserGroup"  
  },  
  "resource": {
```

```
    "entityId": "vacationFolder",
    "entityType": "Album"
  }
}
```

### 示例 2：创建向所有人授予资源访问权限的静态策略

以下create-policy示例创建了一个静态策略，其策略范围仅指定资源。

```
aws verifiedpermissions create-policy \
  --definition file://definition2.txt \
  --policy-store-id PSEXAMPLEEabcdefg111111
```

definition2.txt 文件的内容：

```
{
  "static": {
    "description": "Grant everyone access to the publicFolder Album",
    "statement": "permit(principal, action, resource in Album:\""publicFolder
  \");"
  }
}
```

输出：

```
{
  "createdDate": "2023-06-12T20:39:44.975897+00:00",
  "lastUpdatedDate": "2023-06-12T20:39:44.975897+00:00",
  "policyId": "PbfR73F8oh5MMfr9uRtFDB",
  "policyStoreId": "PSEXAMPLEEabcdefg222222",
  "policyType": "STATIC",
  "resource": {
    "entityId": "publicFolder",
    "entityType": "Album"
  }
}
```

### 示例 3：创建与指定模板关联的模板关联策略

以下create-policy示例使用指定的策略模板创建模板关联策略，并将指定的委托人与新的模板关联策略关联起来。

```
aws verifiedpermissions create-policy \  
  --definition file://definition.txt \  
  --policy-store-id PSEXAMPLEEabcdefg111111
```

definition.txt 的内容：

```
{  
  "templateLinked": {  
    "policyTemplateId": "PTEXAMPLEEabcdefg111111",  
    "principal": {  
      "entityType": "User",  
      "entityId": "alice"  
    }  
  }  
}
```

输出：

```
{  
  "createdDate": "2023-06-12T20:49:51.490211+00:00",  
  "lastUpdatedDate": "2023-06-12T20:49:51.490211+00:00",  
  "policyId": "TPEXAMPLEEabcdefg111111",  
  "policyStoreId": "PSEXAMPLEEabcdefg111111",  
  "policyType": "TEMPLATE_LINKED",  
  "principal": {  
    "entityId": "alice",  
    "entityType": "User"  
  },  
  "resource": {  
    "entityId": "VacationPhoto94.jpg",  
    "entityType": "Photo"  
  }  
}
```

有关政策的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[亚马逊验证权限政策](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreatePolicy](#) 中的。

## delete-identity-source

以下代码示例演示了如何使用 delete-identity-source。

## AWS CLI

### 删除身份源

以下delete-identity-source示例删除具有指定 ID 的身份源。

```
aws verifiedpermissions delete-identity-source \  
  --identity-source-id ISEXAMPLEEabcdefg111111 \  
  --policy-store-id PSEXAMPLEEabcdefg111111
```

此命令不生成任何输出。

有关身份源的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[对身份提供商使用亚马逊验证权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteIdentitySource](#)中的。

## delete-policy-store

以下代码示例演示了如何使用 delete-policy-store。

### AWS CLI

#### 删除策略存储

以下delete-policy-store示例删除具有指定 ID 的策略存储。

```
aws verifiedpermissions delete-policy-store \  
  --policy-store-id PSEXAMPLEEabcdefg111111
```

此命令不生成任何输出。

有关策略存储的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[亚马逊验证权限策略存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePolicyStore](#)中的。

## delete-policy-template

以下代码示例演示了如何使用 delete-policy-template。

### AWS CLI

#### 删除策略模板

以下delete-policy-template示例删除具有指定 ID 的策略模板。

```
aws verifiedpermissions delete-policy \  
  --policy-template-id PTEXAMPLEabcdefg111111 \  
  --policy-store-id PSEXAMPLEabcdefg111111
```

此命令不生成任何输出。

有关策略模板的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[亚马逊验证权限策略模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePolicyTemplate](#)中的。

## delete-policy

以下代码示例演示了如何使用 delete-policy。

### AWS CLI

删除静态或与模板关联的策略

以下delete-policy示例删除具有指定 ID 的策略。

```
aws verifiedpermissions delete-policy \  
  --policy-id SPEXAMPLEabcdefg111111 \  
  --policy-store-id PSEXAMPLEabcdefg111111
```

此命令不生成任何输出。

有关政策的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[亚马逊验证权限政策](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePolicy](#)中的。

## get-identity-source

以下代码示例演示了如何使用 get-identity-source。

### AWS CLI

检索有关身份源的详细信息

以下get-identity-source示例显示具有指定 ID 的身份源的详细信息。

```
aws verifiedpermissions get-identity-source \  
  --identity-source ISEXAMPLEabcdefg111111 \  
  --policy-store-id PSEXAMPLEabcdefg111111
```

输出：

```
{  
  "createdDate": "2023-06-12T22:27:49.150035+00:00",  
  "details": {  
    "clientIds": [ "a1b2c3d4e5f6g7h8i9j0kalbmc" ],  
    "discoveryUrl": "https://cognito-idp.us-west-2.amazonaws.com/us-  
west-2_1a2b3c4d5",  
    "openIdIssuer": "COGNITO",  
    "userPoolArn": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-  
west-2_1a2b3c4d5"  
  },  
  "identitySourceId": "ISEXAMPLEabcdefg111111",  
  "lastUpdatedDate": "2023-06-12T22:27:49.150035+00:00",  
  "policyStoreId": "PSEXAMPLEabcdefg111111",  
  "principalEntityType": "User"  
}
```

有关身份源的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[对身份提供商使用亚马逊验证权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetIdentitySource](#)中的。

## get-policy-store

以下代码示例演示了如何使用 get-policy-store。

### AWS CLI

检索有关策略存储的详细信息

以下get-policy-store示例显示了具有指定 ID 的策略存储的详细信息。

```
aws verifiedpermissions get-policy-store \  
  --policy-store-id PSEXAMPLEabcdefg111111
```

输出：



```
{
  "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111",
  "createdDate": "2023-06-05T20:16:46.225598+00:00",
  "lastUpdatedDate": "2023-06-08T20:40:23.173691+00:00",
  "policyStoreId": "PSEXAMPLEabcdefg111111",
  "validationSettings": { "mode": "OFF" }
}
```

有关策略存储的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[亚马逊验证权限策略存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPolicyStore](#)中的。

## get-policy-template

以下代码示例演示了如何使用 get-policy-template。

### AWS CLI

检索有关策略模板的详细信息

以下get-policy-template示例显示了具有指定 ID 的策略模板的详细信息。

```
aws verifiedpermissions get-policy-template \
  --policy-template-id PTEXAMPLEabcdefg111111 \
  --policy-store-id PSEXAMPLEabcdefg111111
```

输出：

```
{
  "createdDate": "2023-06-12T20:47:42.804511+00:00",
  "lastUpdatedDate": "2023-06-12T20:47:42.804511+00:00",
  "policyStoreId": "PSEXAMPLEabcdefg111111",
  "policyTemplateId": "PTEXAMPLEabcdefg111111",
  "statement": "permit(\n    principal in ?principal,\n    action == Action::\n    \"view\", \n    resource == Photo::\"VacationPhoto94.jpg\" \n);"
}
```

有关策略模板的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[亚马逊验证权限策略模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPolicyTemplate](#)中的。

## get-policy

以下代码示例演示了如何使用 `get-policy`。

### AWS CLI

检索有关政策的详细信息

以下`get-policy`示例显示了具有指定 ID 的策略的详细信息。

```
aws verifiedpermissions get-policy \  
  --policy-id PSEXAMPLEabcdefg111111 \  
  --policy-store-id PSEXAMPLEabcdefg111111
```

输出：

```
{  
  "createdDate": "2023-06-12T20:33:37.382907+00:00",  
  "definition": {  
    "static": {  
      "description": "Grant everyone of janeFriends UserGroup access to the  
vacationFolder Album",  
      "statement": "permit(principal in UserGroup::\"janeFriends\", action,  
resource in Album::\"vacationFolder\" );"  
    }  
  },  
  "lastUpdatedDate": "2023-06-12T20:33:37.382907+00:00",  
  "policyId": "SPEXAMPLEabcdefg111111",  
  "policyStoreId": "PSEXAMPLEabcdefg111111",  
  "policyType": "STATIC",  
  "principal": {  
    "entityId": "janeFriends",  
    "entityType": "UserGroup"  
  },  
  "resource": {  
    "entityId": "vacationFolder",  
    "entityType": "Album"  
  }  
}
```

有关政策的更多信息，请参阅 [《亚马逊验证权限用户指南》中的亚马逊验证权限政策](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetPolicy](#)中的。

## get-schema

以下代码示例演示了如何使用 `get-schema`。

### AWS CLI

在策略存储中检索架构

以下 `get-schema` 示例显示了指定策略存储中架构的详细信息。

```
aws verifiedpermissions get-schema \  
  --policy-store-id PSEXAMPLEEabcdefg111111
```

输出：

```
{  
  "policyStoreId": "PSEXAMPLEEabcdefg111111",  
  "schema": "{\n    \"MySampleNamespace\":{\n      \"entityTypes\":{\n        \"Employee\":{\n          \"shape\":{\n            \"attributes\":{\n              \"jobLevel\":{\n                \"type\": \"Long\"\n              },\n              \"name\":{\n                \"type\": \"String\"\n              }\n            },\n            \"type\": \"Record\"\n          }\n        },\n        \"actions\":{\n          \"remoteAccess\":{\n            \"appliesTo\":{\n              \"principalTypes\":[\n                \"Employee\"\n              ]\n            }\n          }\n        }\n      }\n    }",  
  "createdDate": "2023-06-14T17:47:13.999885+00:00",  
  "lastUpdatedDate": "2023-06-14T17:47:13.999885+00:00"  
}
```

有关架构的更多信息，请参阅《Amazon 验证权限用户指南》中的[策略存储架构](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetSchema](#) 中的。

## is-authorized-with-token

以下代码示例演示了如何使用 `is-authorized-with-token`。

### AWS CLI

示例 1：请求对用户请求做出授权决定（允许）

以下 `is-authorized-with-token` 示例请求已通过 Amazon Cognito 身份验证的用户做出授权决定。该请求使用 Cognito 提供的身份令牌而不是访问令牌。在此示例中，将指定的信息存储配置为将主体作为类型 `CognitoUser` 实体返回。

```
aws verifiedpermissions is-authorized-with-token \  
  --policy-store-id PSEXAMPLEEabcdefg111111
```

```
--action actionId="View",actionType="Action" \
--resource entityId="vacationPhoto94.jpg",entityType="Photo" \
--policy-store-id PSEXAMPLEEabcdefg111111 \
--identity-token "AbCdE12345...long.string...54321EdCbA"
```

策略存储区包含一个带有以下语句的策略，该策略接受来自指定 Cognito 用户池和应用程序 ID 的身份。

```
permit(
  principal == CognitoUser::"us-east-1_1a2b3c4d5|a1b2c3d4e5f6g7h8i9j0ka1bmc",
  action,
  resource == Photo::"VacationPhoto94.jpg"
);
```

输出：

```
{
  "decision":"Allow",
  "determiningPolicies":[
    {
      "determiningPolicyId":"SPEXAMPLEEabcdefg111111"
    }
  ],
  "errors":[]
}
```

有关使用 Cognito 用户池中的身份的更多信息，请参阅 [《亚马逊验证权限用户指南》](#) 中的 [对身份提供商使用亚马逊验证权限](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [IsAuthorizedWithToken](#) 中的。

## is-authorized

以下代码示例演示了如何使用 is-authorized。

### AWS CLI

示例 1：请求对用户请求做出授权决定（允许）

以下 is-authorized 示例请求类型为的委托人做出授权决定 Alice，该委托人想要对 User 名为的类型 Photo 为的资源执行 updatePhoto 操作 VacationPhoto94.jpg。

响应显示一个策略允许该请求。

```
aws verifiedpermissions is-authorized \  
  --principal entityType=User,entityId=alice \  
  --action actionType=Action,actionId=view \  
  --resource entityType=Photo,entityId=VactionPhoto94.jpg \  
  --policy-store-id PSEXAMPLEEabcdefg111111
```

输出：

```
{  
  "decision": "ALLOW",  
  "determiningPolicies": [  
    {  
      "policyId": "SPEXAMPLEEabcdefg111111"  
    }  
  ],  
  "errors": []  
}
```

示例 2：请求对用户请求做出授权决定（拒绝）

以下示例与前面的示例相同，唯一的区别是委托人是 `User::"Bob"`。策略存储区不包含任何允许该用户访问的策略 `Album::"alice_folder"`。

输出表明 `Deny` 是隐式的，因为列表 `DeterminingPolicies` 为空。

```
aws verifiedpermissions create-policy \  
  --definition file://definition2.txt \  
  --policy-store-id PSEXAMPLEEabcdefg111111
```

输出：

```
{  
  "decision": "DENY",  
  "determiningPolicies": [],  
  "errors": []  
}
```

有关更多信息，请参阅 [《Amazon 验证权限用户指南》](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[IsAuthorized](#)中的。

## list-identity-sources

以下代码示例演示了如何使用 `list-identity-sources`。

### AWS CLI

列出可用的身份来源

以下`list-identity-sources`示例列出了指定策略存储中的所有身份源。

```
aws verifiedpermissions list-identity-sources \  
  --policy-store-id PSEXAMPLEEabcdefg111111
```

输出：

```
{  
  "identitySources": [  
    {  
      "createdDate": "2023-06-12T22:27:49.150035+00:00",  
      "details": {  
        "clientIds": [ "a1b2c3d4e5f6g7h8i9j0kalbmc" ],  
        "discoveryUrl": "https://cognito-idp.us-west-2.amazonaws.com/us-west-2_1a2b3c4d5",  
        "openIdIssuer": "COGNITO",  
        "userPoolArn": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-west-2_1a2b3c4d5"  
      },  
      "identitySourceId": "ISEXAMPLEEabcdefg111111",  
      "lastUpdatedDate": "2023-06-12T22:27:49.150035+00:00",  
      "policyStoreId": "PSEXAMPLEEabcdefg111111",  
      "principalEntityType": "User"  
    }  
  ]  
}
```

有关身份源的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[对身份提供商使用亚马逊验证权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListIdentitySources](#)中的。

## list-policies

以下代码示例演示了如何使用 `list-policies`。

### AWS CLI

#### 列出可用策略

以下 `list-policies` 示例列出了指定策略存储区中的所有策略。

```
aws verifiedpermissions list-policies \  
  --policy-store-id PSEXAMPLEabcdefg111111
```

输出：

```
{  
  "policies": [  
    {  
      "createdDate": "2023-06-12T20:33:37.382907+00:00",  
      "definition": {  
        "static": {  
          "description": "Grant everyone of janeFriends UserGroup access  
to the vacationFolder Album"  
        }  
      },  
      "lastUpdatedDate": "2023-06-12T20:33:37.382907+00:00",  
      "policyId": "SPEXAMPLEabcdefg111111",  
      "policyStoreId": "PSEXAMPLEabcdefg111111",  
      "policyType": "STATIC",  
      "principal": {  
        "entityId": "janeFriends",  
        "entityType": "UserGroup"  
      },  
      "resource": {  
        "entityId": "vacationFolder",  
        "entityType": "Album"  
      }  
    },  
    {  
      "createdDate": "2023-06-12T20:39:44.975897+00:00",  
      "definition": {  
        "static": {  
          "description": "Grant everyone access to the publicFolder Album"  
        }  
      }  
    }  
  ]  
}
```

```

    },
    "lastUpdatedDate": "2023-06-12T20:39:44.975897+00:00",
    "policyId": "SPEXAMPLEabcdefg222222",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyType": "STATIC",
    "resource": {
      "entityId": "publicFolder",
      "entityType": "Album"
    }
  },
  {
    "createdDate": "2023-06-12T20:49:51.490211+00:00",
    "definition": {
      "templateLinked": {
        "policyTemplateId": "PTEXAMPLEabcdefg111111"
      }
    },
    "lastUpdatedDate": "2023-06-12T20:49:51.490211+00:00",
    "policyId": "SPEXAMPLEabcdefg333333",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyType": "TEMPLATE_LINKED",
    "principal": {
      "entityId": "alice",
      "entityType": "User"
    },
    "resource": {
      "entityId": "VacationPhoto94.jpg",
      "entityType": "Photo"
    }
  }
]
}

```

有关政策的更多信息，请参阅 [《亚马逊验证权限用户指南》中的亚马逊验证权限政策](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListPolicies](#) 中的。

## list-policy-stores

以下代码示例演示了如何使用 `list-policy-stores`。

### AWS CLI

列出可用的策略存储库



以下 `list-policy-stores` 示例列出了该 AWS 地区的所有策略存储。除了 `create-policy-store` 和 `list-policy-stores` 要求您指定要使用的策略存储的 ID 之外，所有用于验证权限的命令除外。

```
aws verifiedpermissions list-policy-stores
```

输出：

```
{
  "policyStores": [
    {
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111",
      "createdDate": "2023-06-05T20:16:46.225598+00:00",
      "policyStoreId": "PSEXAMPLEabcdefg111111"
    },
    {
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg222222",
      "createdDate": "2023-06-08T18:09:37.364356+00:00",
      "policyStoreId": "PSEXAMPLEabcdefg222222"
    },
    {
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg333333",
      "createdDate": "2023-06-08T18:09:46.920600+00:00",
      "policyStoreId": "PSEXAMPLEabcdefg333333"
    }
  ]
}
```

有关策略存储的更多信息，请参阅 [《亚马逊验证权限用户指南》中的亚马逊验证权限策略存储](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListPolicyStores](#) 中的。

## list-policy-templates

以下代码示例演示了如何使用 `list-policy-templates`。

AWS CLI

列出可用的策略模板

以下 `list-policy-templates` 示例列出了指定策略存储区中的所有策略模板。

```
aws verifiedpermissions list-policy-templates \  
  --policy-store-id PSEXAMPLEEabcdefg111111
```

输出：

```
{  
  "policyTemplates": [  
    {  
      "createdDate": "2023-06-12T20:47:42.804511+00:00",  
      "lastUpdatedDate": "2023-06-12T20:47:42.804511+00:00",  
      "policyStoreId": "PSEXAMPLEEabcdefg111111",  
      "policyTemplateId": "PTEXAMPLEEabcdefg111111"  
    }  
  ]  
}
```

有关策略模板的更多信息，请参阅 [《亚马逊验证权限用户指南》中的亚马逊验证权限策略模板](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListPolicyTemplates](#) 中的。

## put-schema

以下代码示例演示了如何使用 `put-schema`。

### AWS CLI

将架构保存到策略存储中

以下 `put-schema` 示例创建或替换指定策略存储中的架构。

输入文件中的 `cedarJson` 参数采用 JSON 对象的字符串表示形式。它在最外面的引号对中包含嵌入式引号 (")。这要求您将 JSON 转换为字符串，方法是在所有嵌入的引号前面加上反斜杠字符 (\)，然后将所有行组合成一个不带换行符的文本行。

为了便于阅读，可以在此处将示例字符串分成多行显示，但该操作要求将参数作为单行字符串提交。

```
aws verifiedPermissions put-schema — 定义文件：//schema.txt — psexampleabcdefg111111  
policy-store-id
```

schema.txt 的内容：

```
{
  "cedarJson": "{\"MySampleNamespace\": {\"actions\": {\"remoteAccess\": {
    \"appliesTo\": {\"principalTypes\": [\"Employee\"]}},\"entityTypes\": {
    \"Employee\": {\"shape\": {\"attributes\": {\"jobLevel\": {\"type\":
    \"Long\"},\"name\": {\"type\": \"String\"}},\"type\": \"Record\"}}}}}"
}
```

输出：

```
{
  "policyStoreId": "PSEXAMPLEabcdefgh111111",
  "namespaces": [
    "MySampleNamespace"
  ],
  "createdDate": "2023-06-14T17:47:13.999885+00:00",
  "lastUpdatedDate": "2023-06-14T17:47:13.999885+00:00"
}
```

有关架构的更多信息，请参阅《Amazon 验证权限用户指南》中的[策略存储架构](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutSchema](#) 中的。

## update-identity-source

以下代码示例演示了如何使用 update-identity-source。

### AWS CLI

#### 更新身份源

以下 update-identity-source 示例通过提供新的 Cognito 用户池配置并更改身份源返回的实体类型来修改指定的身份源。

```
aws verifiedpermissions update-identity-source
  --identity-source-id ISEXAMPLEabcdefgh111111 \
  --update-configuration file://config.txt \
  --principal-entity-type "Employee" \
  --policy-store-id PSEXAMPLEabcdefgh111111
```

config.txt 的内容：

```
{
  "cognitoUserPoolConfiguration": {
    "userPoolArn": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-west-2_1a2b3c4d5",
    "clientIds": ["a1b2c3d4e5f6g7h8i9j0kalbmc"]
  }
}
```

输出：

```
{
  "createdDate": "2023-05-19T20:30:28.214829+00:00",
  "identitySourceId": "ISEXAMPLEabcdefg111111",
  "lastUpdatedDate": "2023-05-19T20:30:28.214829+00:00",
  "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

有关身份源的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[对身份提供商使用亚马逊验证权限](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateIdentitySource](#)中的。

## update-policy-store

以下代码示例演示了如何使用 update-policy-store。

### AWS CLI

#### 更新策略存储

以下update-policy-store示例通过更改策略存储的验证设置来修改策略存储。

```
aws verifiedpermissions update-policy-store \
  --validation-settings "mode=STRICT" \
  --policy-store-id PSEXAMPLEabcdefg111111
```

输出：

```
{
  "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/PSEXAMPLEabcdefg111111",
}
```

```
"createdDate": "2023-05-16T17:41:29.103459+00:00",  
"lastUpdatedDate": "2023-05-16T17:41:29.103459+00:00",  
"policyStoreId": "PSEXAMPLEEabcdefg111111"  
}
```

有关策略存储的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[亚马逊验证权限策略存储](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePolicyStore](#)中的。

## update-policy-template

以下代码示例演示了如何使用 update-policy-template。

### AWS CLI

#### 示例 1：更新策略模板

以下update-policy-template示例修改了指定的模板链接策略以替换其策略声明。

```
aws verifiedpermissions update-policy-template \  
  --policy-template-id PTEXAMPLEEabcdefg111111 \  
  --statement file://template1.txt \  
  --policy-store-id PSEXAMPLEEabcdefg111111
```

template1.txt 文件的内容：

```
permit(  
  principal in ?principal,  
  action == Action::"view",  
  resource == Photo::"VacationPhoto94.jpg"  
);
```

输出：

```
{  
  "createdDate": "2023-06-12T20:47:42.804511+00:00",  
  "lastUpdatedDate": "2023-06-12T20:47:42.804511+00:00",  
  "policyStoreId": "PSEXAMPLEEabcdefg111111",  
  "policyTemplateId": "PTEXAMPLEEabcdefg111111"  
}
```

有关策略模板的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[亚马逊验证权限策略模板](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdatePolicyTemplate](#)中的。

## update-policy

以下代码示例演示了如何使用 update-policy。

### AWS CLI

#### 示例 1：创建静态策略

以下create-policy示例创建了一个静态策略，其策略范围既指定了委托人，又指定了资源。

```
aws verifiedpermissions create-policy \  
  --definition file://definition.txt \  
  --policy-store-id PSEXAMPLEEabcdefg111111
```

该statement参数采用 JSON 对象的字符串表示形式。它是最外面的引号对中包含嵌入式引号 (")。这要求您将 JSON 转换为字符串，方法是在所有嵌入的引号前面加上反斜杠字符 (\)，然后将所有行组合成一个不带换行符的文本行。

为了便于阅读，可以在此处将示例字符串分成多行显示，但该操作要求将参数作为单行字符串提交。

definition.txt 文件的内容：

```
{  
  "static": {  
    "description": "Grant everyone of janeFriends UserGroup access to the  
vacationFolder Album",  
    "statement": "permit(principal in UserGroup:~\"janeFriends~\", action,  
resource in Album:~\"vacationFolder~\" );"  
  }  
}
```

输出：

```
{  
  "createdDate": "2023-06-12T20:33:37.382907+00:00",
```

```
"lastUpdatedDate": "2023-06-12T20:33:37.382907+00:00",
"policyId": "SPEXAMPLEabcdefghijklmnop111111",
"policyStoreId": "PSEXAMPLEabcdefghijklmnop111111",
"policyType": "STATIC",
"principal": {
  "entityId": "janeFriends",
  "entityType": "UserGroup"
},
"resource": {
  "entityId": "vacationFolder",
  "entityType": "Album"
}
}
```

## 示例 2：创建向所有人授予资源访问权限的静态策略

以下create-policy示例创建了一个静态策略，其策略范围仅指定资源。

```
aws verifiedpermissions create-policy \
  --definition file://definition2.txt \
  --policy-store-id PSEXAMPLEabcdefghijklmnop111111
```

definition2.txt 文件的内容：

```
{
  "static": {
    "description": "Grant everyone access to the publicFolder Album",
    "statement": "permit(principal, action, resource in Album:\""publicFolder
  \");"
  }
}
```

输出：

```
{
  "createdDate": "2023-06-12T20:39:44.975897+00:00",
  "lastUpdatedDate": "2023-06-12T20:39:44.975897+00:00",
  "policyId": "PbfR73F8oh5MMfr9uRtFDB",
  "policyStoreId": "PSEXAMPLEabcdefghijklmnop222222",
  "policyType": "STATIC",
  "resource": {
    "entityId": "publicFolder",
```

```
    "entityType": "Album"
  }
}
```

### 示例 3：创建与指定模板关联的模板关联策略

以下 `create-policy` 示例使用指定的策略模板创建模板关联策略，并将指定的委托人与新的模板关联策略关联起来。

```
aws verifiedpermissions create-policy \  
  --definition file://definition2.txt \  
  --policy-store-id PSEXAMPLEabcdefgh111111
```

definition3.txt 的内容：

```
{  
  "templateLinked": {  
    "policyTemplateId": "PTEXAMPLEabcdefgh111111",  
    "principal": {  
      "entityType": "User",  
      "entityId": "alice"  
    }  
  }  
}
```

输出：

```
{  
  "createdDate": "2023-06-12T20:49:51.490211+00:00",  
  "lastUpdatedDate": "2023-06-12T20:49:51.490211+00:00",  
  "policyId": "TPEXAMPLEabcdefgh111111",  
  "policyStoreId": "PSEXAMPLEabcdefgh111111",  
  "policyType": "TEMPLATE_LINKED",  
  "principal": {  
    "entityId": "alice",  
    "entityType": "User"  
  },  
  "resource": {  
    "entityId": "VacationPhoto94.jpg",  
    "entityType": "Photo"  
  }  
}
```



```
}
```

有关政策的更多信息，请参阅《[亚马逊验证权限用户指南](#)》中的[亚马逊验证权限政策](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdatePolicy](#) 中的。

## 使用的 VPC 莱迪思示例 AWS CLI

以下代码示例向您展示了如何使用 AWS Command Line Interface 与 VPC Lattice 搭配使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

### 操作

#### **create-listener**

以下代码示例演示了如何使用 `create-listener`。

#### AWS CLI

##### 创建监听器

以下 `create-listener` 示例使用默认规则创建一个 HTTPS 侦听器，该规则将流量转发到指定的 VPC Lattice 目标组。

```
aws vpc-lattice create-listener \  
  --name my-service-listener \  
  --protocol HTTPS \  
  --port 443 \  
  --service-identifier svc-0285b53b2eEXAMPLE \  
  --default-action file://listener-config.json
```

listener-config.json 的内容：

```
{
  "forward": {
    "targetGroups": [
      {
        "targetGroupIdentifier": "tg-0eaa4b9ab4EXAMPLE"
      }
    ]
  }
}
```

输出：

```
{
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:service/
svc-0285b53b2eEXAMPLE/listener/listener-07cc7fb0abEXAMPLE",
  "defaultAction": {
    "forward": {
      "targetGroups": [
        {
          "targetGroupIdentifier": "tg-0eaa4b9ab4EXAMPLE",
          "weight": 100
        }
      ]
    }
  },
  "id": "listener-07cc7fb0abEXAMPLE",
  "name": "my-service-listener",
  "port": 443,
  "protocol": "HTTPS",
  "serviceArn": "arn:aws:vpc-lattice:us-east-2:123456789012:service/
svc-0285b53b2eEXAMPLE",
  "serviceId": "svc-0285b53b2eEXAMPLE"
}
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的[监听器](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateListener](#) 中的。

## create-service-network-service-association

以下代码示例演示了如何使用 create-service-network-service-association。

## AWS CLI

### 创建服务关联

以下`create-service-network-service-association`示例将指定的服务与指定的服务网络相关联。

```
aws vpc-lattice create-service-network-service-association \  
  --service-identifier svc-0285b53b2eEXAMPLE \  
  --service-network-identifier sn-080ec7dc93EXAMPLE
```

输出：

```
{  
  "arn": "arn:aws:vpc-lattice:us-  
east-2:123456789012:servicenetworkserviceassociation/snsa-0e16955a8cEXAMPLE",  
  "createdBy": "123456789012",  
  "dnsEntry": {  
    "domainName": "my-lattice-service-0285b53b2eEXAMPLE.7d67968.vpc-lattice-  
svcs.us-east-2.on.aws",  
    "hostedZoneId": "Z09127221KTH2CEXAMPLE"  
  },  
  "id": "sna-0e16955a8cEXAMPLE",  
  "status": "CREATE_IN_PROGRESS"  
}
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的[管理服务关联](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateServiceNetworkServiceAssociation](#)中的。

## `create-service-network-vpc-association`

以下代码示例演示了如何使用 `create-service-network-vpc-association`。

## AWS CLI

### 创建 VPC 关联

以下`create-service-network-vpc-association`示例将指定的 vpc 与指定的服务网络相关联。指定的安全组控制 VPC 中的哪些资源可以访问服务网络及其服务。

```
aws vpc-lattice create-service-network-vpc-association \  
  --vpc-identifier vpc-0a1b2c3d4eEXAMPLE \  
  --service-network-identifier sn-080ec7dc93EXAMPLE \  
  --security-group-ids sg-0aee16bc6cEXAMPLE
```

输出：

```
{  
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:servicenetworkvpcassociation/  
snva-0821fc8631EXAMPLE",  
  "createdBy": "123456789012",  
  "id": "snva-0821fc8631EXAMPLE",  
  "securityGroupIds": [  
    "sg-0aee16bc6cEXAMPLE"  
  ],  
  "status": "CREATE_IN_PROGRESS"  
}
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的管理 VPC [关联](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateServiceNetworkVpcAssociation](#)中的。

## create-service-network

以下代码示例演示了如何使用 create-service-network。

### AWS CLI

#### 创建服务网络

以下create-service-network示例创建了一个具有指定名称的服务网络。

```
aws vpc-lattice create-service-network \  
  --name my-service-network
```

输出：

```
{  
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:servicenetwork/  
sn-080ec7dc93EXAMPLE",  
  "authType": "NONE",  
  "id": "sn-080ec7dc93EXAMPLE",
```

```
"name": "my-service-network"
}
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的[服务网络](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateServiceNetwork](#)中的。

## create-service

以下代码示例演示了如何使用 create-service。

### AWS CLI

#### 创建服务

以下 create-service 示例创建了一个具有指定名称的服务。

```
aws vpc-lattice create-service \
  --name my-lattice-service
```

输出：

```
{
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:service/
svc-0285b53b2eEXAMPLE",
  "authType": "NONE",
  "dnsEntry": {
    "domainName": "my-lattice-service-0285b53b2eEXAMPLE.1a2b3c4.vpc-lattice-
svcs.us-east-2.on.aws",
    "hostedZoneId": "Z09127221KTH2CEXAMPLE"
  },
  "id": "svc-0285b53b2eEXAMPLE",
  "name": "my-lattice-service",
  "status": "CREATE_IN_PROGRESS"
}
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的 VPC Lattice [中的服务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[CreateService](#)中的。

## create-target-group

以下代码示例演示了如何使用 create-target-group。

## AWS CLI

示例 1：创建类型为“实例”的目标组

以下create-target-group示例使用指定的名称、类型和配置创建目标组。

```
aws vpc-lattice create-target-group \  
  --name my-lattice-target-group-instance \  
  --type INSTANCE \  
  --config file://tg-config.json
```

tg-config.json 的内容：

```
{  
  "port": 443,  
  "protocol": "HTTPS",  
  "protocolVersion": "HTTP1",  
  "vpcIdentifier": "vpc-f1663d9868EXAMPLE"  
}
```

输出：

```
{  
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:targetgroup/  
tg-0eaa4b9ab4EXAMPLE",  
  "config": {  
    "healthCheck": {  
      "enabled": true,  
      "healthCheckIntervalSeconds": 30,  
      "healthCheckTimeoutSeconds": 5,  
      "healthyThresholdCount": 5,  
      "matcher": {  
        "httpCode": "200"  
      },  
      "path": "/",  
      "protocol": "HTTPS",  
      "protocolVersion": "HTTP1",  
      "unhealthyThresholdCount": 2  
    },  
    "port": 443,  
    "protocol": "HTTPS",  
    "protocolVersion": "HTTP1",
```

```
    "vpcIdentifier": "vpc-f1663d9868EXAMPLE"
  },
  "id": "tg-0eaa4b9ab4EXAMPLE",
  "name": "my-lattice-target-group-instance",
  "status": "CREATE_IN_PROGRESS",
  "type": "INSTANCE"
}
```

## 示例 2：创建 IP 类型的目标组

以下 `create-target-group` 示例使用指定的名称、类型和配置创建目标组。

```
aws vpc-lattice create-target-group \
  --name my-lattice-target-group-ip \
  --type IP \
  --config file://tg-config.json
```

`tg-config.json` 的内容：

```
{
  "ipAddressType": "IPV4",
  "port": 443,
  "protocol": "HTTPS",
  "protocolVersion": "HTTP1",
  "vpcIdentifier": "vpc-f1663d9868EXAMPLE"
}
```

输出：

```
{
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:targetgroup/
tg-0eaa4b9ab4EXAMPLE",
  "config": {
    "healthCheck": {
      "enabled": true,
      "healthCheckIntervalSeconds": 30,
      "healthCheckTimeoutSeconds": 5,
      "healthyThresholdCount": 5,
      "matcher": {
        "httpCode": "200"
      }
    },
    "path": "/",

```

```
        "protocol": "HTTPS",
        "protocolVersion": "HTTP1",
        "unhealthyThresholdCount": 2
    },
    "ipAddressType": "IPV4",
    "port": 443,
    "protocol": "HTTPS",
    "protocolVersion": "HTTP1",
    "vpcIdentifier": "vpc-f1663d9868EXAMPLE"
},
"id": "tg-0eaa4b9ab4EXAMPLE",
"name": "my-lattice-target-group-ip",
"status": "CREATE_IN_PROGRESS",
"type": "IP"
}
```

### 示例 3：创建 LAMBDA 类型的目标群组

以下create-target-group示例使用指定的名称、类型和配置创建目标组。

```
aws vpc-lattice create-target-group \
  --name my-lattice-target-group-lambda \
  --type LAMBDA
```

输出：

```
{
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:targetgroup/
tg-0eaa4b9ab4EXAMPLE",
  "id": "tg-0eaa4b9ab4EXAMPLE",
  "name": "my-lattice-target-group-lambda",
  "status": "CREATE_IN_PROGRESS",
  "type": "LAMBDA"
}
```

### 示例 4：创建 ALB 类型的目标组

以下create-target-group示例使用指定的名称、类型和配置创建目标组。

```
aws vpc-lattice create-target-group \
  --name my-lattice-target-group-alb \
  --type ALB \
```



```
--config file://tg-config.json
```

tg-config.json 的内容：

```
{
  "port": 443,
  "protocol": "HTTPS",
  "protocolVersion": "HTTP1",
  "vpcIdentifier": "vpc-f1663d9868EXAMPLE"
}
```

输出：

```
{
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:targetgroup/tg-0eaa4b9ab4EXAMPLE",
  "config": {
    "port": 443,
    "protocol": "HTTPS",
    "protocolVersion": "HTTP1",
    "vpcIdentifier": "vpc-f1663d9868EXAMPLE"
  },
  "id": "tg-0eaa4b9ab4EXAMPLE",
  "name": "my-lattice-target-group-alb",
  "status": "CREATE_IN_PROGRESS",
  "type": "ALB"
}
```

有关更多信息，请参阅 Amazon VPC 莱迪思用户指南中的[目标群体](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTargetGroup](#)中的。

## delete-auth-policy

以下代码示例演示了如何使用 delete-auth-policy。

### AWS CLI

#### 删除身份验证策略

以下delete-auth-policy示例删除了指定服务的身份验证策略。

```
aws vpc-lattice delete-auth-policy \  
  --resource-identifier svc-0285b53b2eEXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅 [Amazon VPC Lattice 用户指南中的身份验证政策](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAuthPolicy](#)中的。

## delete-listener

以下代码示例演示了如何使用 delete-listener。

### AWS CLI

#### 删除监听器

以下delete-listener示例删除了指定的监听器。

```
aws vpc-lattice delete-listener \  
  --listener-identifier listener-07cc7fb0abEXAMPLE \  
  --service-identifier svc-0285b53b2eEXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的[监听器](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteListener](#)中的。

## delete-service-network-service-association

以下代码示例演示了如何使用 delete-service-network-service-association。

### AWS CLI

#### 删除服务关联

以下delete-service-network-service-association示例取消与指定服务关联的关联。

```
aws vpc-lattice delete-service-network-service-association \  
  --service-network-identifier svc-0285b53b2eEXAMPLE \  
  --service-identifier svc-0285b53b2eEXAMPLE
```

```
--service-network-service-association-identifier sns-a-031fabb4d8EXAMPLE
```

输出：

```
{
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:servicenetworkserviceassociation/sns-a-031fabb4d8EXAMPLE",
  "id": "sns-a-031fabb4d8EXAMPLE",
  "status": "DELETE_IN_PROGRESS"
}
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的[管理服务关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteServiceNetworkServiceAssociation](#)中的。

## delete-service-network-vpc-association

以下代码示例演示了如何使用 delete-service-network-vpc-association。

AWS CLI

删除 VPC 关联

以下 delete-service-network-vpc-association 示例取消与指定的 VPC 关联的关联。

```
aws vpc-lattice delete-service-network-vpc-association \
  --service-network-vpc-association-identifier snva-0821fc8631EXAMPLE
```

输出：

```
{
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:servicenetworkvpcassociation/snva-0821fc8631EXAMPLE",
  "id": "snva-0821fc8631EXAMPLE",
  "status": "DELETE_IN_PROGRESS"
}
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的管理 VPC [关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteServiceNetworkVpcAssociation](#)中的。

## delete-service-network

以下代码示例演示了如何使用 delete-service-network。

### AWS CLI

#### 删除服务网络

以下delete-service-network示例删除了指定的服务网络。

```
aws vpc-lattice delete-service-network \  
  --service-network-identifier sn-080ec7dc93EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的[服务网络](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteServiceNetwork](#)中的。

## delete-service

以下代码示例演示了如何使用 delete-service。

### AWS CLI

#### 删除服务

以下delete-service示例删除了指定的服务。

```
aws vpc-lattice delete-service \  
  --service-identifier svc-0285b53b2eEXAMPLE
```

输出：

```
{  
  "arn": "arn:aws:vpc-lattice:us-west-2:123456789012:service/  
svc-0285b53b2eEXAMPLE",  
  "id": "svc-0285b53b2eEXAMPLE",  
  "name": "my-lattice-service",  
  "status": "DELETE_IN_PROGRESS"  
}
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的 [VPC Lattice 中的服务](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteService](#) 中的。

## delete-target-group

以下代码示例演示了如何使用 delete-target-group。

### AWS CLI

#### 删除目标组

以下 delete-target-group 示例将删除指定的目标组。

```
aws vpc-lattice delete-target-group \  
  --target-group-identifier tg-0eaa4b9ab4EXAMPLE
```

#### 输出：

```
{  
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:targetgroup/  
tg-0eaa4b9ab4EXAMPLE",  
  "id": "tg-0eaa4b9ab4EXAMPLE",  
  "status": "DELETE_IN_PROGRESS"  
}
```

有关更多信息，请参阅 Amazon VPC 莱迪思用户指南中的 [目标群体](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteTargetGroup](#) 中的。

## deregister-targets

以下代码示例演示了如何使用 deregister-targets。

### AWS CLI

#### 取消注册目标

以下 deregister-targets 示例从指定目标组中取消注册指定目标。

```
aws vpc-lattice deregister-targets \  
  --target-group-identifier tg-0eaa4b9ab4EXAMPLE
```

```
--targets i-07dd579bc5EXAMPLE \  
--target-group-identifier tg-0eaa4b9ab4EXAMPLE
```

输出：

```
{  
  "successful": [  
    {  
      "id": "i-07dd579bc5EXAMPLE",  
      "port": 443  
    }  
  ],  
  "unsuccessful": []  
}
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的[注册目标](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeregisterTargets](#) 中的。

## get-auth-policy

以下代码示例演示了如何使用 get-auth-policy。

### AWS CLI

获取有关身份验证政策的信息

以下 get-auth-policy 示例获取有关指定服务的身份验证策略的信息。

```
aws vpc-lattice get-auth-policy \  
--resource-identifier svc-0285b53b2eEXAMPLE
```

输出：

```
{  
  "createdAt": "2023-06-07T03:51:20.266Z",  
  "lastUpdatedAt": "2023-06-07T04:39:27.082Z",  
  "policy": "{\n    \"Version\": \"2012-10-17\",  
    \"Statement\": [\n      {\n        \"Effect\": \"Allow\",  
        \"Principal\": {\n          \"AWS\": \"arn:aws:iam::123456789012:role/my-clients\"\n        },  
        \"Action\": \"vpc-lattice-svcs:Invoke\",  
        \"Resource\": \"arn:aws:vpc-lattice:us-east-2:123456789012:service/svc-0285b53b2eEXAMPLE\"\n      }\n    ]\n  }",  
  "state": "Active"
```

```
}
```

有关更多信息，请参阅 [Amazon VPC Lattice 用户指南中的身份验证政策](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetAuthPolicy](#) 中的。

## get-listener

以下代码示例演示了如何使用 `get-listener`。

### AWS CLI

获取有关服务侦听器的信息

以下 `get-listener` 示例获取有关指定服务的指定侦听器的信息。

```
aws vpc-lattice get-listener \  
  --listener-identifier listener-0ccf55918cEXAMPLE \  
  --service-identifier svc-0285b53b2eEXAMPLE
```

输出：

```
{  
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:service/  
svc-0285b53b2eEXAMPLE/listener/listener-0ccf55918cEXAMPLE",  
  "createdAt": "2023-05-07T05:08:45.192Z",  
  "defaultAction": {  
    "forward": {  
      "targetGroups": [  
        {  
          "targetGroupIdentifier": "tg-0ff213abb6EXAMPLE",  
          "weight": 1  
        }  
      ]  
    }  
  },  
  "id": "listener-0ccf55918cEXAMPLE",  
  "lastUpdatedAt": "2023-05-07T05:08:45.192Z",  
  "name": "http-80",  
  "port": 80,  
  "protocol": "HTTP",  
  "serviceArn": "arn:aws:vpc-lattice:us-east-2:123456789012:service/  
svc-0285b53b2eEXAMPLE",
```

```
"serviceId": "svc-0285b53b2eEXAMPLE"
}
```

有关更多信息，请参阅 Amazon VPC 莱迪思用户指南中的[定义路由](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetListener](#)中的。

## get-service-network-service-association

以下代码示例演示了如何使用 `get-service-network-service-association`。

### AWS CLI

获取有关服务关联的信息

以下 `get-service-network-service-association` 示例获取有关指定服务关联的信息。

```
aws vpc-lattice get-service-network-service-association \
  --service-network-service-association-identifier sns-a-031fabb4d8EXAMPLE
```

输出：

```
{
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:servicenetworkserviceassociation/sns-a-031fabb4d8EXAMPLE",
  "createdAt": "2023-05-05T21:48:16.076Z",
  "createdBy": "123456789012",
  "dnsEntry": {
    "domainName": "my-lattice-service-0285b53b2eEXAMPLE.7d67968.vpc-lattice-svcs.us-east-2.on.aws",
    "hostedZoneId": "Z09127221KTH2CEXAMPLE"
  },
  "id": "sns-a-031fabb4d8EXAMPLE",
  "serviceArn": "arn:aws:vpc-lattice:us-east-2:123456789012:service/svc-0285b53b2eEXAMPLE",
  "serviceId": "svc-0285b53b2eEXAMPLE",
  "serviceName": "my-lattice-service",
  "serviceNetworkArn": "arn:aws:vpc-lattice:us-east-2:123456789012:servicenetwork/sn-080ec7dc93EXAMPLE",
  "serviceNetworkId": "sn-080ec7dc93EXAMPLE",
  "serviceNetworkName": "my-service-network",
  "status": "ACTIVE"
}
```



```
}
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的[管理服务关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetServiceNetworkServiceAssociation](#)中的。

## get-service-network-vpc-association

以下代码示例演示了如何使用 `get-service-network-vpc-association`。

### AWS CLI

获取有关 VPC 关联的信息

以下 `get-service-network-vpc-association` 示例获取有关指定 VPC 关联的信息。

```
aws vpc-lattice get-service-network-vpc-association \
  --service-network-vpc-association-identifier snva-0821fc8631EXAMPLE
```

输出：

```
{
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:servicenetworkvpcassociation/
snva-0821fc8631EXAMPLE",
  "createdAt": "2023-06-06T23:41:08.421Z",
  "createdBy": "123456789012",
  "id": "snva-0c5dcb60d6EXAMPLE",
  "lastUpdatedAt": "2023-06-06T23:41:08.421Z",
  "securityGroupIds": [
    "sg-0aee16bc6cEXAMPLE"
  ],
  "serviceNetworkArn": "arn:aws:vpc-lattice:us-east-2:123456789012:servicenetwork/
sn-080ec7dc93EXAMPLE",
  "serviceNetworkId": "sn-080ec7dc93EXAMPLE",
  "serviceNetworkName": "my-service-network",
  "status": "ACTIVE",
  "vpcId": "vpc-0a1b2c3d4eEXAMPLE"
}
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的管理 VPC [关联](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetServiceNetworkVpcAssociation](#)中的。

## get-service-network

以下代码示例演示了如何使用 `get-service-network`。

### AWS CLI

获取有关服务网络的信息

以下`get-service-network`示例获取有关指定服务网络的信息。

```
aws vpc-lattice get-service-network \  
  --service-network-identifier sn-080ec7dc93EXAMPLE
```

输出：

```
{  
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:servicenetwork/  
sn-080ec7dc93EXAMPLE",  
  "authType": "AWS_IAM",  
  "createdAt": "2023-05-05T15:26:08.417Z",  
  "id": "sn-080ec7dc93EXAMPLE",  
  "lastUpdatedAt": "2023-05-05T15:26:08.417Z",  
  "name": "my-service-network",  
  "numberOfAssociatedServices": 2,  
  "numberOfAssociatedVPCs": 3  
}
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的[服务网络](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetServiceNetwork](#)中的。

## get-service

以下代码示例演示了如何使用 `get-service`。

### AWS CLI

获取有关服务的信息

以下`get-service`示例获取有关指定服务的信息。

```
aws vpc-lattice get-service \  
  --service-id sn-080ec7dc93EXAMPLE
```

```
--service-identifier svc-0285b53b2eEXAMPLE
```

输出：

```
{
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:service/
svc-0285b53b2eEXAMPLE",
  "authType": "AWS_IAM",
  "createdAt": "2023-05-05T21:35:29.339Z",
  "dnsEntry": {
    "domainName": "my-lattice-service-0285b53b2eEXAMPLE.7d67968.vpc-lattice-
svcs.us-east-2.on.aws",
    "hostedZoneId": "Z09127221KTH2CFU0HIZH"
  },
  "id": "svc-0285b53b2eEXAMPLE",
  "lastUpdatedAt": "2023-05-05T21:35:29.339Z",
  "name": "my-lattice-service",
  "status": "ACTIVE"
}
```

有关更多信息，请参阅 Amazon VPC 莱迪思用户指南中的[服务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetService](#)中的。

## get-target-group

以下代码示例演示了如何使用 get-target-group。

### AWS CLI

获取有关目标群体的信息

以下get-target-group示例获取有关指定目标组的信息，该目标组的目标类型为INSTANCE。

```
aws vpc-lattice get-target-group \
  --target-group-identifier tg-0eaa4b9ab4EXAMPLE
```

输出：

```
{
  "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:targetgroup/
tg-0eaa4b9ab4EXAMPLE",
```

```
"config": {
  "healthCheck": {
    "enabled": true,
    "healthCheckIntervalSeconds": 30,
    "healthCheckTimeoutSeconds": 5,
    "healthyThresholdCount": 5,
    "matcher": {
      "httpCode": "200"
    },
    "path": "/",
    "protocol": "HTTPS",
    "protocolVersion": "HTTP1",
    "unhealthyThresholdCount": 2
  },
  "port": 443,
  "protocol": "HTTPS",
  "protocolVersion": "HTTP1",
  "vpcIdentifier": "vpc-f1663d9868EXAMPLE"
},
"createdAt": "2023-05-06T04:41:04.122Z",
"id": "tg-0eaa4b9ab4EXAMPLE",
"lastUpdatedAt": "2023-05-06T04:41:04.122Z",
"name": "my-target-group",
"serviceArns": [
  "arn:aws:vpc-lattice:us-east-2:123456789012:service/svc-0285b53b2eEXAMPLE"
],
"status": "ACTIVE",
"type": "INSTANCE"
}
```

有关更多信息，请参阅 Amazon VPC 莱迪思用户指南中的[目标群体](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetTargetGroup](#)中的。

## list-listeners

以下代码示例演示了如何使用 `list-listeners`。

### AWS CLI

#### 列出服务侦听器

以下 `list-listeners` 示例列出了指定服务的侦听器。

```
aws vpc-lattice list-listeners \  
  --service-identifier svc-0285b53b2eEXAMPLE
```

输出：

```
{  
  "items": [  
    {  
      "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:service/  
svc-0285b53b2eEXAMPLE/listener/listener-0ccf55918cEXAMPLE",  
      "createdAt": "2023-05-07T05:08:45.192Z",  
      "id": "listener-0ccf55918cEXAMPLE",  
      "lastUpdatedAt": "2023-05-07T05:08:45.192Z",  
      "name": "http-80",  
      "port": 80,  
      "protocol": "HTTP"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon VPC 莱迪思用户指南中的[定义路由](#)。

• 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListListeners](#)中的。

## list-service-network-service-associations

以下代码示例演示了如何使用 list-service-network-service-associations。

### AWS CLI

#### 列出服务关联

以下list-service-network-service-associations示例列出了指定服务网络的服务关联。该--query选项将输出范围限定为服务关联的 ID。

```
aws vpc-lattice list-service-network-service-associations \  
  --service-network-identifier sn-080ec7dc93EXAMPLE \  
  --query items[*].id
```

输出：

```
[
```

```
"snsa-031fabb4d8EXAMPLE",  
"snsa-0e16955a8cEXAMPLE"  
]
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的[管理服务关联](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListServiceNetworkServiceAssociations](#)中的。

## list-service-network-vpc-associations

以下代码示例演示了如何使用 list-service-network-vpc-associations。

### AWS CLI

列出 VPC 关联

以下list-service-network-vpc-associations示例列出了指定服务网络的 VPC 关联。该--query选项将输出范围限定为 VPC 关联的 ID。

```
aws vpc-lattice list-service-network-vpc-associations \  
  --service-network-identifier sn-080ec7dc93EXAMPLE \  
  --query items[*].id
```

输出：

```
[  
  "snva-0821fc8631EXAMPLE",  
  "snva-0c5dcb60d6EXAMPLE"  
]
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的管理 VPC [关联](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListServiceNetworkVpcAssociations](#)中的。

## list-service-networks

以下代码示例演示了如何使用 list-service-networks。

### AWS CLI

列出您的服务网络

以下`list-service-networks`示例列出了主叫账户拥有或与之共享的服务网络。该`--query`选项将结果范围限定为服务网络的 Amazon 资源名称 (ARN)。

```
aws vpc-lattice list-service-networks \  
  --query items[*].arn
```

输出：

```
[  
  "arn:aws:vpc-lattice:us-east-2:123456789012:servicenetwork/  
sn-080ec7dc93EXAMPLE",  
  "arn:aws:vpc-lattice:us-east-2:111122223333:servicenetwork/sn-0ec4d436cfEXAMPLE"  
]
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的[服务网络](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListServiceNetworks](#)中的。

## list-services

以下代码示例演示了如何使用 `list-services`。

### AWS CLI

列出您的服务

以下`list-services`示例列出了调用账户拥有或与之共享的服务。该`--query`选项将结果的范围限定为服务的亚马逊资源名称 (ARN)。

```
aws vpc-lattice list-services \  
  --query items[*].arn
```

输出：

```
[  
  "arn:aws:vpc-lattice:us-east-2:123456789012:service/svc-0285b53b2eEXAMPLE",  
  "arn:aws:vpc-lattice:us-east-2:111122223333:service/svc-0b8ac96550EXAMPLE"  
]
```

有关更多信息，请参阅 Amazon VPC 莱迪思用户指南中的[服务](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListServices](#)中的。

## list-target-groups

以下代码示例演示了如何使用 `list-target-groups`。

### AWS CLI

列出您的目标群体

以下 `list-target-groups` 示例列出了目标类型为 LAMBDA 的目标组。

```
aws vpc-lattice list-target-groups \
  --target-group-type LAMBDA
```

输出：

```
{
  "items": [
    {
      "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:targetgroup/
tg-045c1b7d9dEXAMPLE",
      "createdAt": "2023-05-06T05:22:16.637Z",
      "id": "tg-045c1b7d9dEXAMPLE",
      "lastUpdatedAt": "2023-05-06T05:22:16.637Z",
      "name": "my-target-group-lam",
      "serviceArns": [
        "arn:aws:vpc-lattice:us-east-2:123456789012:service/
svc-0285b53b2eEXAMPLE"
      ],
      "status": "ACTIVE",
      "type": "LAMBDA"
    }
  ]
}
```

有关更多信息，请参阅 Amazon VPC 莱迪思用户指南中的 [目标群体](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTargetGroups](#) 中的。

## list-targets

以下代码示例演示了如何使用 `list-targets`。



## AWS CLI

### 列出目标组的目标

以下list-targets示例列出了指定目标组的目标。

```
aws vpc-lattice list-targets \  
  --target-group-identifier tg-0eaa4b9ab4EXAMPLE
```

输出：

```
{  
  "items": [  
    {  
      "id": "i-07dd579bc5EXAMPLE",  
      "port": 443,  
      "status": "HEALTHY"  
    },  
    {  
      "id": "i-047b3c9078EXAMPLE",  
      "port": 443,  
      "reasonCode": "HealthCheckFailed",  
      "status": "UNHEALTHY"  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon VPC 莱迪思用户指南中的[目标群体](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTargets](#)中的。

## put-auth-policy

以下代码示例演示了如何使用 put-auth-policy。

## AWS CLI

### 为服务创建身份验证策略

以下put-auth-policy示例授予访问来自使用指定 IAM 角色的任何经过身份验证的委托人的请求的权限。资源是策略所关联的服务的 ARN。

```
aws vpc-lattice put-auth-policy \  
  \
```

```
--resource-identifier svc-0285b53b2eEXAMPLE \  
--policy file://auth-policy.json
```

auth-policy.json 的内容 :

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::123456789012:role/my-clients"  
      },  
      "Action": "vpc-lattice-svcs:Invoke",  
      "Resource": "arn:aws:vpc-lattice:us-east-2:123456789012:service/  
svc-0285b53b2eEXAMPLE"  
    }  
  ]  
}
```

输出 :

```
{  
  "policy": "{\n\"Version\": \"2012-10-17\", \"Statement\": [{\n\"Effect\": \"Allow\n\", \"Principal\": {\n\"AWS\": \"arn:aws:iam::123456789012:role/my-clients\n\"},\n\"Action\": \"vpc-lattice-svcs:Invoke\n\", \"Resource\": \"arn:aws:vpc-lattice:us-\neast-2:123456789012:service/svc-0285b53b2eEXAMPLE\n\"}]}\",  
  "state": "Active"  
}
```

有关更多信息，请参阅 [Amazon VPC Lattice 用户指南中的身份验证政策](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [PutAuthPolicy](#) 中的。

## register-targets

以下代码示例演示了如何使用 register-targets。

### AWS CLI

#### 注册目标

以下 register-targets 示例将指定的目标注册到指定的目标组。

```
aws vpc-lattice register-targets \  
  --targets id=i-047b3c9078EXAMPLE id=i-07dd579bc5EXAMPLE \  
  --target-group-identifier tg-0eaa4b9ab4EXAMPLE
```

输出：

```
{  
  "successful": [  
    {  
      "id": "i-07dd579bc5EXAMPLE",  
      "port": 443  
    }  
  ],  
  "unsuccessful": [  
    {  
      "failureCode": "UnsupportedTarget",  
      "failureMessage": "Instance targets must be in the same VPC as their  
target group",  
      "id": "i-047b3c9078EXAMPLE",  
      "port": 443  
    }  
  ]  
}
```

有关更多信息，请参阅 Amazon VPC Lattice 用户指南中的[注册目标](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterTargets](#)中的。

## AWS WAF Classic 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS WAF Classic。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

## 操作

### put-logging-configuration

以下代码示例演示了如何使用 put-logging-configuration。

#### AWS CLI

要使用指定的 Kinesis Firehose 直播为网页 ACL ARN 创建日志配置 ARN

以下put-logging-configuration示例显示了 WAF 的 CloudFront日志配置。

```
aws waf put-logging-configuration \  
  --logging-configuration ResourceArn=arn:aws:waf::123456789012:webacl/3bffd3ed-  
fa2e-445e-869f-a6a7cf153fd3,LogDestinationConfigs=arn:aws:firehose:us-  
east-1:123456789012:deliverystream/aws-waf-logs-firehose-stream,RedactedFields=[]
```

输出：

```
{  
  "LoggingConfiguration": {  
    "ResourceArn": "arn:aws:waf::123456789012:webacl/3bffd3ed-fa2e-445e-869f-  
a6a7cf153fd3",  
    "LogDestinationConfigs": [  
      "arn:aws:firehose:us-east-1:123456789012:deliverystream/aws-waf-logs-  
firehose-stream"  
    ]  
  }  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutLoggingConfiguration](#)中的。

### update-byte-match-set

以下代码示例演示了如何使用 update-byte-match-set。

#### AWS CLI

更新字节匹配集

以下 `update-byte-match-set` 命令删除 `a` 中的 `ByteMatchTuple` 对象 (过滤器) `ByteMatchSet` :

```
aws waf update-byte-match-set --byte-match-set-id a123fae4-
b567-8e90-1234-5ab67ac8ca90 --change-token 12cs345-67cd-890b-1cd2-c3a4567d89f1 --
updates
Action="DELETE",ByteMatchTuple={FieldToMatch={Type="HEADER",Data="referer"},TargetString="b
```

有关更多信息, 请参阅 [AWS WAF 开发者指南](#) 中的使用字符串匹配条件。

- 有关 API 的详细信息, 请参阅 [AWS CLI 命令参考 UpdateByteMatchSet](#) 中的。

## update-ip-set

以下代码示例演示了如何使用 `update-ip-set`。

### AWS CLI

#### 更新 IP 集

以下 `update-ip-set` 命令使用 IPv4 地址更新 IPset 并删除 IPv6 地址 :

```
aws waf update-ip-set --ip-set-id a123fae4-b567-8e90-1234-5ab67ac8ca90
--change-token 12cs345-67cd-890b-1cd2-c3a4567d89f1 --updates
Action="INSERT",IPSetDescriptor={Type="IPv4",Value="12.34.56.78/16"},Action="DELETE",IPSetD
```

或者, 您可以使用 JSON 文件来指定输入。例如 :

```
aws waf update-ip-set --ip-set-id a123fae4-b567-8e90-1234-5ab67ac8ca90 --change-
token 12cs345-67cd-890b-1cd2-c3a4567d89f1 --updates file://change.json
```

其中 JSON 文件的内容是 :

```
[
{
  "Action": "INSERT",
  "IPSetDescriptor":
  {
    "Type": "IPv4",
    "Value": "12.34.56.78/16"
  }
}
```

```
},
{
  "Action": "DELETE",
  "IPSetDescriptor":
  {
    "Type": "IPV6",
    "Value": "1111:0000:0000:0000:0000:0000:0000:0111/128"
  }
}
]
```

有关更多信息，请参阅 AWS WAF 开发者指南中的使用 IP 匹配条件。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateIpSet](#) 中的。

## update-rule

以下代码示例演示了如何使用 update-rule。

### AWS CLI

#### 更新规则

以下 update-rule 命令删除规则中的谓词对象：

```
aws waf update-rule --rule-id a123fae4-b567-8e90-1234-5ab67ac8ca90
--change-token 12cs345-67cd-890b-1cd2-c3a4567d89f1 --updates
Action="DELETE",Predicate={Negated=false,Type="ByteMatch",DataId="MyByteMatchSetID"}
```

有关更多信息，请参阅 AWS WAF 开发者指南中的使用规则。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateRule](#) 中的。

## update-size-constraint-set

以下代码示例演示了如何使用 update-size-constraint-set。

### AWS CLI

#### 更新大小限制集

以下 update-size-constraint-set 命令删除大小约束集中的 SizeConstraint 对象（滤镜）：

```
aws waf update-size-constraint-set --size-constraint-set-id a123fae4-
b567-8e90-1234-5ab67ac8ca90 --change-token 12cs345-67cd-890b-1cd2-c3a4567d89f1 --
updates
Action="DELETE",SizeConstraint={FieldToMatch={Type="QUERY_STRING"},TextTransformation="NONE"
```

有关更多信息，请参阅 AWS WAF 开发者指南中的使用大小限制条件。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateSizeConstraintSet](#) 中的。

## update-sql-injection-match-set

以下代码示例演示了如何使用 `update-sql-injection-match-set`。

### AWS CLI

#### 更新 SQL 注入匹配集

以下 `update-sql-injection-match-set` 命令删除 SQL 注入匹配集中的 `SqlInjectionMatchTuple` 对象（过滤器）：

```
aws waf update-sql-injection-match-set --sql-injection-
match-set-id a123fae4-b567-8e90-1234-5ab67ac8ca90 --
change-token 12cs345-67cd-890b-1cd2-c3a4567d89f1 --updates
Action="DELETE",SqlInjectionMatchTuple={FieldToMatch={Type="QUERY_STRING"},TextTransformation="NONE"
```

有关更多信息，请参阅 AWS WAF 开发者指南中的使用 SQL 注入匹配条件。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateSqlInjectionMatchSet](#) 中的。

## update-web-acl

以下代码示例演示了如何使用 `update-web-acl`。

### AWS CLI

#### 更新 Web ACL

以下 `update-web-acl` 命令删除 Web ActivatedRule ACL 中的对象。

```
aws waf — a123fae4-b567-8e90 update-web-acl -123 web-acl-id 4-5ab67ac8ca90 — change-
token 12cs345-67cd-890b-1cd2-c3a4567d89f1 — updates action= “Delete” , = ' {Priority=1 , =
“wafrule-1-Example” , Action= {type= “Allow”} , type= “Regular”} 'ActivatedRuleRuleId
```

输出：

```
{
  "ChangeToken": "12cs345-67cd-890b-1cd2-c3a4567d89f1"
}
```

有关更多信息，请参阅 [WA AWS F、Fi AWS rewall Manager](#) 和 [AWS Shield 高级开发者指南](#) 中的 [使用 Web ACL](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 UpdateWebAcl](#) 中的。

## update-xss-match-set

以下代码示例演示了如何使用 `update-xss-match-set`。

### AWS CLI

#### 更新 XSS MatchSet

以下 `update-xss-match-set` 命令删除中的 `XssMatchTuple` 对象（滤镜）`XssMatchSet`：

```
aws waf update-xss-match-set --xss-match-set-id a123fae4-b567-8e90-1234-5ab67ac8ca90
--change-token 12cs345-67cd-890b-1cd2-c3a4567d89f1 --updates
Action="DELETE",XssMatchTuple={FieldToMatch={Type="QUERY_STRING"},TextTransformation="URL_D
```

有关更多信息，请参阅 [AWS WAF 开发者指南](#) 中的 [使用跨站点脚本匹配条件](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 UpdateXssMatchSet](#) 中的。

## AWS WAF Classic Regional 使用示例 AWS CLI

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 [AWS WAF Classic Regional](#)。

### AWS Command Line Interface

**操作** 是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

**场景** 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 [GitHub](#)，您可以在其中找到有关如何在上下文中设置和运行代码的说明。



## 主题

- [操作](#)

## 操作

### **associate-web-acl**

以下代码示例演示了如何使用 `associate-web-acl`。

#### AWS CLI

将 Web ACL 与资源关联

以下 `associate-web-acl` 命令将指定的 Web ACL 与由 `resource-arn` 指定的资源相关联。 `web-acl-id` 资源 ARN 可以指应用程序负载均衡器或 API Gateway：

```
aws waf-regional associate-web-acl \  
  --web-acl-id a123fae4-b567-8e90-1234-5ab67ac8ca90 \  
  --resource-arn 12cs345-67cd-890b-1cd2-c3a4567d89f1
```

有关更多信息，请参阅 AWS WAF 开发人员指南中的 [使用 Web ACL](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateWebAcL](#) 中的。

### **put-logging-configuration**

以下代码示例演示了如何使用 `put-logging-configuration`。

#### AWS CLI

要使用指定的 Kinesis Firehose 数据流 ARN 为网页 ACL 创建日志配置，请使用 ARN

以下 `put-logging-configuration` 示例显示了在“区域”中使用 ALB/ApiGateway 的 WAF 的日志配置。 `us-east-1`

```
aws waf-regional put-logging-configuration \  
  --logging-configuration ResourceArn=arn:aws:waf-  
regional:us-east-1:123456789012:webacl/3bffd3ed-fa2e-445e-869f-  
a6a7cf153fd3,LogDestinationConfigs=arn:aws:firehose:us-  
east-1:123456789012:deliverystream/aws-waf-logs-firehose-stream,RedactedFields=[] \  
  --
```

```
--region us-east-1
```

输出：

```
{
  "LoggingConfiguration": {
    "ResourceArn": "arn:aws:waf-regional:us-east-1:123456789012:webacl/3bffd3ed-
fa2e-445e-869f-a6a7cf153fd3",
    "LogDestinationConfigs": [
      "arn:aws:firehose:us-east-1:123456789012:deliverystream/aws-waf-logs-
firehose-stream"
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutLoggingConfiguration](#)中的。

## update-byte-match-set

以下代码示例演示了如何使用 update-byte-match-set。

### AWS CLI

#### 更新字节匹配集

以下update-byte-match-set命令删除中的ByteMatchTuple对象（过滤器）ByteMatchSet。由于该updates值嵌入了双引号，因此必须用单引号将该值括起来。

```
aws waf-regional update-byte-match-set \
  --byte-match-set-id a123fae4-b567-8e90-1234-5ab67ac8ca90 \
  --change-token 12cs345-67cd-890b-1cd2-c3a4567d89f1 \
  --updates
'Action="DELETE",ByteMatchTuple={FieldToMatch={Type="HEADER",Data="referer"},TargetString="'
```

有关更多信息，请参阅 AWS WAF 开发者指南中的[使用字符串匹配条件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateByteMatchSet](#)中的。

## update-ip-set

以下代码示例演示了如何使用 update-ip-set。

## AWS CLI

### 更新 IP 集

以下 `update-ip-set` 命令使用 IPv4 地址更新 IPset 并删除 IPv6 地址。 `change-token` 通过运行 `get-change-token` 命令获取的值。由于更新的值包括嵌入的双引号，因此必须用单引号将该值括起来。

```
aws waf update-ip-set \  
  --ip-set-id a123fae4-b567-8e90-1234-5ab67ac8ca90 \  
  --change-token 12cs345-67cd-890b-1cd2-c3a4567d89f1 \  
  --updates  
'Action="INSERT",IPSetDescriptor={Type="IPV4",Value="12.34.56.78/16"},Action="DELETE",IPSet'
```

或者，您可以使用 JSON 文件来指定输入。例如：

```
aws waf-regional update-ip-set \  
  --ip-set-id a123fae4-b567-8e90-1234-5ab67ac8ca90 \  
  --change-token 12cs345-67cd-890b-1cd2-c3a4567d89f1 \  
  --updates file://change.json
```

的内容 `change.json`

```
[  
  {  
    "Action": "INSERT",  
    "IPSetDescriptor":  
    {  
      "Type": "IPV4",  
      "Value": "12.34.56.78/16"  
    }  
  },  
  {  
    "Action": "DELETE",  
    "IPSetDescriptor":  
    {  
      "Type": "IPV6",  
      "Value": "1111:0000:0000:0000:0000:0000:0000:0111/128"  
    }  
  }  
]
```

有关更多信息，请参阅《AWS WAF 开发者指南》中的[“使用 IP 匹配条件”](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateIpSet](#)中的。

## update-rule

以下代码示例演示了如何使用 update-rule。

### AWS CLI

#### 更新规则

以下update-rule命令删除规则中的Predicate对象。由于该updates值嵌入了双引号，因此必须用单引号将整个值括起来。

```
aws waf-regional update-rule \  
  --rule-id a123fae4-b567-8e90-1234-5ab67ac8ca90 \  
  --change-token 12cs345-67cd-890b-1cd2-c3a4567d89f1 \  
  --updates  
  'Action="DELETE",Predicate={Negated=false,Type="ByteMatch",DataId="MyByteMatchSetID"}'
```

有关更多信息，请参阅《AWS WAF 开发者指南》中的[使用规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateRule](#)中的。

## update-size-constraint-set

以下代码示例演示了如何使用 update-size-constraint-set。

### AWS CLI

#### 更新大小限制集

以下update-size-constraint-set命令删除大小约束集中的 SizeConstraint 对象（过滤器）。由于该updates值包含嵌入的双引号，因此必须用单引号将整个值括起来。

```
aws waf-regional update-size-constraint-set \  
  --size-constraint-set-id a123fae4-b567-8e90-1234-5ab67ac8ca90 \  
  --change-token 12cs345-67cd-890b-1cd2-c3a4567d89f1 \  
  --updates  
  'Action="DELETE",SizeConstraint={FieldToMatch={Type="QUERY_STRING"},TextTransformation="NON"
```

有关更多信息，请参阅 AWS WAF 开发者指南中的[使用大小限制条件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateSizeConstraintSet](#)中的。

## update-sql-injection-match-set

以下代码示例演示了如何使用 update-sql-injection-match-set。

### AWS CLI

#### 更新 SQL 注入匹配集

以下update-sql-injection-match-set命令删除 SQL 注入匹配集中的SqlInjectionMatchTuple对象（过滤器）。由于该updates值包含嵌入式双引号，因此必须用单引号将整个值括起来。：

```
aws waf-regional-se update-sql-injection-match t -- sql-injection-match-set id
a123fae4-b567-8e90-1234-5ab67ac8ca90 -- change-token 12cs345-67cd-1cd2-
c3a4567d89f1 -- 更新 'action= "delete" , = {type= "query_String"} , = "URL_DECODE"}
'SqlInjectionMatchTupleFieldToMatchTextTransformation
```

有关更多信息，请参阅 AWS WAF 开发人员指南中的[使用 SQL 注入匹配条件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateSqlInjectionMatchSet](#)中的。

## update-web-acl

以下代码示例演示了如何使用 update-web-acl。

### AWS CLI

#### 更新 Web ACL

以下update-web-acl命令删除 Web ActivatedRule ACL 中的对象。由于该updates值包含嵌入式双引号，因此必须用单引号将整个值括起来。

```
aws waf-regional update-web-acl \  
  --web-acl-id a123fae4-b567-8e90-1234-5ab67ac8ca90 \  
  --change-token 12cs345-67cd-890b-1cd2-c3a4567d89f1 \  
  --updates Action="DELETE",ActivatedRule='{Priority=1,RuleId="WAFRule-1-  
Example",Action={Type="ALLOW"},Type="ALLOW"}'
```

有关更多信息，请参阅 AWS WAF 开发人员指南中的[使用 Web ACL](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateWebAcl](#)中的。

## update-xss-match-set

以下代码示例演示了如何使用 update-xss-match-set。

### AWS CLI

#### 更新 XSS MatchSet

以下 update-xss-match-set 命令删除中的 XssMatchTuple 对象（滤镜）XssMatchSet。由于该 updates 值包含嵌入的双引号，因此必须用单引号将整个值括起来。

```
aws waf-regional update-xss-match-set \  
  --xss-match-set-id a123fae4-b567-8e90-1234-5ab67ac8ca90 \  
  --change-token 12cs345-67cd-890b-1cd2-c3a4567d89f1 \  
  --updates  
'Action="DELETE",XssMatchTuple={FieldToMatch={Type="QUERY_STRING"},TextTransformation="URL_
```

有关更多信息，请参阅 AWS WAF 开发人员指南中的[使用跨站点脚本匹配条件](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateXssMatchSet](#)中的。

## AWS WAFV2 使用示例 AWS CLI

以下代码示例向您展示了如何使用 with 来执行操作和实现常见场景 AWS WAFV2。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### associate-web-acl

以下代码示例演示了如何使用 `associate-web-acl`。

#### AWS CLI

将 Web ACL 与区域 AWS 资源关联

以下 `associate-web-acl` 示例将指定的 Web ACL 与 Application Load Balancer 关联起来。

```
aws wafv2 associate-web-acl \  
  --web-acl-arn arn:aws:wafv2:us-west-2:123456789012:regional/webacl/test-cli/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --resource-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/  
app/waf-cli-alb/1ea17125f8b25a2a \  
  --region us-west-2
```

此命令不生成任何输出。

有关更多信息，请参阅《WAF》、《Fi AWS rewall Manager AWS》和《[AWS Shield 高级开发者指南](#)》中的“[将 Web ACL 与 AWS 资源关联或取消关联](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [AssociateWebAcl](#) 中的。

### check-capacity

以下代码示例演示了如何使用 `check-capacity`。

#### AWS CLI

获取一组规则使用的容量

以下内容 `check-capacity` 检索包含基于速率的规则语句和包含嵌套规则的 AND 规则语句的规则集的容量要求。

```
aws wafv2 check-capacity \  
  --scope REGIONAL \  
  --rules file://waf-rule-list.json \  
  --region us-west-2
```

文件内容://waf-rule-list.json:

```
[
  {
    "Name":"basic-rule",
    "Priority":0,
    "Statement":{
      "AndStatement":{
        "Statements":[
          {
            "ByteMatchStatement":{
              "SearchString":"example.com",
              "FieldToMatch":{
                "SingleHeader":{
                  "Name":"host"
                }
              },
              "TextTransformations":[
                {
                  "Priority":0,
                  "Type":"LOWERCASE"
                }
              ],
              "PositionalConstraint":"EXACTLY"
            }
          },
          {
            "GeoMatchStatement":{
              "CountryCodes":[
                "US",
                "IN"
              ]
            }
          }
        ]
      }
    },
    "Action":{
      "Allow":{
      }
    },
    "VisibilityConfig":{
      "SampledRequestsEnabled":true,

```



```
        "CloudWatchMetricsEnabled":true,
        "MetricName":"basic-rule"
    },
    {
        "Name":"rate-rule",
        "Priority":1,
        "Statement":{
            "RateBasedStatement":{
                "Limit":1000,
                "AggregateKeyType":"IP"
            }
        },
        "Action":{
            "Block":{
            }
        },
        "VisibilityConfig":{
            "SampledRequestsEnabled":true,
            "CloudWatchMetricsEnabled":true,
            "MetricName":"rate-rule"
        }
    }
]
```

输出：

```
{
  "Capacity":15
}
```

有关更多信息，请参阅 [AWS WAF、AWS Firewall Manager 和 AWS Shield 高级开发者指南中的 WAF Web ACL 容量单位 \(WCU\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CheckCapacity](#) 中的。

## create-ip-set

以下代码示例演示了如何使用 create-ip-set。

## AWS CLI

创建 IP 集以供您的 Web ACL 和规则组使用

以下 `create-ip-set` 命令创建具有单一地址范围规范的 IP 集。

```
aws wafv2 create-ip-set \  
  --name testip \  
  --scope REGIONAL \  
  --ip-address-version IPV4 \  
  --addresses 198.51.100.0/16
```

输出：

```
{  
  "Summary": {  
    "ARN": "arn:aws:wafv2:us-west-2:123456789012:regional/ipset/testip/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "Description": "",  
    "Name": "testip",  
    "LockToken": "447e55ac-0000-0000-0000-86b67c17f8b5",  
    "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"  
  }  
}
```

有关更多信息，请参阅 AWS WAF、Firewal AWS I Manager 和 AWS Shield 高级开发者指南中的 [IP 集](#) 和 [正则表达式模式集](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateIpSet](#) 中的。

## `create-regex-pattern-set`

以下代码示例演示了如何使用 `create-regex-pattern-set`。

## AWS CLI

创建用于您的 Web ACL 和规则组的正则表达式模式集

以下 `create-regex-pattern-set` 命令创建指定了两个正则表达式模式的正则表达式模式集。

```
aws wafv2 create-regex-pattern-set \  
  --name regexPatterSet01 \  
  --scope REGIONAL \  
  --patterns ".*"
```

```
--description 'Test web-acl' \  
--regular-expression-list '[{"RegexString": "/[0-9]*/"}, {"RegexString": "/[a-z]*/"}]'
```

输出：

```
{  
  "Summary": {  
    "ARN": "arn:aws:wafv2:us-west-2:123456789012:regional/regexpatternset/  
regexPatterSet01/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "Description": "Test web-acl",  
    "Name": "regexPatterSet01",  
    "LockToken": "0bc01e21-03c9-4b98-9433-6229cbf1ef1c",  
    "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"  
  }  
}
```

有关更多信息，请参阅 [AWS WAF](#)、[Firewal AWS I Manager](#) 和 [AWS Shield 高级开发者指南](#) 中的 [IP 集](#) 和 [正则表达式模式集](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [CreateRegexPatternSet](#) 中的。

## create-rule-group

以下代码示例演示了如何使用 `create-rule-group`。

### AWS CLI

创建用于您的 Web ACL 的自定义规则组

以下 `create-rule-group` 命令创建供区域使用的自定义规则组。该组的规则语句以 JSON 格式的文件提供。

```
aws wafv2 create-rule-group \  
  --name "TestRuleGroup" \  
  --scope REGIONAL \  
  --capacity 250 \  
  --rules file://waf-rule.json \  
  --visibility-config  
  SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=TestRuleGroupMetrics  
 \  
  --region us-west-2
```

文件内容://waf-rule.json:

```
[
  {
    "Name": "basic-rule",
    "Priority": 0,
    "Statement": {
      "AndStatement": {
        "Statements": [
          {
            "ByteMatchStatement": {
              "SearchString": "example.com",
              "FieldToMatch": {
                "SingleHeader": {
                  "Name": "host"
                }
              },
              "TextTransformations": [
                {
                  "Priority": 0,
                  "Type": "LOWERCASE"
                }
              ],
              "PositionalConstraint": "EXACTLY"
            },
            {
              "GeoMatchStatement": {
                "CountryCodes": [
                  "US",
                  "IN"
                ]
              }
            }
          ]
        }
      },
      "Action": {
        "Allow": {
        }
      },
      "VisibilityConfig": {
        "SampledRequestsEnabled": true,

```

```

        "CloudWatchMetricsEnabled":true,
        "MetricName":"basic-rule"
    }
}
]

```

输出：

```

{
  "Summary":{
    "ARN":"arn:aws:wafv2:us-west-2:123456789012:regional/rulegroup/
TestRuleGroup/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Description":"",
    "Name":"TestRuleGroup",
    "LockToken":"7b3bcec2-374e-4c5a-b2b9-563bf47249f0",
    "Id":"a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  }
}

```

有关更多信息，请参阅《AWS WAF》、《Fi AWS rewall Manager》和《AWS Shield 高级开发者指南》中的“管理[自己的规则组](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRuleGroup](#)中的。

## create-web-acl

以下代码示例演示了如何使用 create-web-acl。

### AWS CLI

#### 创建 Web ACL

以下create-web-acl命令创建供区域使用的 Web ACL。Web ACL 的规则语句在 JSON 格式的文件中提供。

```

aws wafv2 create-web-acl \
  --name TestWebAcl \
  --scope REGIONAL \
  --default-action Allow={} \
  --visibility-config
SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=TestWebAclMetrics
\

```

```
--rules file://waf-rule.json \  
--region us-west-2
```

文件内容://waf-rule.json:

```
[  
  {  
    "Name":"basic-rule",  
    "Priority":0,  
    "Statement":{  
      "AndStatement":{  
        "Statements":[  
          {  
            "ByteMatchStatement":{  
              "SearchString":"example.com",  
              "FieldToMatch":{  
                "SingleHeader":{  
                  "Name":"host"  
                }  
              },  
              "TextTransformations":[  
                {  
                  "Priority":0,  
                  "Type":"LOWERCASE"  
                }  
              ],  
              "PositionalConstraint":"EXACTLY"  
            }  
          },  
          {  
            "GeoMatchStatement":{  
              "CountryCodes":[  
                "US",  
                "IN"  
              ]  
            }  
          }  
        ]  
      }  
    },  
    "Action":{  
      "Allow":{
```

```

    }
  },
  "VisibilityConfig":{
    "SampledRequestsEnabled":true,
    "CloudWatchMetricsEnabled":true,
    "MetricName":"basic-rule"
  }
}
]

```

输出：

```

{
  "Summary":{
    "ARN":"arn:aws:wafv2:us-west-2:123456789012:regional/webacl/TestWebAcl/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Description":"","
    "Name":"TestWebAcl",
    "LockToken":"2294b3a1-eb60-4aa0-a86f-a3ae04329de9",
    "Id":"a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  }
}

```

有关更多信息，请参阅 [W AWS AF、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的管理和使用 Web 访问控制列表 \(Web ACL\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateWebAcl](#) 中的。

## delete-ip-set

以下代码示例演示了如何使用 delete-ip-set。

### AWS CLI

#### 删除 IP 集

以下内容 delete-ip-set 删除指定的 IP 集。此调用需要一个 ID (您可以从调用中获取) 和一个锁定令牌 (可以从调用中获取) list-ip-sets 和 get-ip-set。list-ip-sets

```

aws wafv2 delete-ip-set \
  --name test1 \
  --scope REGIONAL \

```

```
--id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
--lock-token 46851772-db6f-459d-9385-49428812e357
```

此命令不生成任何输出。

有关更多信息，请参阅 [AWS WAF](#)、[Firewal AWS I Manager](#) 和 [AWS Shield 高级开发者指南](#) 中的 [IP 集](#) 和 [正则表达式模式集](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 DeleteIpSet](#) 中的。

## delete-logging-configuration

以下代码示例演示了如何使用 delete-logging-configuration。

### AWS CLI

禁用 Web ACL 的日志记录

以下内容从指定的 Web ACL 中 delete-logging-configuration 删除所有日志配置。

```
aws wafv2 delete-logging-configuration \  
  --resource-arn arn:aws:wafv2:us-west-2:123456789012:regional/webacl/test/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222
```

此命令不生成任何输出。

有关更多信息，请参阅 [W AWS AF](#)、[Fi AWS rewall Manager](#) 和 [AWS Shield 高级开发者指南](#) 中的 [记录 Web ACL 流量信息](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 DeleteLoggingConfiguration](#) 中的。

## delete-regex-pattern-set

以下代码示例演示了如何使用 delete-regex-pattern-set。

### AWS CLI

删除正则表达式模式集

以下内容 delete-regex-pattern-set 更新了指定正则表达式模式集的设置。此呼叫需要一个 ID (您可以从呼叫中获取) 和一个锁定令牌 (您可以从通话 list-regex-pattern-sets 或呼叫中获取) get-regex-pattern-set。list-regex-pattern-sets



```
aws wafv2 delete-regex-pattern-set \  
  --name regexPatterSet01 \  
  --scope REGIONAL \  
  --id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --lock-token 0bc01e21-03c9-4b98-9433-6229cbf1ef1c
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS WAF、Firewal AWS I Manager 和 AWS Shield 高级开发者指南中的 IP 集和[正则表达式模式集](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRegexPatternSet](#)中的。

## delete-rule-group

以下代码示例演示了如何使用 delete-rule-group。

### AWS CLI

#### 删除自定义规则组

以下内容delete-rule-group删除了指定的自定义规则组。此呼叫需要一个 ID ( 您可以从呼叫中获取 ) 和一个锁定令牌 ( 您可以从通话list-rule-groups或呼叫中获取 ) get-rule-group。list-rule-groups

```
aws wafv2 delete-rule-group \  
  --name TestRuleGroup \  
  --scope REGIONAL \  
  --id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --lock-token 7b3bcec2-0000-0000-0000-563bf47249f0
```

此命令不生成任何输出。

有关更多信息，请参阅《AWS WAF》、《Fi AWS rewall Manager》和《 AWS Shield 高级开发者指南》中的“管理[自己的规则组](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRuleGroup](#)中的。

## delete-web-acl

以下代码示例演示了如何使用 delete-web-acl。

## AWS CLI

### 删除 Web ACL

以下`delete-web-acl`操作会从您的账户中删除指定的 Web ACL。只有当 Web ACL 未与任何资源关联时，才能将其删除。此呼叫需要一个 ID（您可以从呼叫中获取）和一个锁定令牌（您可以从通话`list-web-acls`或呼叫中获取）`get-web-acl`。`list-web-acls`

```
aws wafv2 delete-web-acl \  
  --name test \  
  --scope REGIONAL \  
  --id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --lock-token ebab4ed2-155e-4c9a-9efb-e4c45665b1f5
```

此命令不生成任何输出。

有关更多信息，请参阅 [W AWS AF、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的管理和使用 Web 访问控制列表 \(Web ACL\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DeleteWebAcl](#) 中的。

## describe-managed-rule-group

以下代码示例演示了如何使用 `describe-managed-rule-group`。

### AWS CLI

#### 检索托管规则组的描述

以下内容 `describe-managed-rule-group` 检索 AWS 托管规则组的描述。

```
aws wafv2 describe-managed-rule-group \  
  --vendor-name AWS \  
  --name AWSManagedRulesCommonRuleSet \  
  --scope REGIONAL
```

输出：

```
{  
  "Capacity": 700,  
  "Rules": [  
    {  
      "Name": "NoUserAgent_HEADER",
```

```
    "Action": {
      "Block": {}
    }
  },
  {
    "Name": "UserAgent_BadBots_HEADER",
    "Action": {
      "Block": {}
    }
  },
  {
    "Name": "SizeRestrictions_QUERYSTRING",
    "Action": {
      "Block": {}
    }
  },
  {
    "Name": "SizeRestrictions_Cookie_HEADER",
    "Action": {
      "Block": {}
    }
  },
  {
    "Name": "SizeRestrictions_BODY",
    "Action": {
      "Block": {}
    }
  },
  {
    "Name": "SizeRestrictions_URIPATH",
    "Action": {
      "Block": {}
    }
  },
  {
    "Name": "EC2MetaDataSSRF_BODY",
    "Action": {
      "Block": {}
    }
  },
  {
    "Name": "EC2MetaDataSSRF_COOKIE",
    "Action": {
      "Block": {}
    }
  }
}
```

```
    }
  },
  {
    "Name": "EC2MetaDataSSRF_URIPATH",
    "Action": {
      "Block": {}
    }
  },
  {
    "Name": "EC2MetaDataSSRF_QUERYARGUMENTS",
    "Action": {
      "Block": {}
    }
  },
  {
    "Name": "GenericLFI_QUERYARGUMENTS",
    "Action": {
      "Block": {}
    }
  },
  {
    "Name": "GenericLFI_URIPATH",
    "Action": {
      "Block": {}
    }
  },
  {
    "Name": "GenericLFI_BODY",
    "Action": {
      "Block": {}
    }
  },
  {
    "Name": "RestrictedExtensions_URIPATH",
    "Action": {
      "Block": {}
    }
  },
  {
    "Name": "RestrictedExtensions_QUERYARGUMENTS",
    "Action": {
      "Block": {}
    }
  }
}
```

```
    },
    {
      "Name": "GenericRFI_QUERYARGUMENTS",
      "Action": {
        "Block": {}
      }
    },
    {
      "Name": "GenericRFI_BODY",
      "Action": {
        "Block": {}
      }
    },
    {
      "Name": "GenericRFI_URI_PATH",
      "Action": {
        "Block": {}
      }
    },
    {
      "Name": "CrossSiteScripting_COOKIE",
      "Action": {
        "Block": {}
      }
    },
    {
      "Name": "CrossSiteScripting_QUERYARGUMENTS",
      "Action": {
        "Block": {}
      }
    },
    {
      "Name": "CrossSiteScripting_BODY",
      "Action": {
        "Block": {}
      }
    },
    {
      "Name": "CrossSiteScripting_URI_PATH",
      "Action": {
        "Block": {}
      }
    }
  ]
```

```
}
```

有关更多信息，请参阅 AWS WAF、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的[托管规则组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeManagedRuleGroup](#)中的。

## disassociate-web-acl

以下代码示例演示了如何使用 disassociate-web-acl。

### AWS CLI

解除 Web ACL 与区域 AWS 资源的关联

以下disassociate-web-acl示例从指定的 Application Load Balancer 中删除任何现有 Web ACL 关联。

```
aws wafv2 disassociate-web-acl \  
    --resource-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/  
app/waf-cli-alb/1ea17125f8b25a2a \  
    --region us-west-2
```

此命令不生成任何输出。

有关更多信息，请参阅《WAF》、《Fi AWS rewall Manager AWS》和《AWS Sh [ield 高级开发者指南](#)》中的“[将 Web ACL 与 AWS 资源关联或取消关联](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateWebAcl](#)中的。

## get-ip-set

以下代码示例演示了如何使用 get-ip-set。

### AWS CLI

检索特定的 IP 集

以下内容get-ip-set检索具有指定名称、范围和 ID 的 IP 集。您可以从命令create-ip-set和 中获取 IP 集的 ID list-ip-sets。

```
aws wafv2 get-ip-set \  

```

```
--name testip \  
--scope REGIONAL \  
--id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "IPSet":{  
    "Description":"","  
    "Name":"testip",  
    "IPAddressVersion":"IPV4",  
    "Id":"a1b2c3d4-5678-90ab-cdef-EXAMPLE1111",  
    "ARN":"arn:aws:wafv2:us-west-2:123456789012:regional/ipset/testip/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE1111",  
    "Addresses":[  
      "192.0.2.0/16"  
    ]  
  },  
  "LockToken":"447e55ac-2396-4c6d-b9f9-86b67c17f8b5"  
}
```

有关更多信息，请参阅 AWS WAF、Firewal AWS I Manager 和 AWS Shield 高级开发者指南中的 [IP 集](#)和[正则表达式模式集](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetIpSet](#)中的。

## get-logging-configuration

以下代码示例演示了如何使用 get-logging-configuration。

### AWS CLI

检索 Web ACL 的日志配置

以下内容get-logging-configuration检索指定 Web ACL 的日志配置。

```
aws wafv2 get-logging-configuration \  
  --resource-arn arn:aws:wafv2:us-west-2:123456789012:regional/webacl/test/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222 \  
  --region us-west-2
```

输出：

```
{
  "LoggingConfiguration":{
    "ResourceArn":"arn:aws:wafv2:us-west-2:123456789012:regional/webacl/test/
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "RedactedFields":[
      {
        "Method":{
          }
        }
      ],
    "LogDestinationConfigs":[
      "arn:aws:firehose:us-west-2:123456789012:deliverystream/aws-waf-logs-
custom-transformation"
    ]
  }
}
```

有关更多信息，请参阅 [WAF](#)、[AWS IAM](#)、[AWS IAM](#) 和 [AWS Shield 高级开发者指南](#) 中的 [记录 Web ACL 流量信息](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [GetLoggingConfiguration](#) 中的。

## get-rate-based-statement-managed-keys

以下代码示例演示了如何使用 `get-rate-based-statement-managed-keys`。

### AWS CLI

检索被基于速率的规则屏蔽的 IP 地址列表

以下内容 `get-rate-based-statement-managed-keys` 检索当前被区域应用程序使用的基于速率的规则屏蔽的 IP 地址。

```
aws wafv2 get-rate-based-statement-managed-keys \
  --scope REGIONAL \
  --web-acl-name testwebacl2 \
  --web-acl-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
  --rule-name ratebasedtest
```

输出：



```
{
  "ManagedKeysIPV4":{
    "IPAddressVersion":"IPV4",
    "Addresses":[
      "198.51.100.0/32"
    ]
  },
  "ManagedKeysIPV6":{
    "IPAddressVersion":"IPV6",
    "Addresses":[

    ]
  }
}
```

有关更多信息，请参阅 AWS WAF、AWS Firewall Manager 和 AWS Shield 高级开发者指南中的[基于速率的规则声明](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetRateBasedStatementManagedKeys](#)中的。

## get-regex-pattern-set

以下代码示例演示了如何使用 get-regex-pattern-set。

### AWS CLI

#### 检索特定的正则表达式模式集

以下内容 get-regex-pattern-set 检索具有指定名称、范围、区域和 ID 的正则表达式模式集。您可以从命令 create-regex-pattern-set 和中获取正则表达式模式集的 ID。list-regex-pattern-sets

```
aws wafv2 get-regex-pattern-set \
  --name regexPatterSet01 \
  --scope REGIONAL \
  --id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
  --region us-west-2
```

输出：

```
{
  "RegexPatternSet":{
```

```

    "Description":"Test web-acl",
    "RegularExpressionList":[
      {
        "RegexString":"/[0-9]*/"
      },
      {
        "RegexString":"/[a-z]*/"
      }
    ],
    "Name":"regexPatterSet01",
    "ARN":"arn:aws:wafv2:us-west-2:123456789012:regional/regexpatternset/
regexPatterSet01/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Id":"a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  },
  "LockToken":"c8abf33f-b6fc-46ae-846e-42f994d57b29"
}

```

有关更多信息，请参阅 [AWS WAF](#)、[Firewal AWS I Manager](#) 和 [AWS Shield 高级开发者指南](#) 中的 [IP 集](#) 和 [正则表达式模式集](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考](#) [GetRegexPatternSet](#) 中的。

## get-rule-group

以下代码示例演示了如何使用 `get-rule-group`。

### AWS CLI

检索特定的自定义规则组

以下内容 `get-rule-group` 检索具有指定名称、范围和 ID 的自定义规则组。您可以从命令 `create-rule-group` 和 `list-rule-groups` 中获取规则组的 ID。

```

aws wafv2 get-rule-group \
  --name ff \
  --scope REGIONAL \
  --id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111

```

输出：

```

{
  "RuleGroup":{
    "Capacity":1,

```

```
"Description": "",
"Rules": [
  {
    "Priority": 0,
    "Action": {
      "Block": {

      }
    },
    "VisibilityConfig": {
      "SampledRequestsEnabled": true,
      "CloudWatchMetricsEnabled": true,
      "MetricName": "jj"
    },
    "Name": "jj",
    "Statement": {
      "SizeConstraintStatement": {
        "ComparisonOperator": "LE",
        "TextTransformations": [
          {
            "Priority": 0,
            "Type": "NONE"
          }
        ],
        "FieldToMatch": {
          "UriPath": {

          }
        },
        "Size": 7
      }
    }
  },
  {
    "VisibilityConfig": {
      "SampledRequestsEnabled": true,
      "CloudWatchMetricsEnabled": true,
      "MetricName": "ff"
    },
    "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "ARN": "arn:aws:wafv2:us-west-2:123456789012:regional/rulegroup/ff/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Name": "ff"
  },
],
```

```
"LockToken": "485458c9-1830-4234-af31-ec4d52ced1b3"
}
```

有关更多信息，请参阅《AWS WAF》、《Fi AWS rewall Manager》和《AWS Shield 高级开发者指南》中的“管理[自己的规则组](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRuleGroup](#)中的。

## get-sampled-requests

以下代码示例演示了如何使用 get-sampled-requests。

### AWS CLI

检索 Web ACL 的 Web 请求示例

以下内容get-sampled-requests检索指定 Web ACL、规则指标和时间范围的 Web 请求样本。

```
aws wafv2 get-sampled-requests \
  --web-acl-arn arn:aws:wafv2:us-west-2:123456789012:regional/webacl/test-cli/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
  --rule-metric-name AWS-AWSManagedRulesSQLiRuleSet \
  --scope=REGIONAL \
  --time-window StartTime=2020-02-12T20:00Z,EndTime=2020-02-12T21:10Z \
  --max-items 100
```

输出：

```
{
  "TimeWindow": {
    "EndTime": 1581541800.0,
    "StartTime": 1581537600.0
  },
  "SampledRequests": [
    {
      "Action": "BLOCK",
      "Timestamp": 1581541799.564,
      "RuleNameWithinRuleGroup": "AWS#AWSManagedRulesSQLiRuleSet#SQLi_BODY",
      "Request": {
        "Country": "US",
        "URI": "/",
        "Headers": [
          {
```

```
        "Name": "Host",
        "Value": "alb-test-1EXAMPLE1.us-east-1.elb.amazonaws.com"
    },
    {
        "Name": "Content-Length",
        "Value": "7456"
    },
    {
        "Name": "User-Agent",
        "Value": "curl/7.53.1"
    },
    {
        "Name": "Accept",
        "Value": "/"
    },
    {
        "Name": "Content-Type",
        "Value": "application/x-www-form-urlencoded"
    }
    ],
    "ClientIP": "198.51.100.08",
    "Method": "POST",
    "HTTPVersion": "HTTP/1.1"
},
"Weight": 1
},
{
    "Action": "BLOCK",
    "Timestamp": 1581541799.988,
    "RuleNameWithinRuleGroup": "AWS#AWSManagedRulesSQLiRuleSet#SQLi_BODY",
    "Request": {
        "Country": "US",
        "URI": "/",
        "Headers": [
            {
                "Name": "Host",
                "Value": "alb-test-1EXAMPLE1.us-east-1.elb.amazonaws.com"
            },
            {
                "Name": "Content-Length",
                "Value": "7456"
            },
            {
                "Name": "User-Agent",
```

```
        "Value": "curl/7.53.1"
      },
      {
        "Name": "Accept",
        "Value": "/"
      },
      {
        "Name": "Content-Type",
        "Value": "application/x-www-form-urlencoded"
      }
    ],
    "ClientIP": "198.51.100.08",
    "Method": "POST",
    "HTTPVersion": "HTTP/1.1"
  },
  "Weight": 3
},
{
  "Action": "BLOCK",
  "Timestamp": 1581541799.846,
  "RuleNameWithinRuleGroup": "AWS#AWSManagedRulesSQLiRuleSet#SQLi_BODY",
  "Request": {
    "Country": "US",
    "URI": "/",
    "Headers": [
      {
        "Name": "Host",
        "Value": "alb-test-1EXAMPLE1.us-east-1.elb.amazonaws.com"
      },
      {
        "Name": "Content-Length",
        "Value": "7456"
      },
      {
        "Name": "User-Agent",
        "Value": "curl/7.53.1"
      },
      {
        "Name": "Accept",
        "Value": "/"
      },
      {
        "Name": "Content-Type",
        "Value": "application/x-www-form-urlencoded"
      }
    ]
  }
}
```

```
    }
  ],
  "ClientIP": "198.51.100.08",
  "Method": "POST",
  "HTTPVersion": "HTTP/1.1"
},
"Weight": 1
},
{
  "Action": "BLOCK",
  "Timestamp": 1581541799.4,
  "RuleNameWithinRuleGroup": "AWS#AWSManagedRulesSQLiRuleSet#SQLi_BODY",
  "Request": {
    "Country": "US",
    "URI": "/",
    "Headers": [
      {
        "Name": "Host",
        "Value": "alb-test-1EXAMPLE1.us-east-1.elb.amazonaws.com"
      },
      {
        "Name": "Content-Length",
        "Value": "7456"
      },
      {
        "Name": "User-Agent",
        "Value": "curl/7.53.1"
      },
      {
        "Name": "Accept",
        "Value": "/"
      },
      {
        "Name": "Content-Type",
        "Value": "application/x-www-form-urlencoded"
      }
    ]
  },
  "ClientIP": "198.51.100.08",
  "Method": "POST",
  "HTTPVersion": "HTTP/1.1"
},
"Weight": 1
}
],
```

```
"PopulationSize": 4
}
```

有关更多信息，请参阅 AWS WAF、AWS Firewall Manager 和 AWS Shield 高级开发者指南中的[查看 Web 请求示例](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetSampledRequests](#)中的。

## get-web-acl-for-resource

以下代码示例演示了如何使用 `get-web-acl-for-resource`。

### AWS CLI

检索与 AWS 资源关联的 Web ACL

以下内容 `get-web-acl-for-resource` 检索与指定资源关联的 Web ACL 的 JSON。

```
aws wafv2 get-web-acl-for-resource \
  --resource-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/
  app/waf-cli-alb/1ea17125f8b25a2a
```

输出：

```
{
  "WebACL": {
    "Capacity": 3,
    "Description": "",
    "Rules": [
      {
        "Priority": 1,
        "Action": {
          "Block": {}
        }
      },
      "VisibilityConfig": {
        "SampledRequestsEnabled": true,
        "CloudWatchMetricsEnabled": true,
        "MetricName": "testrule01"
      },
      "Name": "testrule01",
      "Statement": {
```



```
    "AndStatement":{
      "Statements":[
        {
          "ByteMatchStatement":{
            "PositionalConstraint":"EXACTLY",
            "TextTransformations":[
              {
                "Priority":0,
                "Type":"NONE"
              }
            ],
            "SearchString":"dGVzdHN0cmVuZw==",
            "FieldToMatch":{
              "UriPath":{
                }
            }
          }
        },
        {
          "SizeConstraintStatement":{
            "ComparisonOperator":"EQ",
            "TextTransformations":[
              {
                "Priority":0,
                "Type":"NONE"
              }
            ],
            "FieldToMatch":{
              "QueryString":{
                }
            }
          },
          "Size":0
        }
      ]
    }
  ],
  "VisibilityConfig":{
    "SampledRequestsEnabled":true,
    "CloudWatchMetricsEnabled":true,
```

```
        "MetricName":"test01"
      },
      "DefaultAction":{
        "Allow":{

        }
      },
      "Id":"9a1b2c3d4-5678-90ab-cdef-EXAMPLE11111  ",
      "ARN":"arn:aws:wafv2:us-west-2:123456789012:regional/webacl/test01/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111  ",
      "Name":"test01"
    }
  }
}
```

有关更多信息，请参阅《WAF》、《Fi AWS rewall Manager AWS》和《AWS Sh [ield 高级开发者指南](#)》中的“[将 Web ACL 与 AWS 资源关联或取消关联](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetWebAclForResource](#)中的。

## get-web-acl

以下代码示例演示了如何使用 get-web-acl。

### AWS CLI

#### 检索 Web ACL

以下内容get-web-acl检索具有指定名称、范围和 ID 的 Web ACL。您可以从命令create-web-acl和中获取 Web ACL 的 ID list-web-acls。

```
aws wafv2 get-web-acl \
  --name test01 \
  --scope REGIONAL \
  --id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{
  "WebACL":{
    "Capacity":3,
    "Description":"",
    "Rules":[
      {
```

```
"Priority":1,
"Action":{
  "Block":{

  }
},
"VisibilityConfig":{
  "SampledRequestsEnabled":true,
  "CloudWatchMetricsEnabled":true,
  "MetricName":"testrule01"
},
"Name":"testrule01",
"Statement":{
  "AndStatement":{
    "Statements":[
      {
        "ByteMatchStatement":{
          "PositionalConstraint":"EXACTLY",
          "TextTransformations":[
            {
              "Priority":0,
              "Type":"NONE"
            }
          ],
          "SearchString":"dGVzdHN0cm1uZw==",
          "FieldToMatch":{
            "UriPath":{

            }
          }
        }
      },
      {
        "SizeConstraintStatement":{
          "ComparisonOperator":"EQ",
          "TextTransformations":[
            {
              "Priority":0,
              "Type":"NONE"
            }
          ],
          "FieldToMatch":{
            "QueryString":{

            }
          }
        }
      }
    ]
  }
}
```

```

    }
  },
  "Size":0
}
]
}
}
},
"VisibilityConfig":{
  "SampledRequestsEnabled":true,
  "CloudWatchMetricsEnabled":true,
  "MetricName":"test01"
},
"DefaultAction":{
  "Allow":{
  }
},
"Id":"a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
"ARN":"arn:aws:wafv2:us-west-2:123456789012:regional/webacl/test01/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
"Name":"test01"
},
"LockToken":"e3db7e2c-d58b-4ee6-8346-6aec5511c6fb"
}

```

有关更多信息，请参阅 [W AWS AF、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的管理和使用 Web 访问控制列表 \(Web ACL\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetWebAcl](#) 中的。

## list-available-managed-rule-groups

以下代码示例演示了如何使用 `list-available-managed-rule-groups`。

### AWS CLI

#### 检索托管规则组

以下内容 `list-available-managed-rule-groups` 返回当前可用于您的 Web ACL 的所有托管规则组的列表。

```
aws wafv2 list-available-managed-rule-groups \  
--scope REGIONAL
```

输出：

```
{  
  "ManagedRuleGroups": [  
    {  
      "VendorName": "AWS",  
      "Name": "AWSManagedRulesCommonRuleSet",  
      "Description": "Contains rules that are generally applicable to web  
applications. This provides protection against exploitation of a wide range of  
vulnerabilities, including those described in OWASP publications and common Common  
Vulnerabilities and Exposures (CVE)."  
    },  
    {  
      "VendorName": "AWS",  
      "Name": "AWSManagedRulesAdminProtectionRuleSet",  
      "Description": "Contains rules that allow you to block external access  
to exposed admin pages. This may be useful if you are running third-party software  
or would like to reduce the risk of a malicious actor gaining administrative access  
to your application."  
    },  
    {  
      "VendorName": "AWS",  
      "Name": "AWSManagedRulesKnownBadInputsRuleSet",  
      "Description": "Contains rules that allow you to block request patterns  
that are known to be invalid and are associated with exploitation or discovery of  
vulnerabilities. This can help reduce the risk of a malicious actor discovering a  
vulnerable application."  
    },  
    {  
      "VendorName": "AWS",  
      "Name": "AWSManagedRulesSQLiRuleSet",  
      "Description": "Contains rules that allow you to block request patterns  
associated with exploitation of SQL databases, like SQL injection attacks. This can  
help prevent remote injection of unauthorized queries."  
    },  
    {  
      "VendorName": "AWS",  
      "Name": "AWSManagedRulesLinuxRuleSet",  
      "Description": "Contains rules that block request patterns associated  
with exploitation of vulnerabilities specific to Linux, including LFI attacks. This
```

```
can help prevent attacks that expose file contents or execute code for which the
attacker should not have had access."
    },
    {
        "VendorName": "AWS",
        "Name": "AWSManagedRulesUnixRuleSet",
        "Description": "Contains rules that block request patterns associated
with exploiting vulnerabilities specific to POSIX/POSIX-like OS, including LFI
attacks. This can help prevent attacks that expose file contents or execute code
for which access should not been allowed."
    },
    {
        "VendorName": "AWS",
        "Name": "AWSManagedRulesWindowsRuleSet",
        "Description": "Contains rules that block request patterns associated
with exploiting vulnerabilities specific to Windows, (e.g., PowerShell commands).
This can help prevent exploits that allow attacker to run unauthorized commands or
execute malicious code."
    },
    {
        "VendorName": "AWS",
        "Name": "AWSManagedRulesPHPRuleSet",
        "Description": "Contains rules that block request patterns associated
with exploiting vulnerabilities specific to the use of the PHP, including injection
of unsafe PHP functions. This can help prevent exploits that allow an attacker to
remotely execute code or commands."
    },
    {
        "VendorName": "AWS",
        "Name": "AWSManagedRulesWordPressRuleSet",
        "Description": "The WordPress Applications group contains rules that
block request patterns associated with the exploitation of vulnerabilities specific
to WordPress sites."
    },
    {
        "VendorName": "AWS",
        "Name": "AWSManagedRulesAmazonIpReputationList",
        "Description": "This group contains rules that are based on Amazon
threat intelligence. This is useful if you would like to block sources associated
with bots or other threats."
    }
]
}
```

有关更多信息，请参阅 AWS WAF、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的[托管规则组](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAvailableManagedRuleGroups](#)中的。

## list-ip-sets

以下代码示例演示了如何使用 list-ip-sets。

### AWS CLI

#### 检索 IP 集列表

以下内容list-ip-sets检索该账户中具有区域范围的所有 IP 集。

```
aws wafv2 list-ip-sets \
  --scope REGIONAL
```

输出：

```
{
  "IPSets": [
    {
      "ARN": "arn:aws:wafv2:us-west-2:123456789012:regional/ipset/testip/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "Description": "",
      "Name": "testip",
      "LockToken": "0674c84b-0304-47fe-8728-c6bff46af8fc",
      "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111  "
    }
  ],
  "NextMarker": "testip"
}
```

有关更多信息，请参阅 AWS WAF、Firewal AWS I Manager 和 AWS Shield 高级开发者指南中的 IP 集和[正则表达式模式集](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListIpSets](#)中的。

## list-logging-configurations

以下代码示例演示了如何使用 list-logging-configurations。

## AWS CLI

### 检索某个区域的所有日志配置列表

以下内容 `list-logging-configurations` 检索 Web ACL 的所有日志配置，这些配置仅限于该区域的区域使用。 `us-west-2`

```
aws wafv2 list-logging-configurations \  
  --scope REGIONAL \  
  --region us-west-2
```

输出：

```
{  
  "LoggingConfigurations": [  
    {  
      "ResourceArn": "arn:aws:wafv2:us-west-2:123456789012:regional/webacl/  
test-2/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "RedactedFields": [  
        {  
          "QueryString": {  
            }  
        }  
      ],  
      "LogDestinationConfigs": [  
        "arn:aws:firehose:us-west-2:123456789012:deliverystream/aws-waf-  
logs-test"  
      ]  
    },  
    {  
      "ResourceArn": "arn:aws:wafv2:us-west-2:123456789012:regional/webacl/  
test/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
      "RedactedFields": [  
        {  
          "Method": {  
            }  
        }  
      ],  
      "LogDestinationConfigs": [  
        "arn:aws:firehose:us-west-2:123456789012:deliverystream/aws-waf-  
logs-custom-transformation"  
      ]  
    }  
  ]  
}
```



```

    ]
  }
]
}

```

有关更多信息，请参阅 [W AWS AF](#)、[Fi AWS rewall Manager](#) 和 [AWS Shield 高级开发者指南](#) 中的 [记录 Web ACL 流量信息](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 ListLoggingConfigurations](#) 中的。

## list-regex-pattern-sets

以下代码示例演示了如何使用 `list-regex-pattern-sets`。

### AWS CLI

#### 检索正则表达式模式集列表

以下内容 `list-regex-pattern-sets` 检索在该区域中定义的账户的所有正则表达式模式集。us-west-2

```

aws wafv2 list-regex-pattern-sets \
--scope REGIONAL \
--region us-west-2

```

输出：

```

{
  "NextMarker": "regexPatterSet01",
  "RegexPatternSets": [
    {
      "ARN": "arn:aws:wafv2:us-west-2:123456789012:regional/regexpatternset/regexPatterSet01/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "Description": "Test web-acl",
      "Name": "regexPatterSet01",
      "LockToken": "f17743f7-0000-0000-0000-19a8b93bfb01",
      "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    }
  ]
}

```

有关更多信息，请参阅 AWS WAF、Firewal AWS I Manager 和 AWS Shield 高级开发者指南中的 IP 集和[正则表达式模式集](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRegexPatternSets](#)中的。

## list-resources-for-web-acl

以下代码示例演示了如何使用 list-resources-for-web-acl。

### AWS CLI

检索与 Web ACL 关联的资源

以下内容list-resources-for-web-acl检索当前与该地区us-west-2指定网页 ACL 关联的 API Gateway REST API 资源。

```
aws wafv2 list-resources-for-web-acl \
  --web-acl-arn arn:aws:wafv2:us-west-2:123456789012:regional/webacl/TestWebAcl/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
  --resource-type API_GATEWAY \
  --region us-west-2
```

输出：

```
{
  "ResourceArns": [
    "arn:aws:apigateway:us-west-2:./restapis/EXAMPLE111/stages/testing"
  ]
}
```

有关更多信息，请参阅《WAF》、《Fi AWS rewall Manager AWS》和《AWS Sh [ield 高级开发](#)者指南》中的“[将 Web ACL 与 AWS 资源关联或取消关联](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResourcesForWebAcl](#)中的。

## list-rule-groups

以下代码示例演示了如何使用 list-rule-groups。

### AWS CLI

检索自定义规则组列表

以下内容`list-rule-groups`检索为指定范围和区域位置的账户定义的所有自定义规则组。

```
aws wafv2 list-rule-groups \  
  --scope REGIONAL \  
  --region us-west-2
```

输出：

```
{  
  "RuleGroups":[  
    {  
      "ARN":"arn:aws:wafv2:us-west-2:123456789012:regional/rulegroup/  
TestRuleGroup/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "Description":"","  
      "Name":"TestRuleGroup",  
      "LockToken":"1eb5ec48-0000-0000-0000-ee9b906c541e",  
      "Id":"a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"  
    },  
    {  
      "ARN":"arn:aws:wafv2:us-west-2:123456789012:regional/rulegroup/test/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",  
      "Description":"","  
      "Name":"test",  
      "LockToken":"b0f4583e-998b-4880-9069-3fbe45738b43",  
      "Id":"a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"  
    }  
  ],  
  "NextMarker":"test"  
}
```

有关更多信息，请参阅《AWS WAF》、《Fi AWS rewall Manager》和《AWS Shield 高级开发者指南》中的“管理[自己的规则组](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListRuleGroups](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

检索 AWS WAF 资源的所有标签

以下内容 `list-tags-for-resource` 检索指定 Web ACL 的所有标签键和值对的列表。

```
aws wafv2 list-tags-for-resource \  
  --resource-arn arn:aws:wafv2:us-west-2:123456789012:regional/webacl/testwebacl/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

输出：

```
{  
  "NextMarker": "",  
  "TagInfoForResource": {  
    "ResourceARN": "arn:aws:wafv2:us-west-2:123456789012:regional/webacl/  
testwebacl/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "TagList": [  
  
    ]  
  }  
}
```

有关更多信息，请参阅 [AWS WAF、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的 AWS WAF 入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListTagsForResource](#) 中的。

## list-web-acls

以下代码示例演示了如何使用 `list-web-acls`。

AWS CLI

检索作用域的 Web ACL

以下内容 `list-web-acls` 检索为指定范围内的账户定义的所有 Web ACL。

```
aws wafv2 list-web-acls \  
  --scope REGIONAL
```

输出：

```
{  
  "NextMarker": "Testt",
```

```

    "WebACLs": [
      {
        "ARN": "arn:aws:wafv2:us-west-2:123456789012:regional/webacl/Testt/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "Description": "sssss",
        "Name": "Testt",
        "LockToken": "7f36cb30-74ef-4cff-8cd4-a77e1aba1746",
        "Id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
      }
    ]
  }
}

```

有关更多信息，请参阅 [W AWS AF、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的管理和使用 Web 访问控制列表 \(Web ACL\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ListWebAcls](#) 中的。

## put-logging-configuration

以下代码示例演示了如何使用 put-logging-configuration。

### AWS CLI

向 Web ACL 添加日志配置

以下内容 put-logging-configuration 将 Amazon Kinesis Data Firehose aws-waf-logs-custom-transformation 日志配置添加到指定的网络 ACL 中，但没有从日志中删除任何字段。

```

aws wafv2 put-logging-configuration \
  --logging-configuration ResourceArn=arn:aws:wafv2:us-
west-2:123456789012:regional/webacl/test-cli/a1b2c3d4-5678-90ab-
cdef-EXAMPLE11111,LogDestinationConfigs=arn:aws:firehose:us-
west-2:123456789012:deliverystream/aws-waf-logs-custom-transformation \
  --region us-west-2

```

输出：

```

{
  "LoggingConfiguration": {
    "ResourceArn": "arn:aws:wafv2:us-west-2:123456789012:regional/webacl/test-
cli/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "LogDestinationConfigs": [

```

```
        "arn:aws:firehose:us-west-2:123456789012:deliverystream/aws-waf-logs-
        custom-transformation"
      ]
    }
  }
}
```

有关更多信息，请参阅 [AWS AF](#)、[AWS rewall Manager](#) 和 [AWS Shield 高级开发者指南](#) 中的 [记录 Web ACL 流量信息](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 PutLoggingConfiguration](#) 中的。

## tag-resource

以下代码示例演示了如何使用 tag-resource。

### AWS CLI

向 AWS WAF 资源添加标签

以下 tag-resource 示例向指定的 Web ACL 添加一个密钥为 Name、值设置 AWSWAF 为的标签。

```
aws wafv2 tag-resource \
  --resource-arn arn:aws:wafv2:us-west-2:123456789012:regional/webacl/
  apiGatewayWebAcl/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
  --tags Key=Name,Value=AWSWAF
```

此命令不生成任何输出。

有关更多信息，请参阅 [AWS WAF](#)、[AWS rewall Manager](#) 和 [AWS Shield 高级开发者指南](#) 中的 [AWS WAF 入门](#)。

- 有关 API 的详细信息，请参阅 [AWS CLI 命令参考 TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 untag-resource。

### AWS CLI

从 AWS WAF 资源中移除标签

以下 untag-resource 示例 KeyName 从指定的 Web ACL 中删除带有密钥的标签。

```
aws wafv2 untag-resource \  
  --resource-arn arn:aws:wafv2:us-west-2:123456789012:regional/webacl/  
  apiGatewayWebAcl/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --tag-keys "KeyName"
```

此命令不生成任何输出。

有关更多信息，请参阅 [AWS WAF](#)、[AWS Firewall Manager](#) 和 [AWS Shield 高级开发者指南中的 AWS WAF 入门](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UntagResource](#) 中的。

## update-ip-set

以下代码示例演示了如何使用 update-ip-set。

### AWS CLI

#### 修改现有 IP 集的设置

以下内容 update-ip-set 更新了指定 IP 集的设置。此调用需要一个 ID（您可以从调用中获取）和一个可以从调用中获取的锁定令牌，list-ip-sets 以及 get-ip-set。list-ip-sets 此调用还会返回一个锁定令牌，您可以将其用于后续更新。

```
aws wafv2 update-ip-set \  
  --name testip \  
  --scope REGIONAL \  
  --id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --addresses 198.51.100.0/16 \  
  --lock-token 447e55ac-2396-4c6d-b9f9-86b67c17f8b5
```

输出：

```
{  
  "NextLockToken": "0674c84b-0304-47fe-8728-c6bff46af8fc"  
}
```

有关更多信息，请参阅 [AWS WAF](#)、[AWS Firewall Manager](#) 和 [AWS Shield 高级开发者指南中的 IP 集和正则表达式模式集](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateIpSet](#) 中的。

## update-regex-pattern-set

以下代码示例演示了如何使用 update-regex-pattern-set。

### AWS CLI

#### 修改现有正则表达式模式集的设置

以下内容update-regex-pattern-set更新了指定正则表达式模式集的设置。此调用需要一个 ID ( 您可以从呼叫中获取 ) 和一个可以从调用中获取的锁定令牌 , list-regex-pattern-sets以及get-regex-pattern-set。list-regex-pattern-sets此调用还会返回一个锁定令牌 , 您可以将其用于后续更新。

```
aws wafv2 update-regex-pattern-set \  
  --name ExampleRegex \  
  --scope REGIONAL \  
  --id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \  
  --regular-expression-list RegexString="^.+ $" \  
  --lock-token ed207e9c-82e9-4a77-aadd-81e6173ab7eb
```

输出 :

```
{  
  "NextLockToken": "12ebc73e-fa68-417d-a9b8-2bdd761a4fa5"  
}
```

有关更多信息 , 请参阅 AWS WAF、Firewal AWS I Manager 和 AWS Shield 高级开发者指南中的 [IP 集](#)和[正则表达式模式集](#)。

- 有关 API 的详细信息 , 请参阅AWS CLI 命令参考[UpdateRegexPatternSet](#)中的。

## update-rule-group

以下代码示例演示了如何使用 update-rule-group。

### AWS CLI

#### 更新自定义规则组

以下内容update-rule-group更改了现有自定义规则组的可见性配置。此调用需要一个 ID ( 您可以从呼叫中获取 ) 和一个可以从调用中获取的锁定令牌 , list-rule-groups以及get-rule-group。list-rule-groups此调用还会返回一个锁定令牌 , 您可以将其用于后续更新。



```
aws wafv2 update-rule-group \
  --name TestRuleGroup \
  --scope REGIONAL \
  --id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
  --lock-token 7b3bcec2-0000-0000-0000-563bf47249f0 \
  --visibility-config
  SampledRequestsEnabled=false,CloudWatchMetricsEnabled=false,MetricName=TestMetricsForRuleGr
  \
  --region us-west-2
```

输出：

```
{
  "NextLockToken": "1eb5ec48-0000-0000-0000-ee9b906c541e"
}
```

有关更多信息，请参阅《AWS WAF》、《AWS Firewall Manager》和《AWS Shield 高级开发者指南》中的“管理[自己的规则组](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateRuleGroup](#) 中的。

## update-web-acl

以下代码示例演示了如何使用 update-web-acl。

### AWS CLI

#### 更新 Web ACL

以下内容 update-web-acl 更改了现有 Web ACL 的设置。此呼叫需要一个 ID (您可以从呼叫中获取) list-web-acls、锁定令牌和其他设置 (您可以从呼叫中获取) get-web-acl。此调用还会返回一个锁定令牌，您可以将其用于后续更新。

```
aws wafv2 update-web-acl \
  --name TestWebAcl \
  --scope REGIONAL \
  --id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
  --lock-token 2294b3a1-0000-0000-0000-a3ae04329de9 \
  --default-action Block={} \
  --visibility-config
  SampledRequestsEnabled=false,CloudWatchMetricsEnabled=false,MetricName=NewMetricTestWebAcl
  \
```

```
--rules file://waf-rule.json \  
--region us-west-2
```

输出：

```
{  
  "NextLockToken": "714a0cfb-0000-0000-0000-2959c8b9a684"  
}
```

有关更多信息，请参阅 [W AWS AF、Fi AWS rewall Manager 和 AWS Shield 高级开发者指南中的管理和使用 Web 访问控制列表 \(Web ACL\)](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateWebAcl](#) 中的。

## 使用亚马逊的 WorkDocs 示例 AWS CLI

以下代码示例向您展示如何在 Amazon 中使用来执行操作和实现常见场景 WorkDocs。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **abort-document-version-upload**

以下代码示例演示了如何使用 abort-document-version-upload。

AWS CLI

停止上传文档版本

此示例停止先前启动的文档版本上传。

**命令:**

```
aws workdocs abort-document-version-upload --document-id
feaba64d4efdf271c2521b60a2a44a8f057e84beaabbe22f01267313209835f2 --version-id
1536773972914-ddb67663e782e7ce8455ebc962217cf9f9e47b5a9a702e5c84dccc417da9313
```

**输出:**

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AbortDocumentVersionUpload](#)中的。

**activate-user**

以下代码示例演示了如何使用 activate-user。

**AWS CLI****激活用户**

此示例激活了一个不活跃的用户。

**命令:**

```
aws workdocs activate-user --user-id
"S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c"
```

**输出:**

```
{
  "User": {
    "Id": "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
    "Username": "exampleUser",
    "EmailAddress": "exampleUser@site.awsapps.com",
    "GivenName": "Example",
    "Surname": "User",
    "OrganizationId": "d-926726012c",
    "RootFolderId":
"75f67c183aa1217409ac87576a45c03a5df5e6d8c51c35c01669970538e86cd0",
    "RecycleBinFolderId":
"642b7dd3e60b14204534f3df7b1959e01b5d170f8c2707f410e40a8149120a57",
    "Status": "ACTIVE",
```

```

    "Type": "MINIMALUSER",
    "CreatedTimestamp": 1521226107.747,
    "ModifiedTimestamp": 1525297406.462,
    "Storage": {
      "StorageUtilizedInBytes": 0,
      "StorageRule": {
        "StorageAllocatedInBytes": 0,
        "StorageType": "QUOTA"
      }
    }
  }
}
}
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ActivateUser](#)中的。

## add-resource-permissions

以下代码示例演示了如何使用 add-resource-permissions。

### AWS CLI

为资源添加权限

此示例为指定委托人添加资源权限。

命令:

```

aws workdocs add-resource-permissions --resource-id
d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65 --principals
Id=anonymous,Type=ANONYMOUS,Role=VIEWER

```

输出:

```

{
  "ShareResults": [
    {
      "PrincipalId": "anonymous",
      "Role": "VIEWER",
      "Status": "SUCCESS",
      "ShareId":
"d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65",
      "StatusMessage": ""
    }
  ]
}

```

```
    }  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AddResourcePermissions](#)中的。

## create-comment

以下代码示例演示了如何使用 create-comment。

### AWS CLI

#### 添加新评论

此示例向指定的文档版本添加了新注释。

命令:

```
aws workdocs create-comment --document-id  
15df51e0335cfcc6a2e4de9dd8be9f22ee40545ad9176f54758dcf903be982d3 --version-id  
1521672507741-9f7df0ea5dd0b121c4f3564a0c7c0b4da95cd12c635d3c442af337a88e297920 --  
text "This is a comment."
```

输出:

```
{  
  "Comment": {  
    "CommentId": "1534799058197-  
c7f5c84de9115875bbca93e0367bbebac609541d461636b760849b88b1609dd5",  
    "ThreadId": "1534799058197-  
c7f5c84de9115875bbca93e0367bbebac609541d461636b760849b88b1609dd5",  
    "Text": "This is a comment.",  
    "Contributor": {  
      "Id": "arn:aws:iam::123456789123:user/exampleUser",  
      "Username": "exampleUser",  
      "GivenName": "Example",  
      "Surname": "User",  
      "Status": "ACTIVE"  
    },  
    "CreatedTimestamp": 1534799058.197,  
    "Status": "PUBLISHED",  
    "Visibility": "PUBLIC"  
  }  
}
```

```
}  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateComment](#)中的。

## create-custom-metadata

以下代码示例演示了如何使用 `create-custom-metadata`。

### AWS CLI

#### 创建自定义元数据

此示例为指定文档创建自定义元数据。

命令:

```
aws workdocs create-custom-metadata --resource-id  
d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65 --custom-metadata  
KeyName1=example,KeyName2=example2
```

输出 :

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateCustomMetadata](#)中的。

## create-folder

以下代码示例演示了如何使用 `create-folder`。

### AWS CLI

#### 创建文件夹

此示例创建了一个文件夹。

命令:

```
aws workdocs create-folder --name documents --parent-folder-id  
1ece93e5fe75315c7407c4967918b4fd9da87ddb2a588e67b7fdaf4a98fde678
```

输出：

```
{
  "Metadata": {
    "Id": "50893c0af679524d1a0e0651130ed6d073e1a05f95bd12c42dcde5d35634ed08",
    "Name": "documents",
    "CreatorId": "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
    "ParentFolderId":
"1ece93e5fe75315c7407c4967918b4fd9da87ddb2a588e67b7fdaf4a98fde678",
    "CreatedTimestamp": 1534450467.622,
    "ModifiedTimestamp": 1534450467.622,
    "ResourceState": "ACTIVE",
    "Signature": "",
    "Size": 0,
    "LatestVersionSize": 0
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateFolder](#)中的。

## create-labels

以下代码示例演示了如何使用 create-labels。

### AWS CLI

#### 创建标签

此示例为文档创建了一系列标签。

命令：

```
aws workdocs create-labels --resource-id
d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65 --labels
"documents" "examples" "my_documents"
```

输出：

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateLabels](#)中的。

## create-notification-subscription

以下代码示例演示了如何使用 `create-notification-subscription`。

### AWS CLI

#### 创建通知订阅

以下 `create-notification-subscription` 示例为指定的 Amazon WorkDocs 组织配置通知订阅。

```
aws workdocs create-notification-subscription \  
  --organization-id d-123456789c \  
  --protocol HTTPS \  
  --subscription-type ALL \  
  --notification-endpoint "https://example.com/example"
```

输出：

```
{  
  "Subscription": {  
    "SubscriptionId": "123ab4c5-678d-901e-f23g-45h6789j0123",  
    "EndPoint": "https://example.com/example",  
    "Protocol": "HTTPS"  
  }  
}
```

有关更多信息，请参阅 [《Amazon WorkDocs 开发者指南》中的订阅通知](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateNotificationSubscription](#) 中的。

## create-user

以下代码示例演示了如何使用 `create-user`。

### AWS CLI

#### 创建新用户

此示例在 Simple AD 或 Microsoft AD 目录中创建了一个新用户。

命令：



```
aws workdocs create-user --organization-id d-926726012c --username exampleUser2
--email-address exampleUser2@site.awsapps.com --given-name example2Name --surname
example2Surname --password examplePa$$w0rd
```

输出：

```
{
  "User": {
    "Id": "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
    "Username": "exampleUser2",
    "EmailAddress": "exampleUser2@site.awsapps.com",
    "GivenName": "example2Name",
    "Surname": "example2Surname",
    "OrganizationId": "d-926726012c",
    "RootFolderId":
"35b886cb17198cbd547655e58b025dff0cf34aaed638be52009567e23dc67390",
    "RecycleBinFolderId":
"9858c3e9ed4c2460dde9aadb4c69fde998070dd46e5e985bd08ec6169ea249ff",
    "Status": "ACTIVE",
    "Type": "MINIMALUSER",
    "CreatedTimestamp": 1535478836.584,
    "ModifiedTimestamp": 1535478836.584,
    "Storage": {
      "StorageUtilizedInBytes": 0,
      "StorageRule": {
        "StorageAllocatedInBytes": 0,
        "StorageType": "QUOTA"
      }
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUser](#)中的。

## deactivate-user

以下代码示例演示了如何使用 deactivate-user。

### AWS CLI

#### 停用用户

此示例停用活跃用户。

命令:

```
aws workdocs deactivate-user --user-id
"S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c"
```

输出:

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeactivateUser](#)中的。

## delete-comment

以下代码示例演示了如何使用 delete-comment。

### AWS CLI

从文档版本中删除指定注释

此示例从指定文档版本中删除指定的注释。

命令:

```
aws workdocs delete-comment --document-id
15df51e0335cfcc6a2e4de9dd8be9f22ee40545ad9176f54758dcf903be982d3 --version-id
1521672507741-9f7df0ea5dd0b121c4f3564a0c7c0b4da95cd12c635d3c442af337a88e297920 --
comment-id 1534799058197-
c7f5c84de9115875bbca93e0367bbebac609541d461636b760849b88b1609dd5
```

输出:

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteComment](#)中的。

## delete-custom-metadata

以下代码示例演示了如何使用 delete-custom-metadata。

## AWS CLI

### 从资源中删除自定义元数据

此示例从指定资源中删除所有自定义元数据。

命令:

```
aws workdocs delete-custom-metadata --resource-id
d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65 --delete-all
```

输出 :

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCustomMetadata](#)中的。

## delete-document

以下代码示例演示了如何使用 delete-document。

## AWS CLI

### 删除文档

此示例删除了指定的文档。

命令:

```
aws workdocs delete-document --document-id
b83ed5e5b167b65ef69de9d597627ff1a0d4f07a45e67f1fab7d26b54427de0a
```

输出 :

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDocument](#)中的。

## delete-folder-contents

以下代码示例演示了如何使用 delete-folder-contents。

## AWS CLI

### 删除文件夹的内容

此示例删除指定文件夹的内容。

命令:

```
aws workdocs delete-folder-contents --folder-id
26fa8aa4ba2071447c194f7b150b07149dbdb9e1c8a301872dcd93a4735ce65d
```

输出:

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFolderContents](#)中的。

## delete-folder

以下代码示例演示了如何使用 delete-folder。

## AWS CLI

### 删除文件夹

此示例删除了指定的文件夹。

命令:

```
aws workdocs delete-folder --folder-id
26fa8aa4ba2071447c194f7b150b07149dbdb9e1c8a301872dcd93a4735ce65d
```

输出:

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteFolder](#)中的。

## delete-labels

以下代码示例演示了如何使用 delete-labels。

## AWS CLI

### 要删除标签

此示例从文档中删除指定的标签。

命令:

```
aws workdocs delete-labels --resource-id
d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65 --labels
"documents" "examples"
```

输出 :

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteLabels](#)中的。

## delete-notification-subscription

以下代码示例演示了如何使用 delete-notification-subscription。

## AWS CLI

### 删除通知订阅

以下delete-notification-subscription示例删除了指定的通知订阅。

```
aws workdocs delete-notification-subscription \
--subscription-id 123ab4c5-678d-901e-f23g-45h6789j0123 \
--organization-id d-123456789c
```

此命令不生成任何输出。

有关更多信息，请参阅[《Amazon WorkDocs 开发者指南》中的订阅通知](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteNotificationSubscription](#)中的。

## delete-user

以下代码示例演示了如何使用 delete-user。

## AWS CLI

### 删除用户

此示例删除一个用户。

命令:

```
aws workdocs delete-user --user-id
"S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c"
```

输出:

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUser](#)中的。

## describe-activities

以下代码示例演示了如何使用 describe-activities。

## AWS CLI

### 获取用户活动列表

此示例返回指定组织的最新用户活动列表，并对最新的两个活动设置了限制。

命令:

```
aws workdocs describe-activities --organization-id d-926726012c --limit 2
```

输出:

```
{
  "UserActivities": [
    {
      "Type": "DOCUMENT_VERSION_DOWNLOADED",
      "TimeStamp": 1534800122.17,
      "Initiator": {
        "Id": "arn:aws:iam::123456789123:user/exampleUser"
      },
      "ResourceMetadata": {
```

```

        "Type": "document",
        "Name": "updatedDoc",
        "Id":
"15df51e0335cfcc6a2e4de9dd8be9f22ee40545ad9176f54758dcf903be982d3",
        "Owner": {
            "Id":
"S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
            "GivenName": "exampleName",
            "Surname": "exampleSurname"
        }
    },
    {
        "Type": "DOCUMENT_VERSION_VIEWED",
        "TimeStamp": 1534799079.207,
        "Initiator": {
            "Id": "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
            "GivenName": "exampleName",
            "Surname": "exampleSurname"
        },
        "ResourceMetadata": {
            "Type": "document",
            "Name": "updatedDoc",
            "Id":
"15df51e0335cfcc6a2e4de9dd8be9f22ee40545ad9176f54758dcf903be982d3",
            "Owner": {
                "Id":
"S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
                "GivenName": "exampleName",
                "Surname": "exampleSurname"
            }
        }
    }
],
"Marker":
"DnF1ZXJ5VGhlbkZldGNoAgAAAAAAS7Fm1TaU10d1FTU1h1UU00VVFibD1RWhcAAAAAAAJTRY3bWh5eUgzaVF1ZX"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeActivities](#)中的。

## describe-comments

以下代码示例演示了如何使用 describe-comments。

## AWS CLI

列出指定文档版本的所有评论

此示例列出了指定文档版本的所有注释。

命令:

```
aws workdocs describe-comments --document-id
15df51e0335cfcc6a2e4de9dd8be9f22ee40545ad9176f54758dcf903be982d3 --version-id
1521672507741-9f7df0ea5dd0b121c4f3564a0c7c0b4da95cd12c635d3c442af337a88e297920
```

输出 :

```
{
  "Comments": [
    {
      "CommentId": "1534799058197-
c7f5c84de9115875bbca93e0367bbebac609541d461636b760849b88b1609dd5",
      "ThreadId": "1534799058197-
c7f5c84de9115875bbca93e0367bbebac609541d461636b760849b88b1609dd5",
      "Text": "This is a comment.",
      "Contributor": {
        "Username": "arn:aws:iam::123456789123:user/exampleUser",
        "Type": "USER"
      },
      "CreatedTimestamp": 1534799058.197,
      "Status": "PUBLISHED",
      "Visibility": "PUBLIC"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeComments](#)中的。

## describe-document-versions

以下代码示例演示了如何使用 describe-document-versions。

## AWS CLI

检索文档的版本



此示例检索指定文档的文档版本，包括初始化的版本和源文档的 URL。

命令:

```
aws workdocs describe-document-versions --document-id
d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65 --fields SOURCE
```

输出:

```
{
  "DocumentVersions": [
    {
      "Id":
      "1534452029587-15e129dfc187505c407588df255be83de2920d733859f1d2762411d22a83e3ef",
      "Name": "exampleDoc.docx",
      "ContentType": "application/vnd.openxmlformats-
officedocument.wordprocessingml.document",
      "Size": 13922,
      "Signature": "1a23456b78901c23d4ef56gh7EXAMPLE",
      "Status": "ACTIVE",
      "CreatedTimestamp": 1534452029.587,
      "ModifiedTimestamp": 1534452029.849,
      "CreatorId":
      "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
      "Source": {
        "ORIGINAL": "https://gb-us-west-2-prod-doc-source.s3.us-
west-2.amazonaws.com/
d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65/1534452029587-15e129dfc1875
response-content-disposition=attachment%3B%20filename%2A
%3DUTF-8%27%27exampleDoc29.docx&X-Amz-Algorithm=AWS1-ABCD-EFG234&X-Amz-
Date=20180816T204149Z&X-Amz-SignedHeaders=host&X-Amz-Expires=900&X-Amz-
Credential=AKIAIOSFODNN7EXAMPLE%2F20180816%2Fus-west-2%2Fs3%2Faws1_request&X-Amz-
Signature=01Ab2c34d567e8f90123g456hi78j901k2345678l901234mno56pqr78EXAMPLE"
      }
    },
    {
      "Id": "1529005196082-
bb75fa19abc287699cb07147f75816dce43a53a10f28dc001bf61ef2fab01c59",
      "Name": "exampleDoc.pdf",
      "ContentType": "application/pdf",
      "Size": 425916,
      "Signature": "1a23456b78901c23d4ef56gh7EXAMPLE",
      "Status": "ACTIVE",
```

```

    "CreatedTimestamp": 1529005196.082,
    "ModifiedTimestamp": 1529005196.796,
    "CreatorId":
    "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
    "Source": {
      "ORIGINAL": "https://gb-us-west-2-prod-doc-source.s3.us-
west-2.amazonaws.com/
d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65/1529005196082-
bb75fa19abc287699cb07147f75816dce43a53a10f28dc001bf61ef2fab01c59?
response-content-disposition=attachment%3B%20filename%2A
%3DUTF-8%27%27exampleDoc29.pdf&X-Amz-Algorithm=AWS1-ABCD-EFG234&X-Amz-
Date=20180816T204149Z&X-Amz-SignedHeaders=host&X-Amz-Expires=900&X-Amz-
Credential=AKIAIOSFODNN7EXAMPLE%2F20180816%2Fus-west-2%2Fs3%2Faws1_request&X-Amz-
Signature=01Ab2c34d567e8f90123g456hi78j901k2345678l901234mno56pqr78EXAMPLE"
    }
  }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeDocumentVersions](#)中的。

## describe-folder-contents

以下代码示例演示了如何使用 `describe-folder-contents`。

### AWS CLI

#### 描述文件夹的内容

此示例描述了按日期升序排序的指定文件夹（包括其文档和子文件夹）的所有活动内容。

#### 命令:

```

aws workdocs describe-folder-contents --folder-id
1ece93e5fe75315c7407c4967918b4fd9da87ddb2a588e67b7fdaf4a98fde678 --sort DATE --
order ASCENDING --type ALL

```

#### 输出:

```

{
  "Folders": [

```

```

    {
      "Id": "50893c0af679524d1a0e0651130ed6d073e1a05f95bd12c42dcde5d35634ed08",
      "Name": "testing",
      "CreatorId":
    "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
      "ParentFolderId":
    "1ece93e5fe75315c7407c4967918b4fd9da87ddb2a588e67b7fdaf4a98fde678",
      "CreatedTimestamp": 1534450467.622,
      "ModifiedTimestamp": 1534451113.504,
      "ResourceState": "ACTIVE",
      "Signature": "1a23456b78901c23d4ef56gh7EXAMPLE",
      "Size": 23019,
      "LatestVersionSize": 11537
    }
  ],
  "Documents": [
    {
      "Id": "d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65",
      "CreatorId":
    "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
      "ParentFolderId":
    "1ece93e5fe75315c7407c4967918b4fd9da87ddb2a588e67b7fdaf4a98fde678",
      "CreatedTimestamp": 1529005196.082,
      "ModifiedTimestamp": 1534452483.01,
      "LatestVersionMetadata": {
        "Id":
    "1534452029587-15e129dfc187505c407588df255be83de2920d733859f1d2762411d22a83e3ef",
        "Name": "exampleDoc.docx",
        "ContentType": "application/vnd.openxmlformats-
officedocument.wordprocessingml.document",
        "Size": 13922,
        "Signature": "1a23456b78901c23d4ef56gh7EXAMPLE",
        "Status": "ACTIVE",
        "CreatedTimestamp": 1534452029.587,
        "ModifiedTimestamp": 1534452029.587,
        "CreatorId":
    "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c"
      },
      "ResourceState": "ACTIVE"
    }
  ]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFolderContents](#)中的。

## describe-groups

以下代码示例演示了如何使用 describe-groups。

### AWS CLI

#### 检索群组列表

以下describe-groups示例列出了与指定 Amazon WorkDocs 组织关联的群组。

```
aws workdocs describe-groups \  
  --search-query "e" \  
  --organization-id d-123456789c
```

输出：

```
{  
  "Groups": [  
    {  
      "Id": "S-1-1-11-1122222222-2222233333-3333334444-4444&d-123456789c",  
      "Name": "Example Group 1"  
    },  
    {  
      "Id": "S-1-1-11-1122222222-2222233333-3333334444-5555&d-123456789c",  
      "Name": "Example Group 2"  
    }  
  ]  
}
```

有关更多信息，请参阅《[亚马逊 WorkDocs 管理指南](#)》WorkDocs中的“[亚马逊入门](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeGroups](#)中的。

## describe-notification-subscriptions

以下代码示例演示了如何使用 describe-notification-subscriptions。

### AWS CLI

#### 检索通知订阅列表

以下describe-notification-subscriptions示例检索指定 Amazon WorkDocs 组织的通知订阅。

```
aws workdocs describe-notification-subscriptions \  
  --organization-id d-123456789c
```

输出：

```
{  
  "Subscriptions": [  
    {  
      "SubscriptionId": "123ab4c5-678d-901e-f23g-45h6789j0123",  
      "EndPoint": "https://example.com/example",  
      "Protocol": "HTTPS"  
    }  
  ]  
}
```

有关更多信息，请参阅 [《Amazon WorkDocs 开发者指南》](#) 中的 [订阅通知](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [DescribeNotificationSubscriptions](#) 中的。

## describe-resource-permissions

以下代码示例演示了如何使用 describe-resource-permissions。

### AWS CLI

获取资源的权限列表

此示例返回指定资源（文档或文件夹）的权限列表。

命令：

```
aws workdocs describe-resource-permissions --resource-id  
15df51e0335cfcc6a2e4de9dd8be9f22ee40545ad9176f54758dcf903be982d3
```

输出：

```
{  
  "Principals": [  
    {  
      "Id": "anonymous",  
      "Type": "ANONYMOUS",
```

```
    "Roles": [
      {
        "Role": "VIEWER",
        "Type": "DIRECT"
      }
    ],
  },
  {
    "Id": "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
    "Type": "USER",
    "Roles": [
      {
        "Role": "OWNER",
        "Type": "DIRECT"
      }
    ]
  },
  {
    "Id": "d-926726012c",
    "Type": "ORGANIZATION",
    "Roles": [
      {
        "Role": "VIEWER",
        "Type": "INHERITED"
      }
    ]
  }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeResourcePermissions](#)中的。

## describe-users

以下代码示例演示了如何使用 describe-users。

### AWS CLI

检索指定用户的详细信息

此示例检索指定组织中所有用户的详细信息。

命令:

```
aws workdocs describe-users --organization-id d-926726012c
```

输出：

```
{
  "Users": [
    {
      "Id": "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
      "Username": "example1User",
      "OrganizationId": "d-926726012c",
      "RootFolderId":
"3c0e3f849dd20a9771d937b9bbcc97e18796150ae56c26d64a4fa0320a2dedc9",
      "RecycleBinFolderId":
"c277f4c4d647be1f5147b3184ffa96e1e2bf708278b696cacba68ba13b91f4fe",
      "Status": "INACTIVE",
      "Type": "USER",
      "CreatedTimestamp": 1535478999.452,
      "ModifiedTimestamp": 1535478999.452
    },
    {
      "Id": "S-1-1-11-1111111111-2222222222-3333333333-4444&d-926726012c",
      "Username": "example2User",
      "EmailAddress": "example2User@site.awsapps.com",
      "GivenName": "example2Name",
      "Surname": "example2Surname",
      "OrganizationId": "d-926726012c",
      "RootFolderId":
"35b886cb17198cbd547655e58b025dff0cf34aaed638be52009567e23dc67390",
      "RecycleBinFolderId":
"9858c3e9ed4c2460dde9aadb4c69fde998070dd46e5e985bd08ec6169ea249ff",
      "Status": "ACTIVE",
      "Type": "MINIMALUSER",
      "CreatedTimestamp": 1535478836.584,
      "ModifiedTimestamp": 1535478836.584
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeUsers](#)中的。

## get-document-path

以下代码示例演示了如何使用 `get-document-path`。

### AWS CLI

#### 检索文档的路径信息

此示例检索指定文档的路径信息（根文件夹中的层次结构），并包括父文件夹的名称。

#### 命令：

```
aws workdocs get-document-path --document-id
d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65 --fields NAME
```

#### 输出：

```
{
  "Path": {
    "Components": [
      {
        "Id":
"a43d29cbb8e7c4d25cfee8b803a504b0dc63e760b55ad0c611c6b87691eb6ff3",
        "Name": "/"
      },
      {
        "Id":
"1ece93e5fe75315c7407c4967918b4fd9da87ddb2a588e67b7fdaf4a98fde678",
        "Name": "Top Level Folder"
      },
      {
        "Id":
"d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65",
        "Name": "exampleDoc.docx"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDocumentPath](#)中的。



## get-document-version

以下代码示例演示了如何使用 `get-document-version`。

### AWS CLI

#### 检索指定文档的版本元数据

此示例检索指定文档的版本元数据，包括源 URL 和自定义元数据。

命令：

```
aws workdocs get-document-version --document-id
15df51e0335cfcc6a2e4de9dd8be9f22ee40545ad9176f54758dcf903be982d3 --version-id
1521672507741-9f7df0ea5dd0b121c4f3564a0c7c0b4da95cd12c635d3c442af337a88e297920 --
fields SOURCE --include-custom-metadata
```

输出：

```
{
  "Metadata": {
    "Id":
"1521672507741-9f7df0ea5dd0b121c4f3564a0c7c0b4da95cd12c635d3c442af337a88e297920",
    "Name": "exampleDoc",
    "ContentType": "application/vnd.openxmlformats-
officedocument.wordprocessingml.document",
    "Size": 11537,
    "Signature": "1a23456b78901c23d4ef56gh7EXAMPLE",
    "Status": "ACTIVE",
    "CreatedTimestamp": 1521672507.741,
    "ModifiedTimestamp": 1534451113.504,
    "CreatorId": "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
    "Source": {
      "ORIGINAL": "https://gb-us-west-2-prod-doc-source.s3.us-
west-2.amazonaws.com/15df51e0335cfcc6a2e4de9dd8be9f22ee40545ad9176f54758dcf903be982d3/152167
response-content-disposition=attachment%3B%20filename%2A
%3DUTF-8%27%27exampleDoc&X-Amz-Algorithm=AWS1-ABCD-EFG234&X-Amz-
Date=20180820T212202Z&X-Amz-SignedHeaders=host&X-Amz-Expires=900&X-Amz-
Credential=AKIAIOSFODNN7EXAMPLE%2F20180820%2Fus-west-2%2Fs3%2Faws1_request&X-Amz-
Signature=01Ab2c34d567e8f90123g456hi78j901k2345678l901234mno56pqr78EXAMPLE"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDocumentVersion](#)中的。

## get-document

以下代码示例演示了如何使用 get-document。

### AWS CLI

#### 检索文档详细信息

此示例检索指定文档的详细信息。

命令：

```
aws workdocs get-document --document-id
d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65
```

输出：

```
{
  "Metadata": {
    "Id": "d90d93c1fe44bad0c8471e973ebaab339090401a95e777cffa58e977d2983b65",
    "CreatorId": "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
    "ParentFolderId":
"1ece93e5fe75315c7407c4967918b4fd9da87ddb2a588e67b7fdaf4a98fde678",
    "CreatedTimestamp": 1529005196.082,
    "ModifiedTimestamp": 1534452483.01,
    "LatestVersionMetadata": {
      "Id":
"1534452029587-15e129dfc187505c407588df255be83de2920d733859f1d2762411d22a83e3ef",
      "Name": "exampleDoc.docx",
      "ContentType": "application/vnd.openxmlformats-
officedocument.wordprocessingml.document",
      "Size": 13922,
      "Signature": "1a23456b78901c23d4ef56gh7EXAMPLE",
      "Status": "ACTIVE",
      "CreatedTimestamp": 1534452029.587,
      "ModifiedTimestamp": 1534452029.587,
      "CreatorId": "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c"
    },
    "ResourceState": "ACTIVE"
  }
}
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDocument](#)中的。

## get-folder-path

以下代码示例演示了如何使用 `get-folder-path`。

### AWS CLI

#### 检索文件夹的路径信息

此示例检索指定文件夹的路径信息（根文件夹的层次结构），并包括父文件夹的名称。

命令：

```
aws workdocs get-folder-path --folder-id
50893c0af679524d1a0e0651130ed6d073e1a05f95bd12c42dcde5d35634ed08 --fields NAME
```

输出：

```
{
  "Path": {
    "Components": [
      {
        "Id":
"a43d29cbb8e7c4d25cfee8b803a504b0dc63e760b55ad0c611c6b87691eb6ff3",
        "Name": "/"
      },
      {
        "Id":
"1ece93e5fe75315c7407c4967918b4fd9da87ddb2a588e67b7fdaf4a98fde678",
        "Name": "Top Level Folder"
      },
      {
        "Id":
"50893c0af679524d1a0e0651130ed6d073e1a05f95bd12c42dcde5d35634ed08",
        "Name": "Sublevel Folder"
      }
    ]
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFolderPath](#)中的。

## get-folder

以下代码示例演示了如何使用 `get-folder`。

### AWS CLI

检索文件夹的元数据

此示例检索指定文件夹的元数据。

命令:

```
aws workdocs get-folder --folder-id
50893c0af679524d1a0e0651130ed6d073e1a05f95bd12c42dcde5d35634ed08
```

输出:

```
{
  "Metadata": {
    "Id": "50893c0af679524d1a0e0651130ed6d073e1a05f95bd12c42dcde5d35634ed08",
    "Name": "exampleFolder",
    "CreatorId": "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
    "ParentFolderId":
"1ece93e5fe75315c7407c4967918b4fd9da87ddb2a588e67b7fdaf4a98fde678",
    "CreatedTimestamp": 1534450467.622,
    "ModifiedTimestamp": 1534451113.504,
    "ResourceState": "ACTIVE",
    "Signature": "1a23456b78901c23d4ef56gh7EXAMPLE",
    "Size": 23019,
    "LatestVersionSize": 11537
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetFolder](#)中的。

## get-resources

以下代码示例演示了如何使用 `get-resources`。

## AWS CLI

### 检索共享资源

以下`get-resources`示例检索与指定 Amazon WorkDocs 用户共享的资源。

```
aws workdocs get-resources \  
  --user-id "S-1-1-11-1111111111-2222222222-3333333333-3333" \  
  --collection-type SHARED_WITH_ME
```

输出：

```
{  
  "Folders": [],  
  "Documents": []  
}
```

有关更多信息，请参阅 Amazon WorkDocs 用户指南中的[共享文件和文件夹](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetResources](#)中的。

## initiate-document-version-upload

以下代码示例演示了如何使用 `initiate-document-version-upload`。

### AWS CLI

#### 启动文档版本上传

以下`initiate-document-upload`示例创建了一个新的文档对象和版本对象。

```
aws workdocs initiate-document-version-upload \  
  --name exampledocname \  
  --parent-folder-id  
  eacd546d952531c633452ed67cac23161aa0d5df2e8061223a59e8f67e7b6189
```

输出：

```
{  
  "Metadata": {  
    "Id": "feaba64d4efdf271c2521b60a2a44a8f057e84beaabbe22f01267313209835f2",
```

```

    "CreatorId": "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
    "ParentFolderId":
"eadc546d952531c633452ed67cac23161aa0d5df2e8061223a59e8f67e7b6189",
    "CreatedTimestamp": 1536773972.914,
    "ModifiedTimestamp": 1536773972.914,
    "LatestVersionMetadata": {
      "Id": "1536773972914-
ddb67663e782e7ce8455ebc962217cf9f9e47b5a9a702e5c84dcccc417da9313",
      "Name": "exampledocname",
      "ContentType": "application/octet-stream",
      "Size": 0,
      "Status": "INITIALIZED",
      "CreatedTimestamp": 1536773972.914,
      "ModifiedTimestamp": 1536773972.914,
      "CreatorId": "arn:aws:iam::123456789123:user/EXAMPLE"
    },
    "ResourceState": "ACTIVE"
  },
  "UploadMetadata": {
    "UploadUrl": "https://gb-us-west-2-prod-doc-source.s3.us-
west-2.amazonaws.com/
feaba64d4efdf271c2521b60a2a44a8f057e84beaabbe22f01267313209835f2/1536773972914-
ddb67663e782e7ce8455ebc962217cf9f9e47b5a9a702e5c84dcccc417da9313?X-Amz-
Algorithm=AWS1-ABCD-EFG234&X-Amz-Date=20180912T173932Z&X-Amz-SignedHeaders=content-
type%3Bhost%3Bx-amz-server-side-encryption&X-Amz-Expires=899&X-Amz-
Credential=AKIAIOSFODNN7EXAMPLE%2F20180912%2Fus-west-2%2Fs3%2Faws1_request&X-Amz-
Signature=01Ab2c34d567e8f90123g456hi78j901k23456781901234mno56pqr78EXAMPLE",
    "SignedHeaders": {
      "Content-Type": "application/octet-stream",
      "x-amz-server-side-encryption": "ABC123"
    }
  }
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[InitiateDocumentVersionUpload](#)中的。

## remove-all-resource-permissions

以下代码示例演示了如何使用 `remove-all-resource-permissions`。

### AWS CLI

移除指定资源的所有权限

此示例从指定资源中删除所有权限。

命令:

```
aws workdocs remove-all-resource-permissions --resource-id
1ece93e5fe75315c7407c4967918b4fd9da87ddb2a588e67b7fdaf4a98fde678
```

输出 :

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveAllResourcePermissions](#)中的。

### **remove-resource-permission**

以下代码示例演示了如何使用 `remove-resource-permission`。

#### AWS CLI

从资源中移除权限

此示例从资源中删除指定委托人的权限。

命令:

```
aws workdocs remove-resource-permission --resource-id
1ece93e5fe75315c7407c4967918b4fd9da87ddb2a588e67b7fdaf4a98fde678 --principal-id
anonymous
```

输出 :

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RemoveResourcePermission](#)中的。

### **update-document-version**

以下代码示例演示了如何使用 `update-document-version`。

## AWS CLI

将文档版本状态更改为“活动”

此示例将文档版本的状态更改为“活动”。

命令:

```
aws workdocs update-document-version --document-id
15df51e0335cfcc6a2e4de9dd8be9f22ee40545ad9176f54758dcf903be982d3 --version-id
1521672507741-9f7df0ea5dd0b121c4f3564a0c7c0b4da95cd12c635d3c442af337a88e297920 --
version-status ACTIVE
```

输出 :

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDocumentVersion](#)中的。

## update-document

以下代码示例演示了如何使用 update-document。

## AWS CLI

更新文档

此示例更新文档的名称和父文件夹。

命令:

```
aws workdocs update-document --document-id
15df51e0335cfcc6a2e4de9dd8be9f22ee40545ad9176f54758dcf903be982d3 --name updatedDoc
--parent-folder-id 50893c0af679524d1a0e0651130ed6d073e1a05f95bd12c42dcde5d35634ed08
```

输出 :

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateDocument](#)中的。



## update-folder

以下代码示例演示了如何使用 update-folder。

### AWS CLI

#### 更新文件夹

此示例更新了文件夹的名称和父文件夹。

#### 命令:

```
aws workdocs update-folder --folder-id
50893c0af679524d1a0e0651130ed6d073e1a05f95bd12c42dcde5d35634ed08 --name
exampleFolder1 --parent-folder-id
1ece93e5fe75315c7407c4967918b4fd9da87ddb2a588e67b7fdaf4a98fde678
```

#### 输出 :

```
None
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateFolder](#)中的。

## update-user

以下代码示例演示了如何使用 update-user。

### AWS CLI

#### 更新用户

此示例更新了指定用户的时区。

#### 命令:

```
aws workdocs update-user --user-id
"S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c" --time-zone-id
"America/Los_Angeles"
```

#### 输出 :

```
{
  "User": {
    "Id": "S-1-1-11-1111111111-2222222222-3333333333-3333&d-926726012c",
    "Username": "exampleUser",
    "EmailAddress": "exampleUser@site.awsapps.com",
    "GivenName": "Example",
    "Surname": "User",
    "OrganizationId": "d-926726012c",
    "RootFolderId":
    "c5eceb5e1a2d1d460c9d1af8330ae117fc8d39bb1d3ed6acd0992d5ff192d986",
    "RecycleBinFolderId":
    "6ca20102926ad15f04b1d248d6d6e44f2449944eda5c758f9a1e9df6a6b7fa66",
    "Status": "ACTIVE",
    "Type": "USER",
    "TimeZoneId": "America/Los_Angeles",
    "Storage": {
      "StorageUtilizedInBytes": 0,
      "StorageRule": {
        "StorageAllocatedInBytes": 53687091200,
        "StorageType": "QUOTA"
      }
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateUser](#)中的。

## 使用亚马逊的 WorkMail 示例 AWS CLI

以下代码示例向您展示如何在 Amazon 中使用来执行操作和实现常见场景 WorkMail。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### **associate-delegate-to-resource**

以下代码示例演示了如何使用 `associate-delegate-to-resource`。

#### AWS CLI

向资源添加委托人

以下`associate-delegate-to-resource`命令为资源添加委托。

```
aws workmail associate-delegate-to-resource \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --resource-id r-68bf2d3b1c0244aab7264c24b9217443 \  
  --entity-id S-1-1-11-1111111111-2222222222-3333333333-3333
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateDelegateToResource](#)中的。

### **associate-member-to-group**

以下代码示例演示了如何使用 `associate-member-to-group`。

#### AWS CLI

向群组中添加成员

以下`associate-member-to-group`命令将指定的成员添加到群组中。

```
aws workmail associate-member-to-group \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --group-id S-1-1-11-1122222222-2222233333-3333334444-4444 \  
  --member-id S-1-1-11-1111111111-2222222222-3333333333-3333
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssociateMemberToGroup](#)中的。

### **create-alias**

以下代码示例演示了如何使用 `create-alias`。

## AWS CLI

### 创建别名

以下`create-alias`命令为指定实体（用户或组）创建别名。

```
aws workmail create-alias \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --entity-id S-1-1-11-1122222222-2222233333-3333334444-4444 \  
  --alias exampleAlias@site.awsapps.com
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAlias](#)中的。

### **create-group**

以下代码示例演示了如何使用 `create-group`。

## AWS CLI

### 创建新群组

以下`create-group`命令为指定组织创建新群组。

```
aws workmail create-group \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --name exampleGroup1
```

输出：

```
{  
  "GroupId": "S-1-1-11-1122222222-2222233333-3333334444-4444"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateGroup](#)中的。

### **create-resource**

以下代码示例演示了如何使用 `create-resource`。

## AWS CLI

### 创建新资源

以下`create-resource`命令为指定组织创建新资源（会议室）。

```
aws workmail create-resource \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --name exampleRoom1 \  
  --type ROOM
```

输出：

```
{  
  "ResourceId": "r-7afe0efbade843a58cdc10251fce992c"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateResource](#)中的。

## create-user

以下代码示例演示了如何使用 `create-user`。

## AWS CLI

### 创建新用户

以下`create-user`命令创建一个新用户。

```
aws workmail create-user \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --name exampleName \  
  --display-name exampleDisplayName \  
  --password examplePa$$w0rd
```

输出：

```
{  
  "UserId": "S-1-1-11-1111111111-2222222222-3333333333-3333"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUser](#)中的。

## delete-access-control-rule

以下代码示例演示了如何使用 delete-access-control-rule。

### AWS CLI

#### 删除访问控制规则

以下delete-access-control-rule示例从指定的 Amazon WorkMail 组织中删除指定的访问控制规则。

```
aws workmail delete-access-control-rule \  
  --organization-id m-n1pq2345678r901st2u3vx45x6789yza \  
  --name "myRule"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon WorkMail 管理员指南》中的[使用访问控制规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAccessControlRule](#)中的。

## delete-alias

以下代码示例演示了如何使用 delete-alias。

### AWS CLI

#### 删除别名

以下delete-alias命令删除指定实体（用户或组）的别名。

```
aws workmail delete-alias \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --entity-id S-1-1-11-1122222222-2222233333-3333334444-4444 \  
  --alias exampleAlias@site.awsapps.com
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAlias](#)中的。

## delete-group

以下代码示例演示了如何使用 delete-group。

### AWS CLI

#### 删除现有群组

以下delete-group命令将从 Amazon 中删除现有群组 WorkMail。

```
aws workmail delete-group \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --group-id S-1-1-11-1122222222-2222233333-3333334444-4444
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteGroup](#)中的。

## delete-mailbox-permissions

以下代码示例演示了如何使用 delete-mailbox-permissions。

### AWS CLI

#### 删除邮箱权限

以下delete-mailbox-permissions命令删除先前授予用户或组的邮箱权限。实体代表拥有邮箱的用户，被授权者代表要删除其权限的用户或组。

```
aws workmail delete-mailbox-permissions \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --entity-id S-1-1-11-1122222222-2222233333-3333334444-4444 \  
  --grantee-id S-1-1-11-1111111111-2222222222-3333333333-3333
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteMailboxPermissions](#)中的。

## delete-resource

以下代码示例演示了如何使用 delete-resource。

## AWS CLI

### 删除现有资源

以下delete-resource命令将从 Amazon 中删除现有资源 WorkMail。

```
aws workmail delete-resource \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --resource-id r-7afe0efbade843a58cdc10251fce992c
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteResource](#)中的。

## delete-user

以下代码示例演示了如何使用 delete-user。

### AWS CLI

#### 删除用户

以下delete-user命令将指定用户从 Amazon WorkMail 和所有后续系统中删除。

```
aws workmail delete-user \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --user-id S-1-1-11-1111111111-2222222222-3333333333-3333
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUser](#)中的。

## deregister-from-work-mail

以下代码示例演示了如何使用 deregister-from-work-mail。

### AWS CLI

#### 禁用现有实体

以下deregister-from-work-mail命令禁止现有实体（用户、群组或资源）使用 Amazon WorkMail。



```
aws workmail deregister-from-work-mail \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --entity-id S-1-1-11-1111111111-2222222222-3333333333-3333
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterFromWorkMail](#)中的。

## describe-group

以下代码示例演示了如何使用 describe-group。

### AWS CLI

检索群组的信息

以下describe-group命令检索有关指定组的信息。

```
aws workmail describe-group \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --group-id S-1-1-11-1122222222-2222233333-3333334444-4444
```

输出：

```
{  
  "GroupId": "S-1-1-11-1122222222-2222233333-3333334444-4444",  
  "Name": "exampleGroup1",  
  "State": "ENABLED"  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeGroup](#)中的。

## describe-organization

以下代码示例演示了如何使用 describe-organization。

### AWS CLI

检索组织的信息

以下describe-organization命令检索指定 Amazon WorkMail 组织的信息。

```
aws workmail describe-organization \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27
```

输出：

```
{  
  "OrganizationId": "m-d281d0a2fd824be5b6cd3d3ce909fd27",  
  "Alias": "alias",  
  "State": "Active",  
  "DirectoryId": "d-926726012c",  
  "DirectoryType": "VpcDirectory",  
  "DefaultMailDomain": "site.awsapps.com",  
  "CompletedDate": 1522693605.468,  
  "ARN": "arn:aws:workmail:us-west-2:111122223333:organization/m-  
n1pq2345678r901st2u3vx45x6789yza"  
}
```

有关更多信息，请参阅《Amazon WorkMail 管理员指南》中的“[与组织合作](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeOrganization](#)中的。

## describe-resource

以下代码示例演示了如何使用 describe-resource。

### AWS CLI

检索资源信息

以下describe-resource命令检索有关指定资源的信息。

```
aws workmail describe-resource \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --resource-id r-7afe0efbade843a58cdc10251fce992c
```

输出：

```
{  
  "ResourceId": "r-7afe0efbade843a58cdc10251fce992c",  
  "Name": "exampleRoom1",  
  "Type": "ROOM",
```

```
"BookingOptions": {
  "AutoAcceptRequests": true,
  "AutoDeclineRecurringRequests": false,
  "AutoDeclineConflictingRequests": true
},
"State": "ENABLED"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeResource](#)中的。

## describe-user

以下代码示例演示了如何使用 describe-user。

### AWS CLI

#### 检索用户信息

以下describe-user命令检索有关指定用户的信息。

```
aws workmail describe-user \
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \
  --user-id S-1-1-11-1111111111-2222222222-3333333333-3333
```

输出：

```
{
  "UserId": "S-1-1-11-1111111111-2222222222-3333333333-3333",
  "Name": "exampleUser1",
  "Email": "exampleUser1@site.awsapps.com",
  "DisplayName": "",
  "State": "ENABLED",
  "UserRole": "USER",
  "EnabledDate": 1532459261.827
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeUser](#)中的。

## disassociate-delegate-from-resource

以下代码示例演示了如何使用 disassociate-delegate-from-resource。

## AWS CLI

### 从资源中移除成员

以下`disassociate-delegate-from-resource`命令从资源中移除指定成员。

```
aws workmail disassociate-delegate-from-resource \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --resource-id r-68bf2d3b1c0244aab7264c24b9217443 \  
  --entity-id S-1-1-11-1111111111-2222222222-3333333333-3333
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateDelegateFromResource](#)中的。

## **disassociate-member-from-group**

以下代码示例演示了如何使用 `disassociate-member-from-group`。

## AWS CLI

### 从群组中移除成员

以下`disassociate-member-from-group`命令将指定成员从群组中移除。

```
aws workmail disassociate-member-from-group \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --group-id S-1-1-11-1122222222-2222233333-3333334444-4444 \  
  --member-id S-1-1-11-1111111111-2222222222-3333333333-3333
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DisassociateMemberFromGroup](#)中的。

## **get-access-control-effect**

以下代码示例演示了如何使用 `get-access-control-effect`。

## AWS CLI

### 要获得访问控制规则的效果

以下 `get-access-control-effect` 示例检索指定 Amazon WorkMail 组织对指定 IP 地址、访问协议操作和用户 ID 的访问控制规则的影响。

```
aws workmail get-access-control-effect \  
  --organization-id m-n1pq2345678r901st2u3vx45x6789yza \  
  --ip-address "192.0.2.0" \  
  --action "WindowsOutlook" \  
  --user-id "S-1-1-11-1111111111-2222222222-3333333333-3333"
```

输出：

```
{  
  "Effect": "DENY",  
  "MatchedRules": [  
    "myRule"  
  ]  
}
```

有关更多信息，请参阅《Amazon WorkMail 管理员指南》中的[使用访问控制规则](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetAccessControlEffect](#)中的。

## get-mailbox-details

以下代码示例演示了如何使用 `get-mailbox-details`。

### AWS CLI

获取用户的邮箱详细信息

以下 `get-mailbox-details` 命令检索有关指定用户邮箱的详细信息。

```
aws workmail get-mailbox-details \  
  --organization-id m-n1pq2345678r901st2u3vx45x6789yza \  
  --user-id S-1-1-11-1111111111-2222222222-3333333333-3333
```

输出：

```
{  
  "MailboxQuota": 51200,  
  "MailboxSize": 0.03890800476074219
```

```
}
```

有关更多信息，请参阅《Amazon WorkMail 管理员指南》中的[管理用户账户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetMailboxDetails](#)中的。

## list-access-control-rules

以下代码示例演示了如何使用 `list-access-control-rules`。

### AWS CLI

列出访问控制规则

以下`list-access-control-rules`示例列出了指定 Amazon WorkMail 组织的访问控制规则。

```
aws workmail list-access-control-rules \  
  --organization-id m-n1pq2345678r901st2u3vx45x6789yza
```

输出：

```
{  
  "Rules": [  
    {  
      "Name": "default",  
      "Effect": "ALLOW",  
      "Description": "Default WorkMail Rule",  
      "DateCreated": 0.0,  
      "DateModified": 0.0  
    },  
    {  
      "Name": "myRule",  
      "Effect": "DENY",  
      "Description": "my rule",  
      "UserIds": [  
        "S-1-1-11-1111111111-2222222222-3333333333-3333"  
      ],  
      "DateCreated": 1581635628.0,  
      "DateModified": 1581635628.0  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon WorkMail 管理员指南》中的[使用访问控制规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAccessControlRules](#)中的。

## list-aliases

以下代码示例演示了如何使用 list-aliases。

### AWS CLI

列出成员的别名

以下list-aliases命令列出指定成员（用户或组）的别名。

```
aws workmail list-aliases \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --entity-id S-1-1-11-1111111111-2222222222-3333333333-3333
```

输出：

```
{  
  "Aliases": [  
    "exampleAlias@site.awsapps.com",  
    "exampleAlias1@site.awsapps.com"  
  ]  
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAliases](#)中的。

## list-group-members

以下代码示例演示了如何使用 list-group-members。

### AWS CLI

列出群组成员

以下list-group-members命令列出了指定组的成员。

```
aws workmail list-group-members \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --group-id S-1-1-11-1111111111-2222222222-3333333333-3333
```

```
--group-id S-1-1-11-1122222222-2222233333-3333334444-4444
```

输出：

```
{
  "Members": [
    {
      "Id": "S-1-1-11-1111111111-2222222222-3333333333-3333",
      "Name": "exampleUser1",
      "Type": "USER",
      "State": "ENABLED",
      "EnabledDate": 1532459261.827
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListGroupMembers](#)中的。

## list-groups

以下代码示例演示了如何使用 list-groups。

### AWS CLI

#### 检索群组列表

以下list-groups命令检索指定组织中群组的摘要。

```
aws workmail list-groups \
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27
```

输出：

```
{
  "Groups": [
    {
      "Id": "S-1-1-11-1122222222-2222233333-3333334444-4444",
      "Name": "exampleGroup1",
      "State": "DISABLED"
    },
    {
      "Id": "S-4-4-44-1122222222-2222233333-3333334444-4444",
```



```
        "Name": "exampleGroup2",
        "State": "ENABLED"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListGroups](#)中的。

## list-mailbox-permissions

以下代码示例演示了如何使用 list-mailbox-permissions。

### AWS CLI

#### 检索邮箱权限

以下list-mailbox-permissions命令检索与指定实体的邮箱关联的邮箱权限。

```
aws workmail list-mailbox-permissions \
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \
  --entity-id S-1-1-11-1111111111-2222222222-3333333333-3333
```

输出：

```
{
  "Permissions": [
    {
      "GranteeId": "S-1-1-11-1122222222-2222233333-3333334444-4444",
      "GranteeType": "USER",
      "PermissionValues": [
        "FULL_ACCESS"
      ]
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListMailboxPermissions](#)中的。

## list-organizations

以下代码示例演示了如何使用 list-organizations。

## AWS CLI

### 检索组织列表

以下`list-organizations`命令检索未删除的组织的摘要。

```
aws workmail list-organizations
```

输出：

```
{
  "OrganizationSummaries": [
    {
      "OrganizationId": "m-d281d0a2fd824be5b6cd3d3ce909fd27",
      "Alias": "exampleAlias",
      "State": "Active"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListOrganizations](#)中的。

## list-resource-delegates

以下代码示例演示了如何使用 `list-resource-delegates`。

### AWS CLI

#### 列出资源的委托人

以下`list-resource-delegates`命令检索与指定资源关联的委托。

```
aws workmail list-resource-delegates \
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \
  --resource-id r-68bf2d3b1c0244aab7264c24b9217443
```

输出：

```
{
  "Delegates": [
    {
      "Id": "S-1-1-11-1111111111-2222222222-3333333333-3333",

```

```
        "Type": "USER"
      }
    ]
  }
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResourceDelegates](#)中的。

## list-resources

以下代码示例演示了如何使用 `list-resources`。

### AWS CLI

#### 检索资源列表

以下`list-resources`命令检索指定组织的资源摘要。

```
aws workmail list-resources \
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27
```

输出：

```
{
  "Resources": [
    {
      "Id": "r-7afe0efbade843a58cdc10251fce992c",
      "Name": "exampleRoom1",
      "Type": "ROOM",
      "State": "ENABLED"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListResources](#)中的。

## list-tags-for-resource

以下代码示例演示了如何使用 `list-tags-for-resource`。

### AWS CLI

#### 列出资源的标签

以下 `list-tags-for-resource` 示例列出了指定 Amazon WorkMail 组织的标签。

```
aws workmail list-tags-for-resource \  
  --resource-arn arn:aws:workmail:us-west-2:111122223333:organization/m-  
n1pq2345678r901st2u3vx45x6789yza
```

输出：

```
{  
  "Tags": [  
    {  
      "Key": "priority",  
      "Value": "1"  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon WorkMail 管理员指南》中的为[组织添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTagsForResource](#)中的。

## list-users

以下代码示例演示了如何使用 `list-users`。

### AWS CLI

检索用户列表

以下 `list-users` 命令检索指定组织中用户的摘要。

```
aws workmail list-users \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27
```

输出：

```
{  
  "Users": [  
    {  
      "Id": "S-1-1-11-1111111111-2222222222-3333333333-3333",  
      "Email": "exampleUser1@site.awsapps.com",
```

```

        "Name": "exampleUser1",
        "State": "ENABLED",
        "UserRole": "USER",
        "EnabledDate": 1532459261.827
    },
    {
        "Id": "S-1-1-11-1122222222-2222233333-3333334444-4444",
        "Name": "exampleGuestUser",
        "State": "DISABLED",
        "UserRole": "SYSTEM_USER"
    }
]
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUsers](#)中的。

## put-access-control-rule

以下代码示例演示了如何使用 `put-access-control-rule`。

### AWS CLI

#### 放置新的访问控制规则

以下`put-access-control-rule`示例拒绝指定用户访问指定的 Amazon WorkMail 组织。

```

aws workmail put-access-control-rule \
  --name "myRule" \
  --effect "DENY" \
  --description "my rule" \
  --user-ids "S-1-1-11-1111111111-2222222222-3333333333-3333" \
  --organization-id m-n1pq2345678r901st2u3vx45x6789yza

```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon WorkMail 管理员指南》中的[使用访问控制规则](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutAccessControlRule](#)中的。

## put-mailbox-permissions

以下代码示例演示了如何使用 `put-mailbox-permissions`。

## AWS CLI

### 设置邮箱权限

以下`put-mailbox-permissions`命令为指定的被授权者（用户或组）设置完全访问权限。该实体代表邮箱的所有者。

```
aws workmail put-mailbox-permissions \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --entity-id S-1-1-11-1111111111-2222222222-3333333333-3333 \  
  --grantee-id S-1-1-11-1122222222-2222233333-3333334444-4444 \  
  --permission-values FULL_ACCESS
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutMailboxPermissions](#)中的。

## register-to-work-mail

以下代码示例演示了如何使用 `register-to-work-mail`。

## AWS CLI

### 注册现有或已禁用的实体

以下`register-to-work-mail`命令允许指定的现有实体（用户、群组或资源）使用 Amazon WorkMail。

```
aws workmail register-to-work-mail \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --entity-id S-1-1-11-1122222222-2222233333-3333334444-4444 \  
  --email exampleGroup1@site.awsapps.com
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterToWorkMail](#)中的。

## reset-password

以下代码示例演示了如何使用 `reset-password`。

## AWS CLI

### 重置用户的密码

以下 `reset-password` 命令重置指定用户的密码。

```
aws workmail reset-password \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --user-id S-1-1-11-1111111111-2222222222-3333333333-3333 \  
  --password examplePa$$w0rd
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ResetPassword](#) 中的。

## tag-resource

以下代码示例演示了如何使用 `tag-resource`。

## AWS CLI

### 对资源应用标签

以下 `tag-resource` 示例将密钥为 “priority” 且值为 “1” 的标签应用于指定的 Amazon WorkMail 组织。

```
aws workmail tag-resource \  
  --resource-arn arn:aws:workmail:us-west-2:111122223333:organization/m-  
n1pq2345678r901st2u3vx45x6789yza \  
  --tags "Key=priority,Value=1"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon WorkMail 管理员指南》中的为 [组织添加标签](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TagResource](#) 中的。

## untag-resource

以下代码示例演示了如何使用 `untag-resource`。

## AWS CLI

### 取消对资源的标记

以下 `untag-resource` 示例从指定的 Amazon WorkMail 组织中移除指定标签。

```
aws workmail untag-resource \  
  --resource-arn arn:aws:workmail:us-west-2:111122223333:organization/m-  
n1pq2345678r901st2u3vx45x6789yza \  
  --tag-keys "priority"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon WorkMail 管理员指南》中的为[组织添加标签](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UntagResource](#)中的。

## update-mailbox-quota

以下代码示例演示了如何使用 `update-mailbox-quota`。

### AWS CLI

#### 更新用户的邮箱配额

以下 `update-mailbox-quota` 命令更改指定用户的邮箱配额。

```
aws workmail update-mailbox-quota \  
  --organization-id m-n1pq2345678r901st2u3vx45x6789yza \  
  --user-id S-1-1-11-1111111111-2222222222-3333333333-3333 \  
  --mailbox-quota 40000
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon WorkMail 管理员指南》中的[管理用户账户](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateMailboxQuota](#)中的。

## update-primary-email-address

以下代码示例演示了如何使用 `update-primary-email-address`。



## AWS CLI

### 更新主电子邮件地址

以下 `update-primary-email-address` 命令更新指定实体（用户、组或资源）的主电子邮件地址。

```
aws workmail update-primary-email-address \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --entity-id S-1-1-11-1111111111-2222222222-3333333333-3333 \  
  --email exampleUser2@site.awsapps.com
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdatePrimaryEmailAddress](#) 中的。

## update-resource

以下代码示例演示了如何使用 `update-resource`。

## AWS CLI

### 更新资源

以下 `update-resource` 命令更新指定资源的名称。

```
aws workmail update-resource \  
  --organization-id m-d281d0a2fd824be5b6cd3d3ce909fd27 \  
  --resource-id r-7afe0efbade843a58cdc10251fce992c \  
  --name exampleRoom2
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateResource](#) 中的。

## 使用的 Amazon WorkMail 消息流示例 AWS CLI

以下代码示例向您展示了如何使用 `with Amazon Mess WorkMail age Flow` 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

## get-raw-message-content

以下代码示例演示了如何使用 `get-raw-message-content`。

AWS CLI

获取电子邮件的原始内容

以下 `get-raw-message-content` 示例获取传输中的电子邮件的原始内容并将其发送到名为 `test` 的文本文件。

```
aws workmailmessageflow get-raw-message-content \  
  --message-id a1b2cd34-ef5g-6h7j-k18m-npq9012345rs \  
  test
```

命令运行 `test` 后的文件内容：

```
Subject: Hello World  
From: =?UTF-8?Q?marymajor_marymajor?= <marymajor@example.com>  
To: =?UTF-8?Q?mateojackson=40example=2Enet?= <mateojackson@example.net>  
Date: Thu, 7 Nov 2019 19:22:46 +0000  
Mime-Version: 1.0  
Content-Type: multipart/alternative;  
  boundary="=_EXAMPLE+"  
References: <mail.1ab23c45.5de6.7f890g123hj45678@storage.wm.amazon.com>  
X-Priority: 3 (Normal)  
X-Mailer: Amazon WorkMail  
Thread-Index: EXAMPLE  
Thread-Topic: Hello World  
Message-Id: <mail.1ab23c45.5de6.7f890g123hj45678@storage.wm.amazon.com>
```

```
This is a multi-part message in MIME format. Your mail reader does not
understand MIME message format.
--=_EXAMPLE+
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 7bit

hello world

--=_EXAMPLE+
Content-Type: text/html; charset=utf-8
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE HTML><html>
<head>
<meta name=3D"Generator" content=3D"Amazon WorkMail v3.0-4510">
<meta http-equiv=3D"Content-Type" content=3D"text/html; charset=3Dutf-8">=

<title>testing</title>
</head>
<body>
<p style=3D"margin: 0px; font-family: Arial, Tahoma, Helvetica, sans-seri=
f; font-size: small;">hello world</p>
</body>
</html>
--=_EXAMPLE+--
```

有关更多信息，请参阅《WorkMail 亚马逊管理员指南》中的[使用 Lamb AWS da 检索消息内容](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetRawMessageContent](#)中的。

## WorkSpaces 使用示例 AWS CLI

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 WorkSpaces。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

## 主题

- [操作](#)

## 操作

### create-tags

以下代码示例演示了如何使用 create-tags。

#### AWS CLI

向 a 添加标签 Workspace

以下 create-tags 示例将指定的标签添加到指定的 Workspace。

```
aws workspaces create-tags \  
  --resource-id ws-dk1x zr417 \  
  --tags Key=Department,Value=Finance
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》中的[标签 WorkSpaces 资源](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [CreateTags](#) 中的。

### create-workspaces

以下代码示例演示了如何使用 create-workspaces。

#### AWS CLI

示例 1：创建 AlwaysOn Workspace

以下 create-workspaces 示例使用指定的目录和捆绑包 AlwaysOn Workspace 为指定用户创建一个。

```
aws workspaces create-workspaces \  
  --workspaces DirectoryId=d-926722edaf,UserName=Mateo,BundleId=wsb-0zsvgp8fc
```

输出：

```
{
  "FailedRequests": [],
  "PendingRequests": [
    {
      "WorkspaceId": "ws-kcqms853t",
      "DirectoryId": "d-926722edaf",
      "UserName": "Mateo",
      "State": "PENDING",
      "BundleId": "wsb-0zsvgp8fc"
    }
  ]
}
```

### 示例 2：创建 AutoStop WorkSpace

以下create-workspaces示例使用指定的目录和捆绑包 AutoStop WorkSpace 为指定用户创建一个。

```
aws workspaces create-workspaces \
  --workspaces
  DirectoryId=d-926722edaf,UserName=Mary,BundleId=wsb-0zsvgp8fc,WorkspaceProperties={RunningM
```

输出：

```
{
  "FailedRequests": [],
  "PendingRequests": [
    {
      "WorkspaceId": "ws-dk1x zr417",
      "DirectoryId": "d-926722edaf",
      "UserName": "Mary",
      "State": "PENDING",
      "BundleId": "wsb-0zsvgp8fc"
    }
  ]
}
```

### 示例 3：创建用户解耦对象 WorkSpace

以下create-workspaces示例 WorkSpace 通过将用户名设置为并指定 WorkSpace 名称 [UNDEFINED]、目录 ID 和捆绑包 ID 来创建用户解耦对象。

```
aws workspaces create-workspaces \  
  --workspaces \  
  DirectoryId=d-926722edaf,UserName='"[UNDEFINED]"',WorkspaceName=MaryWorkspace1,BundleId=wsb
```

输出：

```
{  
  "FailedRequests": [],  
  "PendingRequests": [  
    {  
      "WorkspaceId": "ws-abcd1234",  
      "DirectoryId": "d-926722edaf",  
      "UserName": "[UNDEFINED]",  
      "State": "PENDING",  
      "BundleId": "wsb-0zsvgp8fc",  
      "WorkspaceName": "MaryWorkspace1"  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》中的[启动虚拟桌面](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateWorkspaces](#)中的。

## delete-tags

以下代码示例演示了如何使用 delete-tags。

### AWS CLI

从中删除标签 WorkSpace

以下delete-tags示例从指定标签中删除指定的标签 WorkSpace。

```
aws workspaces delete-tags \  
  --resource-id ws-dk1x zr417 \  
  --tag-keys Department
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》中的[标签 WorkSpaces 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTags](#)中的。

## deregister-workspace-directory

以下代码示例演示了如何使用 `deregister-workspace-directory`。

### AWS CLI

取消注册目录

以下`deregister-workspace-directory`示例取消注册指定目录。

```
aws workspaces deregister-workspace-directory \  
  --directory-id d-926722edaf
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》WorkSpaces中的向[注册目录](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeregisterWorkspaceDirectory](#)中的。

## describe-tags

以下代码示例演示了如何使用 `describe-tags`。

### AWS CLI

描述 a 的标签 Workspace

以下`describe-tags`示例描述了指定的标签 Workspace。

```
aws workspaces describe-tags \  
  --resource-id ws-dk1xzr417
```

输出：

```
{  
  "TagList": [  
    {  
      "Key": "Department",  
      "Value": "Finance"    }  
  ]  
}
```

```
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》中的[标签 WorkSpaces 资源](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTags](#)中的。

## describe-workspace-bundles

以下代码示例演示了如何使用 describe-workspace-bundles。

### AWS CLI

列出 Amazon 提供的捆绑包

以下describe-workspace-bundles示例列出了由 Amazon 提供的捆绑包的名称和 ID，采用表格格式并按名称排序。

```
aws workspaces describe-workspace-bundles \  
  --owner AMAZON \  
  --query "Bundles[*].[Name, BundleId]"
```

输出：

```
[  
  [  
    "Standard with Amazon Linux 2",  
    "wsb-clj85qzj1"  
  ],  
  [  
    "Performance with Windows 10 (Server 2016 based)",  
    "wsb-gm4d5tx2v"  
  ],  
  [  
    "PowerPro with Windows 7",  
    "wsb-1pzkp0bx4"  
  ],  
  [  
    "Power with Amazon Linux 2",  
    "wsb-2bs6k5lgn"  
  ],  
]
```



```
[
  "Graphics with Windows 10 (Server 2019 based)",
  "wsb-03gyjnfyy"
],
...
]
```

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》中的[WorkSpaces 捆绑包和图片](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeWorkspaceBundles](#)中的。

## describe-workspace-directories

以下代码示例演示了如何使用 describe-workspace-directories。

### AWS CLI

描述已注册的目录

以下describe-workspace-directories示例描述了指定的注册目录。

```
aws workspaces describe-workspace-directories \
  --directory-ids d-926722edaf
```

输出：

```
{
  "Directories": [
    {
      "DirectoryId": "d-926722edaf",
      "Alias": "d-926722edaf",
      "DirectoryName": "example.com",
      "RegistrationCode": "WSpdx+9RJ8JT",
      "SubnetIds": [
        "subnet-9d19c4c6",
        "subnet-500d5819"
      ],
      "DnsIpAddresses": [
        "172.16.1.140",
        "172.16.0.30"
      ],
      "CustomerUserName": "Administrator",
    }
  ]
}
```

```

    "IamRoleId": "arn:aws:iam::123456789012:role/workspaces_DefaultRole",
    "DirectoryType": "SIMPLE_AD",
    "WorkspaceSecurityGroupId": "sg-0d89e927e5645d7c5",
    "State": "REGISTERED",
    "WorkspaceCreationProperties": {
      "EnableWorkDocs": false,
      "EnableInternetAccess": false,
      "UserEnabledAsLocalAdministrator": true,
      "EnableMaintenanceMode": true
    },
    "WorkspaceAccessProperties": {
      "DeviceTypeWindows": "ALLOW",
      "DeviceTypeOsx": "ALLOW",
      "DeviceTypeWeb": "DENY",
      "DeviceTypeIos": "ALLOW",
      "DeviceTypeAndroid": "ALLOW",
      "DeviceTypeChromeOs": "ALLOW",
      "DeviceTypeZeroClient": "ALLOW",
      "DeviceTypeLinux": "DENY"
    },
    "Tenancy": "SHARED",
    "SelfservicePermissions": {
      "RestartWorkspace": "ENABLED",
      "IncreaseVolumeSize": "DISABLED",
      "ChangeComputeType": "DISABLED",
      "SwitchRunningMode": "DISABLED",
      "RebuildWorkspace": "DISABLED"
    }
  }
]
}

```

有关更多信息，请参阅《Amazon 管理指南》WorkSpaces中的 WorkSpaces 管理[目录](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeWorkspaceDirectories](#)中的。

## describe-workspaces-connection-status

以下代码示例演示了如何使用 describe-workspaces-connection-status。

### AWS CLI

描述的连接状态 Workspace

以下describe-workspaces-connection-status示例描述了指定的连接状态Workspace。

```
aws workspaces describe-workspaces-connection-status \  
  --workspace-ids ws-dk1x zr417
```

输出：

```
{  
  "WorkspacesConnectionStatus": [  
    {  
      "WorkspaceId": "ws-dk1x zr417",  
      "ConnectionState": "CONNECTED",  
      "ConnectionStateCheckTimestamp": 1662526214.744  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon 管理指南》WorkSpaces中的 WorkSpaces 管理您的。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeWorkspacesConnectionStatus](#)中的。

## describe-workspaces

以下代码示例演示了如何使用 describe-workspaces。

AWS CLI

描述一个 Workspace

以下describe-workspaces示例描述了指定的 Workspace。

```
aws workspaces describe-workspaces \  
  --workspace-ids ws-dk1x zr417
```

输出：

```
{  
  "Workspaces": [  
    {  
      "WorkspaceId": "ws-dk1x zr417",
```

```

    "DirectoryId": "d-926722edaf",
    "UserName": "Mary",
    "IpAddress": "172.16.0.175",
    "State": "STOPPED",
    "BundleId": "wsb-0zsvgp8fc",
    "SubnetId": "subnet-500d5819",
    "ComputerName": "WSAMZN-RBSLTDD9",
    "WorkspaceProperties": {
      "RunningMode": "AUTO_STOP",
      "RunningModeAutoStopTimeoutInMinutes": 60,
      "RootVolumeSizeGib": 80,
      "UserVolumeSizeGib": 10,
      "ComputeTypeName": "VALUE"
    },
    "ModificationStates": []
  }
]
}

```

有关更多信息，请参阅《Amazon 管理指南》WorkSpaces中的 WorkSpaces 管理[您的](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeWorkspaces](#)中的。

## migrate-workspace

以下代码示例演示了如何使用 migrate-workspace。

### AWS CLI

#### 要迁移 Workspace

以下migrate-workspace示例将指定的迁移 Workspace 到指定的捆绑包。

```

aws workspaces migrate-workspace \
  --source-workspace-id ws-dk1x zr417 \
  --bundle-id wsb-j4d ky1gs4

```

#### 输出：

```

{
  "SourceWorkspaceId": "ws-dk1x zr417",
  "TargetWorkspaceId": "ws-x5h11b kp5"
}

```

```
}
```

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》Workspace 中的 [Migrate a](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [MigrateWorkspace](#) 中的。

## modify-workspace-creation-properties

以下代码示例演示了如何使用 `modify-workspace-creation-properties`。

### AWS CLI

修改目录的 Workspace 创建属性

以下 `modify-workspace-creation-properties` 示例为指定目录启用该 `EnableInternetAccess` 属性。这允许为目录 WorkSpaces 创建的自动分配公有 IP 地址。

```
aws workspaces modify-workspace-creation-properties \  
  --resource-id d-926722edaf \  
  --workspace-creation-properties EnableInternetAccess=true
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》WorkSpaces 中的“[更新您的目录详情](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [ModifyWorkspaceCreationProperties](#) 中的。

## modify-workspace-properties

以下代码示例演示了如何使用 `modify-workspace-properties`。

### AWS CLI

修改的运行模式 Workspace

以下 `modify-workspace-properties` 示例将指定的运行模式设置 Workspace 为 `AUTO_STOP`。

```
aws workspaces modify-workspace-properties \  
  --workspace-id ws-dk1x zr417 \  
  --run-mode AUTO_STOP
```

```
--workspace-properties RunningMode=AUTO_STOP
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》Workspace 中的[修改](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyWorkspaceProperties](#)中的。

## modify-workspace-state

以下代码示例演示了如何使用 modify-workspace-state。

### AWS CLI

修改 a 的状态 Workspace

以下modify-workspace-state示例将指定的状态设置 Workspace 为ADMIN\_MAINTENANCE。

```
aws workspaces modify-workspace-state \  
  --workspace-id ws-dk1x zr417 \  
  --workspace-state ADMIN_MAINTENANCE
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》中的[Workspace 维护](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ModifyWorkspaceState](#)中的。

## reboot-workspaces

以下代码示例演示了如何使用 reboot-workspaces。

### AWS CLI

要重新启动 Workspace

以下reboot-workspaces示例重新启动指定 Workspace的。

```
aws workspaces reboot-workspaces \  
  --reboot-workspace-requests ws-dk1x zr417
```

输出：

```
{
  "FailedRequests": []
}
```

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》Workspace 中的[重启](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RebootWorkspaces](#) 中的。

## rebuild-workspaces

以下代码示例演示了如何使用 rebuild-workspaces。

### AWS CLI

要重建 Workspace

以下 rebuild-workspaces 示例将重建指定 Workspace 的。

```
aws workspaces rebuild-workspaces \
  --rebuild-workspace-requests ws-dk1xzr417
```

输出：

```
{
  "FailedRequests": []
}
```

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》Workspace 中的[重建](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[RebuildWorkspaces](#) 中的。

## register-workspace-directory

以下代码示例演示了如何使用 register-workspace-directory。

### AWS CLI

注册目录

以下 register-workspace-directory 示例注册了要在 Amazon 上使用的指定目录 WorkSpaces。

```
aws workspaces register-workspace-directory \  
  --directory-id d-926722edaf \  
  --no-enable-work-docs
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》WorkSpaces中的向[注册目录](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RegisterWorkspaceDirectory](#)中的。

## restore-workspace

以下代码示例演示了如何使用 restore-workspace。

### AWS CLI

要恢复 Workspace

以下restore-workspace示例恢复指定 Workspace的。

```
aws workspaces restore-workspace \  
  --workspace-id ws-dk1xzr417
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》Workspace中的“[恢复](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[RestoreWorkspace](#)中的。

## start-workspaces

以下代码示例演示了如何使用 start-workspaces。

### AWS CLI

要开始 AutoStop Workspace

以下start-workspaces示例从指定开始 Workspace。的运行模式 Workspace 必须为AutoStop。

```
aws workspaces start-workspaces \  
  --workspace-id ws-dk1xzr417
```



```
--start-workspace-requests WorkspaceId=ws-dk1xzr417
```

输出：

```
{
  "FailedRequests": []
}
```

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》 AutoStop WorkSpace中的“[停止并启动](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartWorkspaces](#)中的。

## stop-workspaces

以下代码示例演示了如何使用 stop-workspaces。

### AWS CLI

要停止 AutoStop WorkSpace

以下stop-workspaces示例停止指定的 WorkSpace。的运行模式 WorkSpace 必须为AutoStop。

```
aws workspaces stop-workspaces \
  --stop-workspace-requests WorkspaceId=ws-dk1xzr417
```

输出：

```
{
  "FailedRequests": []
}
```

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》 AutoStop WorkSpace中的“[停止并启动](#)”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopWorkspaces](#)中的。

## terminate-workspaces

以下代码示例演示了如何使用 terminate-workspaces。

## AWS CLI

### 要终止 Workspace

以下 `terminate-workspaces` 示例终止指定的工作空间。

```
aws workspaces terminate-workspaces \  
  --terminate-workspace-requests ws-dk1xzr417
```

输出：

```
{  
  "FailedRequests": []  
}
```

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》Workspace 中的“[删除](#)”。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [TerminateWorkspaces](#) 中的。

## 使用的 X-ray 示例 AWS CLI

以下代码示例向您展示了如何使用 with X-Ray 来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### **batch-traces-get**

以下代码示例演示了如何使用 `batch-traces-get`。

## AWS CLI

## 要获取跟踪列表

以下 `batch-get-traces` 示例检索由 ID 指定的跟踪列表。完整跟踪为每个分段包括一个文档，该文档根据收到的具有相同跟踪 ID 的所有分段文档编译。

```
aws xray batch-get-traces \
  --trace-ids 1-5d82881a-0a9126e92a73e971eed891b9
```

输出：

```
{
  "Traces": [
    {
      "Id": "1-5d82881a-0a9126e92a73e971eed891b9",
      "Duration": 0.232,
      "Segments": [
        {
          "Id": "54aff5735b12dd28",
          "Document": "{\"id\":\"54aff5735b12dd28\",\"name\":
\\\"Scorekeep\\\",\\\"start_time\\\":1.568835610432E9,\\\"end_time\\\":1.568835610664E9,
\\\"http\\\":{\\\"request\\\":{\\\"url\\\":\\\"http://scorekeep-env-1.m4fg2pfzpv.us-
east-2.elasticbeanstalk.com/api/user\\\",\\\"method\\\":\\\"POST\\\",\\\"user_agent\\\":
\\\"curl/7.59.0\\\",\\\"client_ip\\\":\\\"52.95.4.28\\\",\\\"x_forwarded_for\\\":true},
\\\"response\\\":{\\\"status\\\":200}},\\\"aws\\\":{\\\"elastic_beanstalk\\\":{\\\"version_label
\\\":\\\"Sample Application-1\\\",\\\"deployment_id\\\":3,\\\"environment_name\\\":\\\"Scorekeep-
env-1\\\"},\\\"ec2\\\":{\\\"availability_zone\\\":\\\"us-east-2b\\\",\\\"instance_id\\\":
\\\"i-0e3cf4d2de0f3f37a\\\"},\\\"xray\\\":{\\\"sdk_version\\\":\\\"1.1.0\\\",\\\"sdk\\\":\\\"X-Ray for
Java\\\"}},\\\"service\\\":{\\\"runtime\\\":\\\"OpenJDK 64-Bit Server VM\\\",\\\"runtime_version
\\\":\\\"1.8.0_222\\\"},\\\"trace_id\\\":\\\"1-5d82881a-0a9126e92a73e971eed891b9\\\",
\\\"origin\\\":\\\"AWS::ElasticBeanstalk::Environment\\\",\\\"subsegments\\\":[{\\\"id\\\":
\\\"2d6900034ccfe558\\\",\\\"name\\\":\\\"DynamoDB\\\",\\\"start_time\\\":1.568835610658E9,
\\\"end_time\\\":1.568835610664E9,\\\"http\\\":{\\\"response\\\":{\\\"status\\\":200,
\\\"content_length\\\":61}},\\\"aws\\\":{\\\"table_name\\\":\\\"scorekeep-user\\\",\\\"operation\\\":
\\\"UpdateItem\\\",\\\"request_id\\\":\\\"TPEIDNDUROMLPOV17U4A79555NVV4KQNS05AEMVJF66Q9ASUAAJG
\\\",\\\"resource_names\\\":[\\\"scorekeep-user\\\"]},\\\"namespace\\\":\\\"aws\\\"}]}"
        },
        {
          "Id": "0f278b6334c34e6b",
          "Document": "{\"id\":\"0f278b6334c34e6b\",\"name\":
\\\"DynamoDB\\\",\\\"start_time\\\":1.568835610658E9,\\\"end_time\\\":1.568835610664E9,
\\\"parent_id\\\":\\\"2d6900034ccfe558\\\",\\\"inferred\\\":true,\\\"http\\\":{\\\"response
```

```

\":{\\"status\\":200,\\\"content_length\\":61}},\\\"aws\\":{\\"table_name
\\":\\"scorekeep-user\\",\\\"operation\\":\\"UpdateItem\\",\\\"request_id\\":
\\\"TPEIDNDUR0MLP0V17U4A79555NVV4KQNS05AEMVJF66Q9ASUAAJG\\",\\\"resource_names\\":
[\\\"scorekeep-user\\"]},\\\"trace_id\\":\\"1-5d82881a-0a9126e92a73e971eed891b9\\",\\\"origin
\\":\\"AWS::DynamoDB::Table\\"}"
    }
  ]
}
],
"UnprocessedTraceIds": []
}

```

有关更多信息，请参阅《X-Ray 开发者指南》中的将 AWS X-Ray 与 [AP AWS I 与 CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [BatchTracesGet](#) 中的。

## create-group

以下代码示例演示了如何使用 create-group。

### AWS CLI

#### 创建群组

以下 create-group 示例创建了一个名为的组资源 AdminGroup。该组将获得一个筛选表达式，该表达式将组的标准定义为与导致故障或错误的特定服务相关的区段。

```

aws xray create-group \
  --group-name "AdminGroup" \
  --filter-expression "service(\"mydomain.com\") {fault OR error}"

```

输出：

```

{
  "GroupName": "AdminGroup",
  "GroupARN": "arn:aws:xray:us-west-2:123456789012:group/AdminGroup/123456789",
  "FilterExpression": "service(\"mydomain.com\") {fault OR error}"
}

```

有关更多信息，请参阅《X-Ray 开发者指南》中的“[使用 AWS X-Ray API 配置采样、分组和加密设置](#)”。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateGroup](#)中的。

## create-sampling-rule

以下代码示例演示了如何使用 create-sampling-rule。

### AWS CLI

#### 创建采样规则

以下create-sampling-rule示例创建了一条规则，用于控制已检测应用程序的采样行为。这些规则由 JSON 文件提供。大多数采样规则字段都是创建规则所必需的。

```
aws xray create-sampling-rule \  
  --cli-input-json file://9000-base-scorekeep.json
```

9000-base-scorekeep.json 的内容：

```
{  
  "SamplingRule": {  
    "RuleName": "base-scorekeep",  
    "ResourceARN": "*",  
    "Priority": 9000,  
    "FixedRate": 0.1,  
    "ReservoirSize": 5,  
    "ServiceName": "Scorekeep",  
    "ServiceType": "*",  
    "Host": "*",  
    "HTTPMethod": "*",  
    "URLPath": "*",  
    "Version": 1  
  }  
}
```

输出：

```
{  
  "SamplingRuleRecord": {  
    "SamplingRule": {  
      "RuleName": "base-scorekeep",  
      "RuleARN": "arn:aws:xray:us-west-2:123456789012:sampling-rule/base-scorekeep",  
      "ResourceARN": "*",  
      "Priority": 9000,  
      "FixedRate": 0.1,  
      "ReservoirSize": 5,  
      "ServiceName": "Scorekeep",  
      "ServiceType": "*",  
      "Host": "*",  
      "HTTPMethod": "*",  
      "URLPath": "*",  
      "Version": 1  
    }  
  }  
}
```

```
    "ResourceARN": "*",
    "Priority": 9000,
    "FixedRate": 0.1,
    "ReservoirSize": 5,
    "ServiceName": "Scorekeep",
    "ServiceType": "*",
    "Host": "*",
    "HTTPMethod": "*",
    "URLPath": "*",
    "Version": 1,
    "Attributes": {}
  },
  "CreatedAt": 1530574410.0,
  "ModifiedAt": 1530574410.0
}
```

有关更多信息，请参阅《X-Ray 开发者指南》中的[“使用 AWS X-Ray API 配置采样、分组和加密设置”](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateSamplingRule](#)中的。

## delete-group

以下代码示例演示了如何使用 delete-group。

### AWS CLI

#### 删除群组

以下delete-group示例删除了指定的组资源。

```
aws xray delete-group \
  --group-name "AdminGroup" \
  --group-arn "arn:aws:xray:us-east-2:123456789012:group/AdminGroup/123456789"
```

此命令不生成任何输出。

有关更多信息，请参阅《X-Ray 开发者指南》中的[“使用 AWS X-Ray API 配置采样、分组和加密设置”](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteGroup](#)中的。

## delete-sampling-rule

以下代码示例演示了如何使用 delete-sampling-rule。

### AWS CLI

#### 删除采样规则

以下delete-sampling-rule示例删除了指定的采样规则。您可以使用群组名称或群组 ARN 来指定群组。

```
aws xray delete-sampling-rule \  
  --rule-name polling-scorekeep
```

输出：

```
{  
  "SamplingRuleRecord": {  
    "SamplingRule": {  
      "RuleName": "polling-scorekeep",  
      "RuleARN": "arn:aws:xray:us-west-2:123456789012:sampling-rule/polling-  
scorekeep",  
      "ResourceARN": "*",  
      "Priority": 5000,  
      "FixedRate": 0.003,  
      "ReservoirSize": 0,  
      "ServiceName": "Scorekeep",  
      "ServiceType": "*",  
      "Host": "*",  
      "HTTPMethod": "GET",  
      "URLPath": "/api/state/*",  
      "Version": 1,  
      "Attributes": {}  
    },  
    "CreatedAt": 1530574399.0,  
    "ModifiedAt": 1530574399.0  
  }  
}
```

有关更多信息，请参阅《X-Ray 开发者指南》中的[“使用 AWS X-Ray API 配置采样、分组和加密设置”](#)。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteSamplingRule](#)中的。

## get-encryption-config

以下代码示例演示了如何使用 `get-encryption-config`。

### AWS CLI

#### 检索加密配置

以下 `get-encryption-config` 示例检索 AWS X-Ray 数据的当前加密配置。

```
aws xray get-encryption-config
```

输出：

```
{
  "EncryptionConfig": {
    "KeyId": "ae4aa6d49-a4d8-9df9-a475-4ff6d7898456",
    "Status": "ACTIVE",
    "Type": "NONE"
  }
}
```

有关更多信息，请参阅《X-Ray 开发者指南》中的“[使用 AWS X-Ray API 配置采样、分组和加密设置](#)”。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetEncryptionConfig](#) 中的。

## get-group

以下代码示例演示了如何使用 `get-group`。

### AWS CLI

#### 检索群组

以下 `get-group` 示例显示了指定组资源的详细信息。详细信息包括群组名称、群组 ARN 以及定义该群组标准的筛选表达式。ARN 也可以检索群组。

```
aws xray get-group \
  --group-name "AdminGroup"
```

输出：



```
{
  "Group": [
    {
      "GroupName": "AdminGroup",
      "GroupARN": "arn:aws:xray:us-west-2:123456789012:group/
AdminGroup/123456789",
      "FilterExpression": "service(\"mydomain.com\") {fault OR error}"
    }
  ]
}
```

有关更多信息，请参阅《X-Ray 开发者指南》中的“[使用 AWS X-Ray API 配置采样、分组和加密设置](#)”。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetGroup](#) 中的。

## get-groups

以下代码示例演示了如何使用 get-groups。

### AWS CLI

#### 检索所有群组

以下示例显示所有活动组的详细信息。

```
aws xray get-groups
```

输出：

```
{
  "Groups": [
    {
      "GroupName": "AdminGroup",
      "GroupARN": "arn:aws:xray:us-west-2:123456789012:group/
AdminGroup/123456789",
      "FilterExpression": "service(\"example.com\") {fault OR error}"
    },
    {
      "GroupName": "SDETGroup",
      "GroupARN": "arn:aws:xray:us-west-2:123456789012:group/
SDETGroup/987654321",

```

```
        "FilterExpression": "responsetime > 2"
      }
    ]
  }
```

有关更多信息，请参阅《X-Ray 开发者指南》中的“[使用 AWS X-Ray API 配置采样、分组和加密设置](#)”。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetGroups](#) 中的。

## get-sampling-rules

以下代码示例演示了如何使用 `get-sampling-rules`。

### AWS CLI

检索所有采样规则

以下 `get-sampling-rules` 示例显示了所有可用采样规则的详细信息。：

```
aws xray get-sampling-rules
```

输出：

```
{
  "SamplingRuleRecords": [
    {
      "SamplingRule": {
        "RuleName": "Default",
        "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/Default",
        "ResourceARN": "*",
        "Priority": 10000,
        "FixedRate": 0.01,
        "ReservoirSize": 0,
        "ServiceName": "*",
        "ServiceType": "*",
        "Host": "*",
        "HTTPMethod": "*",
        "URLPath": "*",
        "Version": 1,
        "Attributes": {}
      },
      "CreatedAt": 0.0,
```

```
    "ModifiedAt": 1530558121.0
  },
  {
    "SamplingRule": {
      "RuleName": "base-scorekeep",
      "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/base-scorekeep",
      "ResourceARN": "*",
      "Priority": 9000,
      "FixedRate": 0.1,
      "ReservoirSize": 2,
      "ServiceName": "Scorekeep",
      "ServiceType": "*",
      "Host": "*",
      "HTTPMethod": "*",
      "URLPath": "*",
      "Version": 1,
      "Attributes": {}
    },
    "CreatedAt": 1530573954.0,
    "ModifiedAt": 1530920505.0
  },
  {
    "SamplingRule": {
      "RuleName": "polling-scorekeep",
      "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/polling-
scorekeep",
      "ResourceARN": "*",
      "Priority": 5000,
      "FixedRate": 0.003,
      "ReservoirSize": 0,
      "ServiceName": "Scorekeep",
      "ServiceType": "*",
      "Host": "*",
      "HTTPMethod": "GET",
      "URLPath": "/api/state/*",
      "Version": 1,
      "Attributes": {}
    },
    "CreatedAt": 1530918163.0,
    "ModifiedAt": 1530918163.0
  }
]
}
```

有关更多信息，请参阅《X-Ray 开发者指南》中的将采样规则与 X-AWS Ray API 配合使用。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetSamplingRules](#) 中的。

## get-sampling-targets

以下代码示例演示了如何使用 get-sampling-targets。

### AWS CLI

#### 申请抽样配额

以下 get-sampling-targets 示例请求服务用于采样请求的规则采样配额。AWS X-Ray 的回应包括一个配额，该配额可以用来代替向水库借款。

```
aws xray get-sampling-targets \  
  --sampling-statistics-documents '[ { "RuleName": "base-scorekeep", "ClientID":  
  "ABCDEF1234567890ABCDEF10", "Timestamp": "2018-07-07T00:20:06", "RequestCount": 110,  
  "SampledCount": 20, "BorrowCount": 10 }, { "RuleName": "polling-scorekeep", 31,  
  "BorrowCount": 0 } ]'
```

输出：

```
{  
  "SamplingTargetDocuments": [  
    {  
      "RuleName": "base-scorekeep",  
      "FixedRate": 0.1,  
      "ReservoirQuota": 2,  
      "ReservoirQuotaTTL": 1530923107.0,  
      "Interval": 10  
    },  
    {  
      "RuleName": "polling-scorekeep",  
      "FixedRate": 0.003,  
      "ReservoirQuota": 0,  
      "ReservoirQuotaTTL": 1530923107.0,  
      "Interval": 10  
    }  
  ],  
  "LastRuleModification": 1530920505.0,  
  "UnprocessedStatistics": []  
}
```

有关更多信息，请参阅《X-Ray 开发者指南》中的将采样规则与 X-AWS Ray API 配合使用。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetSamplingTargets](#) 中的。

## get-service-graph

以下代码示例演示了如何使用 get-service-graph。

### AWS CLI

#### 获取服务图

以下示例显示了指定时间段内的文档，该文档描述了处理传入请求的服务以及它们因此调用的下游服务。：

```
aws xray get-service-graph \  
  --start-time 1568835392.0 \  
  --end-time 1568835446.0
```

输出：

```
{  
  "Services": [  
    {  
      "ReferenceId": 0,  
      "Name": "Scorekeep",  
      "Names": [  
        "Scorekeep"  
      ],  
      "Root": true,  
      "Type": "AWS::ElasticBeanstalk::Environment",  
      "State": "active",  
      "StartTime": 1568835392.0,  
      "EndTime": 1568835446.0,  
      "Edges": [  
        {  
          "ReferenceId": 1,  
          "StartTime": 1568835392.0,  
          "EndTime": 1568835446.0,  
          "SummaryStatistics": {  
            "OkCount": 14,  
            "ErrorStatistics": {  
              "ThrottleCount": 0,
```

```
        "OtherCount": 0,  
        "TotalCount": 0  
    },  
    "FaultStatistics": {  
        "OtherCount": 0,  
        "TotalCount": 0  
    },  
    "TotalCount": 14,  
    "TotalResponseTime": 0.13  
},  
"ResponseTimeHistogram": [  
    {  
        "Value": 0.008,  
        "Count": 1  
    },  
    {  
        "Value": 0.005,  
        "Count": 7  
    },  
    {  
        "Value": 0.009,  
        "Count": 1  
    },  
    {  
        "Value": 0.021,  
        "Count": 1  
    },  
    {  
        "Value": 0.038,  
        "Count": 1  
    },  
    {  
        "Value": 0.007,  
        "Count": 1  
    },  
    {  
        "Value": 0.006,  
        "Count": 2  
    }  
],  
"Aliases": []  
},  
  
... TRUNCATED FOR BREVITY ...
```

```

        ]
    }
],
"StartTime": 1568835392.0,
"EndTime": 1568835446.0,
"ContainsOldGroupVersions": false
}

```

有关更多信息，请参阅《X-Ray 开发者指南》中的将 AWS X-Ray 与 [AP AWS I 与 CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [GetServiceGraph](#) 中的。

## get-trace-summaries

以下代码示例演示了如何使用 `get-trace-summaries`。

### AWS CLI

#### 获取追踪摘要

以下 `get-trace-summaries` 示例检索指定时间范围内可用的跟踪的 ID 和元数据。

```

aws xray get-trace-summaries \
  --start-time 1568835392.0 \
  --end-time 1568835446.0

```

#### 输出：

```

[
  "http://scorekeep-env-1.123456789.us-east-2.elasticbeanstalk.com/api/move/
VSAE93HF/GSSD2NTB/DP0PCC09",
  "http://scorekeep-env-1.123456789.us-east-2.elasticbeanstalk.com/api/move/
GCQ2B35P/FREELDFT/4LRE643M",
  "http://scorekeep-env-1.123456789.us-east-2.elasticbeanstalk.com/api/game/
VSAE93HF/GSSD2NTB/starttime/1568835513",
  "http://scorekeep-env-1.123456789.us-east-2.elasticbeanstalk.com/api/
move/4MQNA5NN/L99KK2RF/null"
]

```

有关更多信息，请参阅《X-Ray 开发者指南》中的[将 AWS X-Ray 与 AWS API 与 CLI 配合使用](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetTraceSummaries](#)中的。

## put-encryption-config

以下代码示例演示了如何使用 put-encryption-config。

### AWS CLI

#### 更新加密配置

以下 put-encryption-config 示例更新加密配置，以便 AWS X-Ray 数据使用默认 AWS 托管 KMS 密钥 `aws/xray`。

```
aws xray put-encryption-config \
  --type KMS \
  --key-id alias/aws/xray
```

输出：

```
{
  "EncryptionConfig": {
    "KeyId": "arn:aws:kms:us-west-2:123456789012:key/c234g4e8-39e9-4gb0-84e2-
b0ea215cbba5",
    "Status": "UPDATING",
    "Type": "KMS"
  }
}
```

有关更多信息，请参阅《X-Ray 开发者指南》中的[“使用 AWS X-Ray API 配置采样、分组和加密设置”](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[PutEncryptionConfig](#)中的。

## put-trace-segments

以下代码示例演示了如何使用 put-trace-segments。



## AWS CLI

### 上传片段

以下put-trace-segments示例将分段文档上传到 AWS X-Ray。区段文档作为 JSON 区段文档列表使用。

```
aws xray put-trace-segments \  
  --trace-segment-documents '{"id":"20312a0e2b8809f4","name\  
":"DynamoDB","trace_id":"1-5832862d-a43aafded3334a971fe312db",\  
"start_time":1.479706157195E9,"end_time":1.479706157202E9,"parent_id":\  
"79736b962fe3239e","http":{"response":{"content_length":60,"status\  
":200}},"inferred":true,"aws":{"consistent_read":false,"table_name\  
":"scorekeep-session-xray","operation":"GetItem","request_id":\  
"SCAU230M6M8F038UASGC7785ARVV4KQNS05AEMVJF66Q9ASUAAJG"},"resource_names":\  
["scorekeep-session-xray"],"origin":"AWS::DynamoDB::Table"}'
```

输出：

```
{  
  "UnprocessedTraceSegments": []  
}
```

有关更多信息，请参阅《X-Ray 开发人员指南》中的向 AWS X-Ray 发送跟踪数据。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutTraceSegments](#)中的。

## update-group

以下代码示例演示了如何使用 update-group。

### AWS CLI

#### 更新群组

以下update-group示例更新了接受跟踪到名为的组的标准AdminGroup。您可以使用组名或组ARN 来指定所需的组。

```
aws xray update-group \  
  --group-name "AdminGroup" \  
  --group-arn "arn:aws:xray:us-west-2:123456789012:group/AdminGroup/123456789" \  
  --filter-expression "service(\"mydomain.com\") {fault}"
```

输出：

```
{
  "GroupName": "AdminGroup",
  "GroupARN": "arn:aws:xray:us-east-2:123456789012:group/AdminGroup/123456789",
  "FilterExpression": "service(\"mydomain.com\") {fault}"
}
```

有关更多信息，请参阅《X-Ray 开发者指南》中的“[使用 AWS X-Ray API 配置采样、分组和加密设置](#)”。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [UpdateGroup](#) 中的。

## update-sampling-rule

以下代码示例演示了如何使用 update-sampling-rule。

### AWS CLI

#### 更新采样规则

以下 update-sampling-rule 示例修改了采样规则的配置。这些规则是从 JSON 文件中使用的。只有要更新的字段才是必填字段。

```
aws xray update-sampling-rule \
  --cli-input-json file://1000-default.json
```

1000-default.json 的内容：

```
{
  "SamplingRuleUpdate": {
    "RuleName": "Default",
    "FixedRate": 0.01,
    "ReservoirSize": 0
  }
}
```

输出：

```
{
  "SamplingRuleRecords": [
```

```
{
  "SamplingRule": {
    "RuleName": "Default",
    "RuleARN": "arn:aws:xray:us-west-2:123456789012:sampling-rule/
Default",
    "ResourceARN": "*",
    "Priority": 10000,
    "FixedRate": 0.01,
    "ReservoirSize": 0,
    "ServiceName": "*",
    "ServiceType": "*",
    "Host": "*",
    "HTTPMethod": "*",
    "URLPath": "*",
    "Version": 1,
    "Attributes": {}
  },
  "CreatedAt": 0.0,
  "ModifiedAt": 1529959993.0
}
]
```

有关更多信息，请参阅《X-Ray 开发者指南》中的[“使用 AWS X-Ray API 配置采样、分组和加密设置”](#)。AWS

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[UpdateSamplingRule](#) 中的。

## AWS CLI 使用 Bash 脚本代码示例

本主题中的代码示例向您展示了如何将 with Bash 脚本 AWS Command Line Interface 与一起 AWS 使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

跨服务示例是指跨多个 AWS 服务工作的示例应用程序。

示例

- [AWS CLI 与 Bash 脚本配合使用的操作和场景](#)

## AWS CLI 与 Bash 脚本配合使用的操作和场景

以下代码示例展示了如何通过将 with Bash 脚本 AWS Command Line Interface 与一起使用来执行操作和实现常见场景。AWS 服务

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是指显示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

### 服务

- [使用 Bash 脚本的 DynamoDB 示例 AWS CLI](#)
- [HealthImaging AWS CLI 与 Bash 脚本一起使用的示例](#)
- [AWS CLI 与 Bash 脚本一起使用的 IAM 示例](#)
- [使用 AWS CLI Bash 脚本的 Amazon S3 示例](#)
- [AWS STSAWS CLI 与 Bash 脚本一起使用的示例](#)

### 使用 Bash 脚本的 DynamoDB 示例 AWS CLI

以下代码示例向您展示了如何在 DynamoDB 中使用 with Bash 脚本来执行操作和实现常见场景。AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题


- [操作](#)
- [场景](#)

### 操作

#### BatchGetItem

以下代码示例演示了如何使用 BatchGetItem。

## AWS CLI 使用 Bash 脚本

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function dynamodb_batch_get_item
#
# This function gets a batch of items from a DynamoDB table.
#
# Parameters:
#     -i item -- Path to json file containing the keys of the items to get.
#
# Returns:
#     The items as json output.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_batch_get_item() {
    local item response
    local option OPTARG # Required to use getopt command in a function.

    #####
    # Function usage explanation
    #####
    function usage() {
        echo "function dynamodb_batch_get_item"
        echo "Get a batch of items from a DynamoDB table."
        echo " -i item -- Path to json file containing the keys of the items to get."
        echo ""
    }

    while getopt "i:h" option; do
        case "${option}" in
            i) item="${OPTARG}" ;;
            h)
                usage
                return 0
        esac
    done
}
```

```

        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done
export OPTIND=1

if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
fi

response=$(aws dynamodb batch-get-item \
    --request-items file://"${item}")
local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-get-item operation failed.$response"
    return 1
fi

echo "$response"

return 0
}

```

本示例中使用的实用程序函数。

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

```

```
#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi

    return 0
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchGetItem](#)中的。

## BatchWriteItem

以下代码示例演示了如何使用 BatchWriteItem。

## AWS CLI 使用 Bash 脚本

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function dynamodb_batch_write_item
#
# This function writes a batch of items into a DynamoDB table.
#
# Parameters:
#     -i item -- Path to json file containing the items to write.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_batch_write_item() {
    local item response
    local option OPTARG # Required to use getopt command in a function.

    #####
    # Function usage explanation
    #####
    function usage() {
        echo "function dynamodb_batch_write_item"
        echo "Write a batch of items into a DynamoDB table."
        echo " -i item -- Path to json file containing the items to write."
        echo ""
    }
    while getopt "i:h" option; do
        case "${option}" in
            i) item="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
```



```

        usage
        return 1
    ;;
    esac
done
export OPTIND=1

if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "    table_name:  $table_name"
iecho "    item:        $item"
iecho ""

response=$(aws dynamodb batch-write-item \
    --request-items file://"${item}")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-write-item operation failed.$response"
    return 1
fi

return 0
}

```

本示例中使用的实用程序函数。

```

#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then

```

```

    echo "$@"
  fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-return-codes.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho " One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho " Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho " Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho " Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho " The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho " The service returned an error."
  elif [ "$err_code" == 255 ]; then

```

```

    errecho " 255 is a catch-all error."
fi

return 0
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[BatchWriteItem](#)中的。

## CreateTable

以下代码示例演示了如何使用 CreateTable。

### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# function dynamodb_create_table
#
# This function creates an Amazon DynamoDB table.
#
# Parameters:
#     -n table_name -- The name of the table to create.
#     -a attribute_definitions -- JSON file path of a list of attributes and their
types.
#     -k key_schema -- JSON file path of a list of attributes and their key types.
#     -p provisioned_throughput -- Provisioned throughput settings for the table.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_create_table() {
    local table_name attribute_definitions key_schema provisioned_throughput response
    local option OPTARG # Required to use getopt command in a function.

#####

```

```
# Function usage explanation
#####
function usage() {
    echo "function dynamodb_create_table"
    echo "Creates an Amazon DynamoDB table."
    echo " -n table_name -- The name of the table to create."
    echo " -a attribute_definitions -- JSON file path of a list of attributes and
their types."
    echo " -k key_schema -- JSON file path of a list of attributes and their key
types."
    echo " -p provisioned_throughput -- Provisioned throughput settings for the
table."
    echo ""
}

# Retrieve the calling parameters.
while getopts "n:a:k:p:h" option; do
    case "${option}" in
        n) table_name="${OPTARG}" ;;
        a) attribute_definitions="${OPTARG}" ;;
        k) key_schema="${OPTARG}" ;;
        p) provisioned_throughput="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$attribute_definitions" ]]; then
    errecho "ERROR: You must provide an attribute definitions json file path the -a
parameter."
fi
```

```
usage
return 1
fi

if [[ -z "$key_schema" ]]; then
    errecho "ERROR: You must provide a key schema json file path the -k parameter."
    usage
    return 1
fi

if [[ -z "$provisioned_throughput" ]]; then
    errecho "ERROR: You must provide a provisioned throughput json file path the -p
parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "  table_name:  $table_name"
iecho "  attribute_definitions:  $attribute_definitions"
iecho "  key_schema:  $key_schema"
iecho "  provisioned_throughput:  $provisioned_throughput"
iecho ""

response=$(aws dynamodb create-table \
  --table-name "$table_name" \
  --attribute-definitions file://"${attribute_definitions}" \
  --key-schema file://"${key_schema}" \
  --provisioned-throughput "$provisioned_throughput")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-table operation failed.$response"
    return 1
fi

return 0
}
```

本示例中使用的实用程序函数。

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-return-codes.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    fi
}
```

```

elif [ "$err_code" == 130 ]; then
    errecho " Process received SIGINT."
elif [ "$err_code" == 252 ]; then
    errecho " Command syntax invalid."
elif [ "$err_code" == 253 ]; then
    errecho " The system environment or configuration was invalid."
elif [ "$err_code" == 254 ]; then
    errecho " The service returned an error."
elif [ "$err_code" == 255 ]; then
    errecho " 255 is a catch-all error."
fi

return 0
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateTable](#)中的。

## DeleteItem

以下代码示例演示了如何使用 DeleteItem。

AWS CLI 使用 Bash 脚本

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# function dynamodb_delete_item
#
# This function deletes an item from a DynamoDB table.
#
# Parameters:
#     -n table_name  -- The name of the table.
#     -k keys        -- Path to json file containing the keys that identify the item to
#     delete.
#
# Returns:
#     0 - If successful.

```

```

#      1 - If it fails.
#####
function dynamodb_delete_item() {
    local table_name keys response
    local option OPTARG # Required to use getopt command in a function.

    # #####
    # Function usage explanation
    #####
    function usage() {
        echo "function dynamodb_delete_item"
        echo "Delete an item from a DynamoDB table."
        echo " -n table_name -- The name of the table."
        echo " -k keys -- Path to json file containing the keys that identify the item
to delete."
        echo ""
    }
    while getopt "n:k:h" option; do
        case "${option}" in
            n) table_name="${OPTARG}" ;;
            k) keys="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$table_name" ]]; then
        errecho "ERROR: You must provide a table name with the -n parameter."
        usage
        return 1
    fi

    if [[ -z "$keys" ]]; then
        errecho "ERROR: You must provide a keys json file path the -k parameter."
        usage
        return 1
    fi
}

```



```

fi

iecho "Parameters:\n"
iecho "  table_name:  $table_name"
iecho "  keys:    $keys"
iecho ""

response=$(aws dynamodb delete-item \
  --table-name "$table_name" \
  --key file://"$keys")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-item operation failed.$response"
  return 1
fi

return 0
}

```

本示例中使用的实用程序函数。

```

#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####

```

```

function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi

    return 0
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteItem](#)中的。

## DeleteTable

以下代码示例演示了如何使用 DeleteTable。

### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function dynamodb_delete_table
#
# This function deletes a DynamoDB table.
#
# Parameters:
#     -n table_name  -- The name of the table to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_delete_table() {
    local table_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function dynamodb_delete_table"
        echo "Deletes an Amazon DynamoDB table."
        echo " -n table_name  -- The name of the table to delete."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:h" option; do
        case "${option}" in
            n) table_name="${OPTARG}" ;;
            h)
                usage
                return 0
        esac
    done
}
```

```

        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "    table_name:  $table_name"
iecho ""

response=$(aws dynamodb delete-table \
    --table-name "$table_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-table operation failed.$response"
    return 1
fi

return 0
}

```

本示例中使用的实用程序函数。

```

#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####

```

```

function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-return-codes.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then

```

```

    errecho " The service returned an error."
elif [ "$err_code" == 255 ]; then
    errecho " 255 is a catch-all error."
fi

return 0
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteTable](#)中的。

## DescribeTable

以下代码示例演示了如何使用 DescribeTable。

### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# function dynamodb_describe_table
#
# This function returns the status of a DynamoDB table.
#
# Parameters:
#     -n table_name  -- The name of the table.
#
# Response:
#     - TableStatus:
#     And:
#     0 - Table is active.
#     1 - If it fails.
#####
function dynamodb_describe_table {
    local table_name
    local option OPTARG # Required to use getopt command in a function.

#####

```

```
# Function usage explanation
#####
function usage() {
    echo "function dynamodb_describe_table"
    echo "Describe the status of a DynamoDB table."
    echo "  -n table_name  -- The name of the table."
    echo ""
}

# Retrieve the calling parameters.
while getopts "n:h" option; do
    case "${option}" in
        n) table_name="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
fi

local table_status
table_status=$(
    aws dynamodb describe-table \
        --table-name "$table_name" \
        --output text \
        --query 'Table.TableStatus'
)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log "$error_code"
```

```

    errecho "ERROR: AWS reports describe-table operation failed.$table_status"
    return 1
fi

echo "$table_status"

return 0
}

```

本示例中使用的实用程序函数。

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-return-codes.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    fi
}

```



```

elif [ "$err_code" == 130 ]; then
    errecho " Process received SIGINT."
elif [ "$err_code" == 252 ]; then
    errecho " Command syntax invalid."
elif [ "$err_code" == 253 ]; then
    errecho " The system environment or configuration was invalid."
elif [ "$err_code" == 254 ]; then
    errecho " The service returned an error."
elif [ "$err_code" == 255 ]; then
    errecho " 255 is a catch-all error."
fi

return 0
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeTable](#)中的。

## GetItem

以下代码示例演示了如何使用 GetItem。

### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# function dynamodb_get_item
#
# This function gets an item from a DynamoDB table.
#
# Parameters:
#     -n table_name  -- The name of the table.
#     -k keys       -- Path to json file containing the keys that identify the item to
get.
#     [-q query]    -- Optional JMESPath query expression.
#
# Returns:

```

```

#       The item as text output.
# And:
#       0 - If successful.
#       1 - If it fails.
#####
function dynamodb_get_item() {
    local table_name keys query response
    local option OPTARG # Required to use getopt command in a function.

    # #####
    # Function usage explanation
    #####
    function usage() {
        echo "function dynamodb_get_item"
        echo "Get an item from a DynamoDB table."
        echo " -n table_name -- The name of the table."
        echo " -k keys -- Path to json file containing the keys that identify the item
to get."
        echo " [-q query] -- Optional JMESPath query expression."
        echo ""
    }
    query=""
    while getopt "n:k:q:h" option; do
        case "${option}" in
            n) table_name="${OPTARG}" ;;
            k) keys="${OPTARG}" ;;
            q) query="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$table_name" ]]; then
        errecho "ERROR: You must provide a table name with the -n parameter."
        usage
        return 1
    fi
}

```

```
fi

if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
fi

if [[ -n "$query" ]]; then
    response=$(aws dynamodb get-item \
        --table-name "$table_name" \
        --key file://"${keys}" \
        --output text \
        --query "$query")
else
    response=$(
        aws dynamodb get-item \
            --table-name "$table_name" \
            --key file://"${keys}" \
            --output text
    )
fi

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports get-item operation failed.$response"
    return 1
fi

if [[ -n "$query" ]]; then
    echo "$response" | sed "/^\t/s/\t//1" # Remove initial tab that the JMSEPath
query inserts on some strings.
else
    echo "$response"
fi

return 0
}
```

本示例中使用的实用程序函数。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-return-codes.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi
}
```

```
    return 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetItem](#)中的。

## ListTables

以下代码示例演示了如何使用 ListTables。

AWS CLI 使用 Bash 脚本

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function dynamodb_list_tables
#
# This function lists all the tables in a DynamoDB.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_list_tables() {
    response=$(aws dynamodb list-tables \
        --output text \
        --query "TableNames")

    local error_code=${?}

    if [[ $error_code -ne 0 ]]; then
        aws_cli_error_log $error_code
        errecho "ERROR: AWS reports batch-write-item operation failed.$response"
        return 1
    fi

    echo "$response" | tr -s "[:space:]" "\n"
```

```
    return 0
}
```

本示例中使用的实用程序函数。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-return-codes.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    }
}
```

```

elif [ "$err_code" == 254 ]; then
    errecho " The service returned an error."
elif [ "$err_code" == 255 ]; then
    errecho " 255 is a catch-all error."
fi

return 0
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListTables](#)中的。

## PutItem

以下代码示例演示了如何使用 PutItem。

### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# function dynamodb_put_item
#
# This function puts an item into a DynamoDB table.
#
# Parameters:
#     -n table_name -- The name of the table.
#     -i item -- Path to json file containing the item values.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_put_item() {
    local table_name item response
    local option OPTARG # Required to use getopt command in a function.

#####

```

```
# Function usage explanation
#####
function usage() {
    echo "function dynamodb_put_item"
    echo "Put an item into a DynamoDB table."
    echo " -n table_name -- The name of the table."
    echo " -i item -- Path to json file containing the item values."
    echo ""
}

while getopts "n:i:h" option; do
    case "${option}" in
        n) table_name="${OPTARG}" ;;
        i) item="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "    table_name:  $table_name"
iecho "    item:       $item"
iecho ""
iecho ""
```



```

response=$(aws dynamodb put-item \
  --table-name "$table_name" \
  --item file://"item")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports put-item operation failed.$response"
  return 1
fi

return 0
}

```

本示例中使用的实用程序函数。

```

#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()

```

```

#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi

    return 0
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutItem](#)中的。

## Query

以下代码示例演示了如何使用 Query。

## AWS CLI 使用 Bash 脚本

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function dynamodb_query
#
# This function queries a DynamoDB table.
#
# Parameters:
#     -n table_name -- The name of the table.
#     -k key_condition_expression -- The key condition expression.
#     -a attribute_names -- Path to JSON file containing the attribute names.
#     -v attribute_values -- Path to JSON file containing the attribute values.
#     [-p projection_expression] -- Optional projection expression.
#
# Returns:
#     The items as json output.
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_query() {
    local table_name key_condition_expression attribute_names attribute_values
    projection_expression response
    local option OPTARG # Required to use getopt command in a function.

    # #####
    # Function usage explanation
    # #####
    function usage() {
        echo "function dynamodb_query"
        echo "Query a DynamoDB table."
        echo " -n table_name -- The name of the table."
        echo " -k key_condition_expression -- The key condition expression."
        echo " -a attribute_names -- Path to JSON file containing the attribute names."
        echo " -v attribute_values -- Path to JSON file containing the attribute
values."
```

```
    echo " [-p projection_expression] -- Optional projection expression."
    echo ""
}

while getopts "n:k:a:v:p:h" option; do
    case "${option}" in
        n) table_name="${OPTARG}" ;;
        k) key_condition_expression="${OPTARG}" ;;
        a) attribute_names="${OPTARG}" ;;
        v) attribute_values="${OPTARG}" ;;
        p) projection_expression="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$key_condition_expression" ]]; then
    errecho "ERROR: You must provide a key condition expression with the -k
parameter."
    usage
    return 1
fi

if [[ -z "$attribute_names" ]]; then
    errecho "ERROR: You must provide a attribute names with the -a parameter."
    usage
    return 1
fi

if [[ -z "$attribute_values" ]]; then
```

```

    errecho "ERROR: You must provide a attribute values with the -v parameter."
    usage
    return 1
fi

if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb query \
        --table-name "$table_name" \
        --key-condition-expression "$key_condition_expression" \
        --expression-attribute-names file://"$attribute_names" \
        --expression-attribute-values file://"$attribute_values")
else
    response=$(aws dynamodb query \
        --table-name "$table_name" \
        --key-condition-expression "$key_condition_expression" \
        --expression-attribute-names file://"$attribute_names" \
        --expression-attribute-values file://"$attribute_values" \
        --projection-expression "$projection_expression")
fi

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports query operation failed.$response"
    return 1
fi

echo "$response"

return 0
}

```

本示例中使用的实用程序函数。

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

```

```
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi

    return 0
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [Query](#)。

## Scan

以下代码示例演示了如何使用 Scan。

### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function dynamodb_scan
#
# This function scans a DynamoDB table.
#
# Parameters:
#     -n table_name -- The name of the table.
#     -f filter_expression -- The filter expression.
#     -a expression_attribute_names -- Path to JSON file containing the expression
attribute names.
#     -v expression_attribute_values -- Path to JSON file containing the
expression attribute values.
#     [-p projection_expression] -- Optional projection expression.
#
# Returns:
#     The items as json output.
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_scan() {
    local table_name filter_expression expression_attribute_names
expression_attribute_values projection_expression response
    local option OPTARG # Required to use getopt command in a function.

    # #####
    # Function usage explanation
    #####
    function usage() {
        echo "function dynamodb_scan"
        echo "Scan a DynamoDB table."
    }
}
```

```
    echo " -n table_name -- The name of the table."
    echo " -f filter_expression -- The filter expression."
    echo " -a expression_attribute_names -- Path to JSON file containing the
expression attribute names."
    echo " -v expression_attribute_values -- Path to JSON file containing the
expression attribute values."
    echo " [-p projection_expression] -- Optional projection expression."
    echo ""
}

while getopts "n:f:a:v:p:h" option; do
    case "${option}" in
        n) table_name="${OPTARG}" ;;
        f) filter_expression="${OPTARG}" ;;
        a) expression_attribute_names="${OPTARG}" ;;
        v) expression_attribute_values="${OPTARG}" ;;
        p) projection_expression="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$filter_expression" ]]; then
    errecho "ERROR: You must provide a filter expression with the -f parameter."
    usage
    return 1
fi

if [[ -z "$expression_attribute_names" ]]; then
```



```
errecho "ERROR: You must provide expression attribute names with the -a
parameter."
usage
return 1
fi

if [[ -z "$expression_attribute_values" ]]; then
errecho "ERROR: You must provide expression attribute values with the -v
parameter."
usage
return 1
fi

if [[ -z "$projection_expression" ]]; then
response=$(aws dynamodb scan \
--table-name "$table_name" \
--filter-expression "$filter_expression" \
--expression-attribute-names file://"${expression_attribute_names}" \
--expression-attribute-values file://"${expression_attribute_values}")
else
response=$(aws dynamodb scan \
--table-name "$table_name" \
--filter-expression "$filter_expression" \
--expression-attribute-names file://"${expression_attribute_names}" \
--expression-attribute-values file://"${expression_attribute_values}" \
--projection-expression "$projection_expression")
fi

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
aws_cli_error_log $error_code
errecho "ERROR: AWS reports scan operation failed.$response"
return 1
fi

echo "$response"

return 0
}
```

本示例中使用的实用程序函数。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-return-codes.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi
}
```

```
    return 0
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 [Scan](#)。

## UpdateItem

以下代码示例演示了如何使用 UpdateItem。

### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function dynamodb_update_item
#
# This function updates an item in a DynamoDB table.
#
# Parameters:
#     -n table_name -- The name of the table.
#     -k keys -- Path to json file containing the keys that identify the item to
#     update.
#     -e update expression -- An expression that defines one or more attributes
#     to be updated.
#     -v values -- Path to json file containing the update values.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_update_item() {
    local table_name keys update_expression values response
    local option OPTARG # Required to use getopt command in a function.

    #####
    # Function usage explanation
```

```
#####  
function usage() {  
    echo "function dynamodb_update_item"  
    echo "Update an item in a DynamoDB table."  
    echo " -n table_name  -- The name of the table."  
    echo " -k keys      -- Path to json file containing the keys that identify the item  
to update."  
    echo " -e update expression  -- An expression that defines one or more  
attributes to be updated."  
    echo " -v values    -- Path to json file containing the update values."  
    echo ""  
}  
  
while getopts "n:k:e:v:h" option; do  
    case "${option}" in  
        n) table_name="${OPTARG}" ;;  
        k) keys="${OPTARG}" ;;  
        e) update_expression="${OPTARG}" ;;  
        v) values="${OPTARG}" ;;  
        h)  
            usage  
            return 0  
            ;;  
        \?)  
            echo "Invalid parameter"  
            usage  
            return 1  
            ;;  
    esac  
done  
export OPTIND=1  
  
if [[ -z "$table_name" ]]; then  
    errecho "ERROR: You must provide a table name with the -n parameter."  
    usage  
    return 1  
fi  
  
if [[ -z "$keys" ]]; then  
    errecho "ERROR: You must provide a keys json file path the -k parameter."  
    usage  
    return 1  
fi  
if [[ -z "$update_expression" ]]; then
```

```

    errecho "ERROR: You must provide an update expression with the -e parameter."
    usage
    return 1
fi

if [[ -z "$values" ]]; then
    errecho "ERROR: You must provide a values json file path the -v parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "  table_name:  $table_name"
iecho "  keys:        $keys"
iecho "  update_expression:  $update_expression"
iecho "  values:      $values"

response=$(aws dynamodb update-item \
  --table-name "$table_name" \
  --key file://" $keys" \
  --update-expression "$update_expression" \
  --expression-attribute-values file://" $values")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports update-item operation failed.$response"
    return 1
fi

return 0
}

```

本示例中使用的实用程序函数。

```

#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.

```

```
#####
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-return-codes.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho " One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho " Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho " Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho " Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho " The system environment or configuration was invalid."
  fi
}
```

```
elif [ "$err_code" == 254 ]; then
    errecho " The service returned an error."
elif [ "$err_code" == 255 ]; then
    errecho " 255 is a catch-all error."
fi

return 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateItem](#)中的。

## 场景

### 开始使用表、项目和查询

以下代码示例展示了如何：

- 创建可保存电影数据的表。
- 在表中加入单一电影，获取并更新此电影。
- 向 JSON 示例文件的表中写入电影数据。
- 查询在给定年份发行的电影。
- 扫描在年份范围内发行的电影。
- 删除表中的电影后再删除表。

## AWS CLI 使用 Bash 脚本

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

### DynamoDB 入门场景。

```
#####
# function dynamodb_getting_started_movies
#
```

```
# Scenario to create an Amazon DynamoDB table and perform a series of operations on
the table.
#
# Returns:
#     0 - If successful.
#     1 - If an error occurred.
#####
function dynamodb_getting_started_movies() {

    source ./dynamodb_operations.sh

    key_schema_json_file="dynamodb_key_schema.json"
    attribute_definitions_json_file="dynamodb_attr_def.json"
    item_json_file="movie_item.json"
    key_json_file="movie_key.json"
    batch_json_file="batch.json"
    attribute_names_json_file="attribute_names.json"
    attributes_values_json_file="attribute_values.json"

    echo_repeat "*" 88
    echo
    echo "Welcome to the Amazon DynamoDB getting started demo."
    echo
    echo_repeat "*" 88
    echo

    local table_name
    echo -n "Enter a name for a new DynamoDB table: "
    get_input
    table_name=$get_input_result

    local provisioned_throughput="ReadCapacityUnits=5,WriteCapacityUnits=5"

    echo '[
{"AttributeName": "year", "KeyType": "HASH"},
{"AttributeName": "title", "KeyType": "RANGE"}
]' >"$key_schema_json_file"

    echo '[
{"AttributeName": "year", "AttributeType": "N"},
{"AttributeName": "title", "AttributeType": "S"}
]' >"$attribute_definitions_json_file"

    if dynamodb_create_table -n "$table_name" -a "$attribute_definitions_json_file" \
```



```
-k "$key_schema_json_file" -p "$provisioned_throughput" 1>/dev/null; then
echo "Created a DynamoDB table named $table_name"
else
errecho "The table failed to create. This demo will exit."
clean_up
return 1
fi

echo "Waiting for the table to become active...."

if dynamodb_wait_table_active -n "$table_name"; then
echo "The table is now active."
else
errecho "The table failed to become active. This demo will exit."
cleanup "$table_name"
return 1
fi

echo
echo_repeat "*" 88
echo

echo -n "Enter the title of a movie you want to add to the table: "
get_input
local added_title
added_title=$get_input_result

local added_year
get_int_input "What year was it released? "
added_year=$get_input_result

local rating
get_float_input "On a scale of 1 - 10, how do you rate it? " "1" "10"
rating=$get_input_result

local plot
echo -n "Summarize the plot for me: "
get_input
plot=$get_input_result

echo '{
  "year": {"N" : ""$added_year""},
  "title": {"S" : ""$added_title""},
  "info": {"M" : {"plot": {"S" : ""$plot""}, "rating": {"N" : ""$rating""} } }
```

```
}' >"$item_json_file"

if dynamodb_put_item -n "$table_name" -i "$item_json_file"; then
  echo "The movie '$added_title' was successfully added to the table
'$table_name'."
else
  errecho "Put item failed. This demo will exit."
  clean_up "$table_name"
  return 1
fi

echo
echo_repeat "*" 88
echo

echo "Let's update your movie '$added_title'."
get_float_input "You rated it $rating, what new rating would you give it? " "1"
"10"
rating=$get_input_result

echo -n "You summarized the plot as '$plot'."
echo "What would you say now? "
get_input
plot=$get_input_result

echo '{
  "year": {"N" : ""$added_year""},
  "title": {"S" : ""$added_title""}
}' >"$key_json_file"

echo '{
  ":r": {"N" : ""$rating""},
  ":p": {"S" : ""$plot""}
}' >"$item_json_file"

local update_expression="SET info.rating = :r, info.plot = :p"

if dynamodb_update_item -n "$table_name" -k "$key_json_file" -e
"$update_expression" -v "$item_json_file"; then
  echo "Updated '$added_title' with new attributes."
else
  errecho "Update item failed. This demo will exit."
  clean_up "$table_name"
  return 1
fi
```

```
fi

echo
echo_repeat "*" 88
echo

echo "We will now use batch write to upload 150 movie entries into the table."

local batch_json
for batch_json in movie_files/movies_*.json; do
    echo "{ \"\$table_name\" : $(<"$batch_json") }" >"$batch_json_file"
    if dynamodb_batch_write_item -i "$batch_json_file" 1>/dev/null; then
        echo "Entries in $batch_json added to table."
    else
        errecho "Batch write failed. This demo will exit."
        clean_up "$table_name"
        return 1
    fi
done

local title="The Lord of the Rings: The Fellowship of the Ring"
local year="2001"

if get_yes_no_input "Let's move on...do you want to get info about '$title'? (y/n)"; then
    echo '{
"year": {"N" : ""$year""},
"title": {"S" : ""$title""}
}' >"$key_json_file"
    local info
    info=$(dynamodb_get_item -n "$table_name" -k "$key_json_file")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "Get item failed. This demo will exit."
        clean_up "$table_name"
        return 1
    fi

    echo "Here is what I found:"
    echo "$info"
fi

local ask_for_year=true
```

```
while [[ "$ask_for_year" == true ]]; do
    echo "Let's get a list of movies released in a given year."
    get_int_input "Enter a year between 1972 and 2018: " "1972" "2018"
    year=$get_input_result
    echo '{
"#n": "year"
}' >"$attribute_names_json_file"

    echo '{
":v": {"N" : ""$year""}
}' >"$attributes_values_json_file"

    response=$(dynamodb_query -n "$table_name" -k "#n=:v" -a
"$attribute_names_json_file" -v "$attributes_values_json_file")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "Query table failed. This demo will exit."
        clean_up "$table_name"
        return 1
    fi

    echo "Here is what I found:"
    echo "$response"

    if ! get_yes_no_input "Try another year? (y/n) "; then
        ask_for_year=false
    fi
done

echo "Now let's scan for movies released in a range of years. Enter a year: "
get_int_input "Enter a year between 1972 and 2018: " "1972" "2018"
local start=$get_input_result

get_int_input "Enter another year: " "1972" "2018"
local end=$get_input_result

echo '{
"#n": "year"
}' >"$attribute_names_json_file"

echo '{
":v1": {"N" : ""$start""},
":v2": {"N" : ""$end""}
```

```
    }' >"$attributes_values_json_file"

response=$(dynamodb_scan -n "$table_name" -f "#n BETWEEN :v1 AND :v2" -a
"$attribute_names_json_file" -v "$attributes_values_json_file")

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    errecho "Scan table failed. This demo will exit."
    clean_up "$table_name"
    return 1
fi

echo "Here is what I found:"
echo "$response"

echo
echo_repeat "*" 88
echo

echo "Let's remove your movie '$added_title' from the table."

if get_yes_no_input "Do you want to remove '$added_title'? (y/n) "; then
    echo '{
"year": {"N" : ""'$added_year'""},
"title": {"S" : ""'$added_title'""}
}' >"$key_json_file"

    if ! dynamodb_delete_item -n "$table_name" -k "$key_json_file"; then
        errecho "Delete item failed. This demo will exit."
        clean_up "$table_name"
        return 1
    fi
fi

if get_yes_no_input "Do you want to delete the table '$table_name'? (y/n) "; then
    if ! clean_up "$table_name"; then
        return 1
    fi
else
    if ! clean_up; then
        return 1
    fi
fi
fi
```

```

return 0
}

```

此场景中使用的 DynamoDB 函数。

```

#####
# function dynamodb_create_table
#
# This function creates an Amazon DynamoDB table.
#
# Parameters:
#   -n table_name -- The name of the table to create.
#   -a attribute_definitions -- JSON file path of a list of attributes and their
#   types.
#   -k key_schema -- JSON file path of a list of attributes and their key types.
#   -p provisioned_throughput -- Provisioned throughput settings for the table.
#
# Returns:
#   0 - If successful.
#   1 - If it fails.
#####
function dynamodb_create_table() {
    local table_name attribute_definitions key_schema provisioned_throughput response
    local option OPTARG # Required to use getopt command in a function.

    #####
    # Function usage explanation
    #####
    function usage() {
        echo "function dynamodb_create_table"
        echo "Creates an Amazon DynamoDB table."
        echo " -n table_name -- The name of the table to create."
        echo " -a attribute_definitions -- JSON file path of a list of attributes and
their types."
        echo " -k key_schema -- JSON file path of a list of attributes and their key
types."
        echo " -p provisioned_throughput -- Provisioned throughput settings for the
table."
        echo ""
    }

    # Retrieve the calling parameters.

```

```
while getopts "n:a:k:p:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    a) attribute_definitions="${OPTARG}" ;;
    k) key_schema="${OPTARG}" ;;
    p) provisioned_throughput="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$attribute_definitions" ]]; then
  errecho "ERROR: You must provide an attribute definitions json file path the -a
parameter."
  usage
  return 1
fi

if [[ -z "$key_schema" ]]; then
  errecho "ERROR: You must provide a key schema json file path the -k parameter."
  usage
  return 1
fi

if [[ -z "$provisioned_throughput" ]]; then
  errecho "ERROR: You must provide a provisioned throughput json file path the -p
parameter."
  usage
  return 1
fi
```

```

iecho "Parameters:\n"
iecho "  table_name:  $table_name"
iecho "  attribute_definitions:  $attribute_definitions"
iecho "  key_schema:  $key_schema"
iecho "  provisioned_throughput:  $provisioned_throughput"
iecho ""

response=$(aws dynamodb create-table \
  --table-name "$table_name" \
  --attribute-definitions file://"$attribute_definitions" \
  --key-schema file://"$key_schema" \
  --provisioned-throughput "$provisioned_throughput")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-table operation failed.$response"
  return 1
fi

return 0
}

#####
# function dynamodb_describe_table
#
# This function returns the status of a DynamoDB table.
#
# Parameters:
#   -n table_name  -- The name of the table.
#
# Response:
#   - TableStatus:
#   And:
#   0 - Table is active.
#   1 - If it fails.
#####
function dynamodb_describe_table {
  local table_name
  local option OPTARG # Required to use getopt command in a function.

#####

```



```
# Function usage explanation
#####
function usage() {
    echo "function dynamodb_describe_table"
    echo "Describe the status of a DynamoDB table."
    echo "  -n table_name  -- The name of the table."
    echo ""
}

# Retrieve the calling parameters.
while getopts "n:h" option; do
    case "${option}" in
        n) table_name="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
fi

local table_status
table_status=$(
    aws dynamodb describe-table \
        --table-name "$table_name" \
        --output text \
        --query 'Table.TableStatus'
)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log "$error_code"
```

```

        errecho "ERROR: AWS reports describe-table operation failed.$table_status"
        return 1
    fi

    echo "$table_status"

    return 0
}

#####
# function dynamodb_put_item
#
# This function puts an item into a DynamoDB table.
#
# Parameters:
#     -n table_name -- The name of the table.
#     -i item -- Path to json file containing the item values.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_put_item() {
    local table_name item response
    local option OPTARG # Required to use getopt command in a function.

    #####
    # Function usage explanation
    #####
    function usage() {
        echo "function dynamodb_put_item"
        echo "Put an item into a DynamoDB table."
        echo " -n table_name -- The name of the table."
        echo " -i item -- Path to json file containing the item values."
        echo ""
    }

    while getopt "n:i:h" option; do
        case "${option}" in
            n) table_name="${OPTARG}" ;;
            i) item="${OPTARG}" ;;
            h)
                usage
                return 0
        esac
    done
}

```

```
    ;;
    \?)
    echo "Invalid parameter"
    usage
    return 1
    ;;
esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "  table_name:  $table_name"
iecho "  item:       $item"
iecho ""
iecho ""

response=$(aws dynamodb put-item \
  --table-name "$table_name" \
  --item file://"${item}")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports put-item operation failed.$response"
    return 1
fi

return 0
}
```

```
#####
# function dynamodb_update_item
#
# This function updates an item in a DynamoDB table.
#
#
# Parameters:
#     -n table_name -- The name of the table.
#     -k keys -- Path to json file containing the keys that identify the item to
# update.
#     -e update expression -- An expression that defines one or more attributes
# to be updated.
#     -v values -- Path to json file containing the update values.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_update_item() {
    local table_name keys update_expression values response
    local option OPTARG # Required to use getopt command in a function.

    #####
    # Function usage explanation
    #####
    function usage() {
        echo "function dynamodb_update_item"
        echo "Update an item in a DynamoDB table."
        echo " -n table_name -- The name of the table."
        echo " -k keys -- Path to json file containing the keys that identify the item
to update."
        echo " -e update expression -- An expression that defines one or more
attributes to be updated."
        echo " -v values -- Path to json file containing the update values."
        echo ""
    }

    while getopt "n:k:e:v:h" option; do
        case "${option}" in
            n) table_name="${OPTARG}" ;;
            k) keys="${OPTARG}" ;;
            e) update_expression="${OPTARG}" ;;
            v) values="${OPTARG}" ;;
            h)

```

```
        usage
        return 0
        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
fi

if [[ -z "$update_expression" ]]; then
    errecho "ERROR: You must provide an update expression with the -e parameter."
    usage
    return 1
fi

if [[ -z "$values" ]]; then
    errecho "ERROR: You must provide a values json file path the -v parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "  table_name:  $table_name"
iecho "  keys:        $keys"
iecho "  update_expression:  $update_expression"
iecho "  values:     $values"

response=$(aws dynamodb update-item \
  --table-name "$table_name" \
  --key file://" $keys \
```

```

--update-expression "$update_expression" \
--expression-attribute-values file://"values")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports update-item operation failed.$response"
    return 1
fi

return 0
}

#####
# function dynamodb_batch_write_item
#
# This function writes a batch of items into a DynamoDB table.
#
# Parameters:
#     -i item -- Path to json file containing the items to write.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_batch_write_item() {
    local item response
    local option OPTARG # Required to use getopt command in a function.

    #####
    # Function usage explanation
    #####
    function usage() {
        echo "function dynamodb_batch_write_item"
        echo "Write a batch of items into a DynamoDB table."
        echo " -i item -- Path to json file containing the items to write."
        echo ""
    }
    while getopt "i:h" option; do
        case "${option}" in
            i) item="${OPTARG}" ;;
            h)

```

```

        usage
        return 0
        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done
export OPTIND=1

if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "    table_name:  $table_name"
iecho "    item:        $item"
iecho ""

response=$(aws dynamodb batch-write-item \
    --request-items file://"${item}")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-write-item operation failed.$response"
    return 1
fi

return 0
}

#####
# function dynamodb_get_item
#
# This function gets an item from a DynamoDB table.
#
# Parameters:
#     -n table_name  -- The name of the table.

```

```

#     -k keys  -- Path to json file containing the keys that identify the item to
get.
#     [-q query]  -- Optional JMESPath query expression.
#
# Returns:
#     The item as text output.
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_get_item() {
    local table_name keys query response
    local option OPTARG # Required to use getopt command in a function.

# #####
# Function usage explanation
#####
function usage() {
    echo "function dynamodb_get_item"
    echo "Get an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the item
to get."
    echo " [-q query]  -- Optional JMESPath query expression."
    echo ""
}
query=""
while getopt "n:k:q:h" option; do
    case "${option}" in
        n) table_name="${OPTARG}" ;;
        k) keys="${OPTARG}" ;;
        q) query="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

```



```
if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
fi

if [[ -n "$query" ]]; then
    response=$(aws dynamodb get-item \
        --table-name "$table_name" \
        --key file://" $keys" \
        --output text \
        --query "$query")
else
    response=$(
        aws dynamodb get-item \
            --table-name "$table_name" \
            --key file://" $keys" \
            --output text
    )
fi

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports get-item operation failed.$response"
    return 1
fi

if [[ -n "$query" ]]; then
    echo "$response" | sed "/^\t/s/\t//1" # Remove initial tab that the JMSEPath
query inserts on some strings.
else
    echo "$response"
fi

return 0
```

```

}

#####
# function dynamodb_query
#
# This function queries a DynamoDB table.
#
# Parameters:
#   -n table_name -- The name of the table.
#   -k key_condition_expression -- The key condition expression.
#   -a attribute_names -- Path to JSON file containing the attribute names.
#   -v attribute_values -- Path to JSON file containing the attribute values.
#   [-p projection_expression] -- Optional projection expression.
#
# Returns:
#   The items as json output.
# And:
#   0 - If successful.
#   1 - If it fails.
#####
function dynamodb_query() {
    local table_name key_condition_expression attribute_names attribute_values
    projection_expression response
    local option OPTARG # Required to use getopt command in a function.

    # #####
    # Function usage explanation
    # #####
    function usage() {
        echo "function dynamodb_query"
        echo "Query a DynamoDB table."
        echo " -n table_name -- The name of the table."
        echo " -k key_condition_expression -- The key condition expression."
        echo " -a attribute_names -- Path to JSON file containing the attribute names."
        echo " -v attribute_values -- Path to JSON file containing the attribute
values."
        echo " [-p projection_expression] -- Optional projection expression."
        echo ""
    }

    while getopt "n:k:a:v:p:h" option; do
        case "${option}" in
            n) table_name="${OPTARG}" ;;
            k) key_condition_expression="${OPTARG}" ;;

```

```
a) attribute_names="${OPTARG}" ;;
v) attribute_values="${OPTARG}" ;;
p) projection_expression="${OPTARG}" ;;
h)
    usage
    return 0
    ;;
\?)
    echo "Invalid parameter"
    usage
    return 1
    ;;
esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$key_condition_expression" ]]; then
    errecho "ERROR: You must provide a key condition expression with the -k
parameter."
    usage
    return 1
fi

if [[ -z "$attribute_names" ]]; then
    errecho "ERROR: You must provide a attribute names with the -a parameter."
    usage
    return 1
fi

if [[ -z "$attribute_values" ]]; then
    errecho "ERROR: You must provide a attribute values with the -v parameter."
    usage
    return 1
fi

if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb query \
        --table-name "$table_name" \
```

```

    --key-condition-expression "$key_condition_expression" \
    --expression-attribute-names file://"$attribute_names" \
    --expression-attribute-values file://"$attribute_values")
else
    response=$(aws dynamodb query \
        --table-name "$table_name" \
        --key-condition-expression "$key_condition_expression" \
        --expression-attribute-names file://"$attribute_names" \
        --expression-attribute-values file://"$attribute_values" \
        --projection-expression "$projection_expression")
fi

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports query operation failed.$response"
    return 1
fi

echo "$response"

return 0
}

#####
# function dynamodb_scan
#
# This function scans a DynamoDB table.
#
# Parameters:
#     -n table_name -- The name of the table.
#     -f filter_expression -- The filter expression.
#     -a expression_attribute_names -- Path to JSON file containing the expression
#     attribute names.
#     -v expression_attribute_values -- Path to JSON file containing the
#     expression attribute values.
#     [-p projection_expression] -- Optional projection expression.
#
# Returns:
#     The items as json output.
# And:
#     0 - If successful.
#     1 - If it fails.

```

```
#####
function dynamodb_scan() {
    local table_name filter_expression expression_attribute_names
    expression_attribute_values projection_expression response
    local option OPTARG # Required to use getopt command in a function.

    # #####
    # Function usage explanation
    #####
    function usage() {
        echo "function dynamodb_scan"
        echo "Scan a DynamoDB table."
        echo " -n table_name -- The name of the table."
        echo " -f filter_expression -- The filter expression."
        echo " -a expression_attribute_names -- Path to JSON file containing the
expression attribute names."
        echo " -v expression_attribute_values -- Path to JSON file containing the
expression attribute values."
        echo " [-p projection_expression] -- Optional projection expression."
        echo ""
    }

    while getopt "n:f:a:v:p:h" option; do
        case "${option}" in
            n) table_name="${OPTARG}" ;;
            f) filter_expression="${OPTARG}" ;;
            a) expression_attribute_names="${OPTARG}" ;;
            v) expression_attribute_values="${OPTARG}" ;;
            p) projection_expression="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$table_name" ]]; then
        errecho "ERROR: You must provide a table name with the -n parameter."
    fi
}
#####
```

```
usage
return 1
fi

if [[ -z "$filter_expression" ]]; then
    errecho "ERROR: You must provide a filter expression with the -f parameter."
    usage
    return 1
fi

if [[ -z "$expression_attribute_names" ]]; then
    errecho "ERROR: You must provide expression attribute names with the -a
parameter."
    usage
    return 1
fi

if [[ -z "$expression_attribute_values" ]]; then
    errecho "ERROR: You must provide expression attribute values with the -v
parameter."
    usage
    return 1
fi

if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb scan \
        --table-name "$table_name" \
        --filter-expression "$filter_expression" \
        --expression-attribute-names file://"${expression_attribute_names}" \
        --expression-attribute-values file://"${expression_attribute_values}")
else
    response=$(aws dynamodb scan \
        --table-name "$table_name" \
        --filter-expression "$filter_expression" \
        --expression-attribute-names file://"${expression_attribute_names}" \
        --expression-attribute-values file://"${expression_attribute_values}" \
        --projection-expression "$projection_expression")
fi

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports scan operation failed.$response"
```

```

    return 1
fi

echo "$response"

return 0
}

#####
# function dynamodb_delete_item
#
# This function deletes an item from a DynamoDB table.
#
# Parameters:
#   -n table_name  -- The name of the table.
#   -k keys        -- Path to json file containing the keys that identify the item to
delete.
#
# Returns:
#   0 - If successful.
#   1 - If it fails.
#####
function dynamodb_delete_item() {
    local table_name keys response
    local option OPTARG # Required to use getopt command in a function.

    # #####
    # Function usage explanation
    #####
    function usage() {
        echo "function dynamodb_delete_item"
        echo "Delete an item from a DynamoDB table."
        echo " -n table_name  -- The name of the table."
        echo " -k keys        -- Path to json file containing the keys that identify the item
to delete."
        echo ""
    }
    while getopt "n:k:h" option; do
        case "${option}" in
            n) table_name="${OPTARG}" ;;
            k) keys="${OPTARG}" ;;
            h)
                usage
                return 0
        esac
    done
}

```

```

        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "  table_name:  $table_name"
iecho "  keys:       $keys"
iecho ""

response=$(aws dynamodb delete-item \
  --table-name "$table_name" \
  --key file://" $keys")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-item operation failed.$response"
    return 1
fi

return 0
}

#####

```



```

# function dynamodb_delete_table
#
# This function deletes a DynamoDB table.
#
# Parameters:
#     -n table_name  -- The name of the table to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function dynamodb_delete_table() {
    local table_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function dynamodb_delete_table"
        echo "Deletes an Amazon DynamoDB table."
        echo " -n table_name  -- The name of the table to delete."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:h" option; do
        case "${option}" in
            n) table_name="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$table_name" ]]; then
        errecho "ERROR: You must provide a table name with the -n parameter."
        usage
        return 1
    fi
}

```

```

fi

iecho "Parameters:\n"
iecho "    table_name:  $table_name"
iecho ""

response=$(aws dynamodb delete-table \
  --table-name "$table_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-table operation failed.$response"
  return 1
fi

return 0
}

```

此场景中使用的实用程序函数。

```

#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
  printf "%s\n" "$*" 1>&2
}

```

```
#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi

    return 0
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。
  - [BatchWriteItem](#)
  - [CreateTable](#)

- [DeleteItem](#)
- [DeleteTable](#)
- [DescribeTable](#)
- [GetItem](#)
- [PutItem](#)
- [查询](#)
- [扫描](#)
- [UpdateItem](#)

## HealthImaging AWS CLI 与 Bash 脚本一起使用的示例

以下代码示例向您展示了如何使用 with Bash 脚本来执行操作和实现常见场景。AWS Command Line Interface HealthImaging

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

### CreateDatastore

以下代码示例演示了如何使用 CreateDatastore。

AWS CLI 使用 Bash 脚本

```
#####  
# function errecho  
#  
# This function outputs everything sent to it to STDERR (standard error output).  
#####
```

```

function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function imaging_create_datastore
#
# This function creates an AWS HealthImaging data store for importing DICOM P10
# files.
#
# Parameters:
#     -n data_store_name - The name of the data store.
#
# Returns:
#     The datastore ID.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function imaging_create_datastore() {
    local datastore_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function imaging_create_datastore"
        echo "Creates an AWS HealthImaging data store for importing DICOM P10 files."
        echo "  -n data_store_name - The name of the data store."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:h" option; do
        case "${option}" in
            n) datastore_name="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
}

```

```
    esac
done
export OPTIND=1

if [[ -z "$datastore_name" ]]; then
    errecho "ERROR: You must provide a data store name with the -n parameter."
    usage
    return 1
fi

response=$(aws medical-imaging create-datastore \
    --datastore-name "$datastore_name" \
    --output text \
    --query 'datastoreId')

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports medical-imaging create-datastore operation failed.
$response"
    return 1
fi

echo "$response"

return 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDatastore](#)中的。

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

## DeleteDatastore

以下代码示例演示了如何使用 DeleteDatastore。

## AWS CLI 使用 Bash 脚本

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function imaging_delete_datastore
#
# This function deletes an AWS HealthImaging data store.
#
# Parameters:
#     -i datastore_id - The ID of the data store.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function imaging_delete_datastore() {
    local datastore_id response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function imaging_delete_datastore"
        echo "Deletes an AWS HealthImaging data store."
        echo "  -i datastore_id - The ID of the data store."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopts "i:h" option; do
        case "${option}" in
            i) datastore_id="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)

```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done
export OPTIND=1

if [[ -z "$datastore_id" ]]; then
    errecho "ERROR: You must provide a data store ID with the -i parameter."
    usage
    return 1
fi

response=$(aws medical-imaging delete-datastore \
    --datastore-id "$datastore_id")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports medical-imaging delete-datastore operation failed.
$response"
    return 1
fi

return 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteDatastore](#)中的。

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

## GetDatastore

以下代码示例演示了如何使用 GetDatastore。



## AWS CLI 使用 Bash 脚本

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function imaging_get_datastore
#
# Get a data store's properties.
#
# Parameters:
#     -i data_store_id - The ID of the data store.
#
# Returns:
#     [datastore_name, datastore_id, datastore_status, datastore_arn, created_at,
#     updated_at]
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function imaging_get_datastore() {
    local datastore_id option OPTARG # Required to use getopt command in a function.
    local error_code
    # bashsupport disable=BP5008
    function usage() {
        echo "function imaging_get_datastore"
        echo "Gets a data store's properties."
        echo "  -i datastore_id - The ID of the data store."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "i:h" option; do
        case "${option}" in
            i) datastore_id="${OPTARG}" ;;
            h)
                usage
                return 0
        esac
    done
}
```

```
    ;;
    \?)
    echo "Invalid parameter"
    usage
    return 1
    ;;
esac
done
export OPTIND=1

if [[ -z "$datastore_id" ]]; then
    errecho "ERROR: You must provide a data store ID with the -i parameter."
    usage
    return 1
fi

local response

response=$(
    aws medical-imaging get-datastore \
        --datastore-id "$datastore_id" \
        --output text \
        --query "[ datastoreProperties.datastoreName,
datastoreProperties.datastoreId, datastoreProperties.datastoreStatus,
datastoreProperties.datastoreArn, datastoreProperties.createdAt,
datastoreProperties.updatedAt]"
)
error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-datastores operation failed.$response"
    return 1
fi

echo "$response"

return 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetDatastore](#)中的。

**Note**

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

## ListDatastores

以下代码示例演示了如何使用 ListDatastores。

### AWS CLI 使用 Bash 脚本

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function imaging_list_datastores
#
# List the HealthImaging data stores in the account.
#
# Returns:
#     [[datastore_name, datastore_id, datastore_status]]
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function imaging_list_datastores() {
    local option OPTARG # Required to use getopt command in a function.
    local error_code
    # bashsupport disable=BP5008
    function usage() {
        echo "function imaging_list_datastores"
        echo "Lists the AWS HealthImaging data stores in the account."
        echo ""
    }

    # Retrieve the calling parameters.
```

```
while getopts "h" option; do
  case "${option}" in
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

local response
response=$(aws medical-imaging list-datastores \
  --output text \
  --query "datastoreSummaries[*][datastoreName, datastoreId, datastoreStatus]")
error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports list-datastores operation failed.$response"
  return 1
fi

echo "$response"

return 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListDatastores](#)中的。

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

## AWS CLI 与 Bash 脚本一起使用的 IAM 示例

以下代码示例向您展示了如何使用 AWS Command Line Interface 带有 IAM 的 with Bash 脚本来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)
- [场景](#)

### 操作

#### AttachRolePolicy

以下代码示例演示了如何使用 AttachRolePolicy。

#### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####  
# function errecho  
#  
# This function outputs everything sent to it to STDERR (standard error output).  
#####  
function errecho() {  
    printf "%s\n" "$*" 1>&2  
}  
  
#####
```

```
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#     -p policy_arn -- The IAM policy document ARN..
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_attach_role_policy() {
    local role_name policy_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_attach_role_policy"
        echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
        echo " -n role_name    The name of the IAM role."
        echo " -p policy_arn -- The IAM policy document ARN."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            p) policy_arn="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1
}
```

```
if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
fi

response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
fi

echo "$response"

return 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AttachRolePolicy](#)中的。

## CreateAccessKey

以下代码示例演示了如何使用 CreateAccessKey。

AWS CLI 使用 Bash 脚本

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#     -u user_name -- The name of the IAM user.
#     [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#     [access_key_id access_key_secret]
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_user_access_key() {
    local user_name file_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_user_access_key"
        echo "Creates an AWS Identity and Access Management (IAM) key pair."
        echo "  -u user_name  The name of the IAM user."
        echo "  [-f file_name]  Optional file name for the access key output."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "u:f:h" option; do
        case "${option}" in
            u) user_name="${OPTARG}" ;;
            f) file_name="${OPTARG}" ;;
            h)

```



```
        usage
        return 0
        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
fi

response=$(aws iam create-access-key \
    --user-name "$user_name" \
    --output text)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
fi

if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
fi

local key_id key_secret
# shellcheck disable=SC2086
key_id=$(echo $response | cut -f 2 -d ' ')
# shellcheck disable=SC2086
key_secret=$(echo $response | cut -f 4 -d ' ')

echo "$key_id $key_secret"

return 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateAccessKey](#)中的。

## CreatePolicy

以下代码示例演示了如何使用 CreatePolicy。

### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#     -n policy_name -- The name of the IAM policy.
#     -p policy_json -- The policy document.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_policy() {
    local policy_name policy_document response
    local option OPTARG # Required to use getopt command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name  The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
    case "${option}" in
        n) policy_name="${OPTARG}" ;;
        p) policy_document="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$policy_name" ]]; then
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
fi

response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)
```

```

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
fi

echo "$response"
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreatePolicy](#)中的。

## CreateRole

以下代码示例演示了如何使用 CreateRole。

### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:

```

```
# -n role_name -- The name of the IAM role.
# -p policy_json -- The assume role policy document.
#
# Returns:
#   The ARN of the role.
# And:
#   0 - If successful.
#   1 - If it fails.
#####
function iam_create_role() {
    local role_name policy_document response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_user_access_key"
        echo "Creates an AWS Identity and Access Management (IAM) role."
        echo " -n role_name The name of the IAM role."
        echo " -p policy_json -- The assume role policy document."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            p) policy_document="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$role_name" ]]; then
        errecho "ERROR: You must provide a role name with the -n parameter."
        usage
        return 1
    fi
}
```

```
fi

if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
fi

response=$(aws iam create-role \
    --role-name "$role_name" \
    --assume-role-policy-document "$policy_document" \
    --output text \
    --query Role.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
fi

echo "$response"

return 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateRole](#)中的。

## CreateUser

以下代码示例演示了如何使用 CreateUser。

### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#     -u user_name -- The name of the user to create.
#
# Returns:
#     The ARN of the user.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_user() {
    local user_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_user"
```

```
    echo "Creates an WS Identity and Access Management (IAM) user. You must supply a
username:"
    echo "  -u user_name    The name of the user. It must be unique within the
account."
    echo ""
}

# Retrieve the calling parameters.
while getopts "u:h" option; do
    case "${option}" in
        u) user_name="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "  User name:  $user_name"
iecho ""

# If the user already exists, we don't want to try to create it.
if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
fi

response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')
```



```

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
fi

echo "$response"

return 0
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateUser](#)中的。

## DeleteAccessKey

以下代码示例演示了如何使用 DeleteAccessKey。

AWS CLI 使用 Bash 脚本

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#

```

```
# Parameters:
#     -u user_name  -- The name of the user.
#     -k access_key -- The access key to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_access_key() {
    local user_name access_key response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_access_key"
        echo "Deletes an WS Identity and Access Management (IAM) access key for the
specified IAM user"
        echo "  -u user_name    The name of the user."
        echo "  -k access_key    The access key to delete."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "u:k:h" option; do
        case "${option}" in
            u) user_name="${OPTARG}" ;;
            k) access_key="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$user_name" ]]; then
        errecho "ERROR: You must provide a username with the -u parameter."
        usage
        return 1
    fi
}
```

```
fi

if [[ -z "$access_key" ]]; then
    errecho "ERROR: You must provide an access key with the -k parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "    Username:  $user_name"
iecho "    Access key:  $access_key"
iecho ""

response=$(aws iam delete-access-key \
    --user-name "$user_name" \
    --access-key-id "$access_key")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
fi

iecho "delete-access-key response:$response"
iecho

return 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteAccessKey](#)中的。

## DeletePolicy

以下代码示例演示了如何使用 DeletePolicy。

## AWS CLI 使用 Bash 脚本

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#     -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_policy() {
```

```
local policy_arn response
local option OPTARG # Required to use getopt command in a function.

# bashsupport disable=BP5008
function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an WS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
}

# Retrieve the calling parameters.
while getopt "n:h" option; do
    case "${option}" in
        n) policy_arn="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "  Policy arn: $policy_arn"
iecho ""

response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
```

```

aws_cli_error_log $error_code
errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
return 1
fi

iecho "delete-policy response:$response"
iecho

return 0
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeletePolicy](#)中的。

## DeleteRole

以下代码示例演示了如何使用 DeleteRole。

AWS CLI 使用 Bash 脚本

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).

```

```
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_role() {
    local role_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_role"
        echo "Deletes an WS Identity and Access Management (IAM) role"
        echo "  -n role_name -- The name of the IAM role."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
done
```

```
export OPTIND=1

echo "role_name:$role_name"
if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "    Role name: $role_name"
iecho ""

response=$(aws iam delete-role \
    --role-name "$role_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
fi

iecho "delete-role response:$response"
iecho

return 0
}
```


- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteRole](#)中的。

## DeleteUser

以下代码示例演示了如何使用 DeleteUser。



## AWS CLI 使用 Bash 脚本

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#     -u user_name -- The name of the user to create.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_user() {
```

```
local user_name response
local option OPTARG # Required to use getopt command in a function.

# bashsupport disable=BP5008
function usage() {
    echo "function iam_delete_user"
    echo "Deletes an WS Identity and Access Management (IAM) user. You must supply a
username:"
    echo "  -u user_name    The name of the user."
    echo ""
}

# Retrieve the calling parameters.
while getopt "u:h" option; do
    case "${option}" in
        u) user_name="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "  User name:  $user_name"
iecho ""

# If the user does not exist, we don't want to try to delete it.
if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
fi
```

```

response=$(aws iam delete-user \
  --user-name "$user_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-user operation failed.$response"
  return 1
fi

iecho "delete-user response:$response"
iecho

return 0
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteUser](#)中的。

## DetachRolePolicy

以下代码示例演示了如何使用 DetachRolePolicy。

AWS CLI 使用 Bash 脚本

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
  printf "%s\n" "$*" 1>&2
}

```

```
#####
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a role.
#
# Parameters:
#   -n role_name -- The name of the IAM role.
#   -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#   0 - If successful.
#   1 - If it fails.
#####
function iam_detach_role_policy() {
    local role_name policy_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_detach_role_policy"
        echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
        echo "  -n role_name  The name of the IAM role."
        echo "  -p policy_ARN -- The IAM policy document ARN."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            p) policy_arn="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1
}
```

```
if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
fi

response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
fi

echo "$response"


return 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DetachRolePolicy](#)中的。

## GetUser

以下代码示例演示了如何使用 GetUser。

## AWS CLI 使用 Bash 脚本

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
# (IAM) user already exists.
#
# Parameters:
#     $1 - The name of the IAM user to check.
#
# Returns:
#     0 - If the user already exists.
#     1 - If the user doesn't exist.
#####
function iam_user_exists() {
    local user_name
    user_name=$1

    # Check whether the IAM user already exists.
    # We suppress all output - we're interested only in the return code.

    local errors
    errors=$(aws iam get-user \
        --user-name "$user_name" 2>&1 >/dev/null)

    local error_code=${?}
```

```

if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
else
    if [[ $errors != *"error"*(NoSuchEntity)* ]]; then
        aws_cli_error_log $error_code
        errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
fi
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetUser](#)中的。

## ListAccessKeys

以下代码示例演示了如何使用 ListAccessKeys。

AWS CLI 使用 Bash 脚本

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function iam_list_access_keys
#
# This function lists the access keys for the specified user.
#

```

```

# Parameters:
#     -u user_name -- The name of the IAM user.
#
# Returns:
#     access_key_ids
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_list_access_keys() {

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_list_access_keys"
        echo "Lists the AWS Identity and Access Management (IAM) access key IDs for the
specified user."
        echo "  -u user_name  The name of the IAM user."
        echo ""
    }

    local user_name response
    local option OPTARG # Required to use getopt command in a function.
    # Retrieve the calling parameters.
    while getopt "u:h" option; do
        case "${option}" in
            u) user_name="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$user_name" ]]; then
        errecho "ERROR: You must provide a username with the -u parameter."
        usage
        return 1
    fi
}

```



```

response=$(aws iam list-access-keys \
  --user-name "$user_name" \
  --output text \
  --query 'AccessKeyMetadata[].AccessKeyId')

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports list-access-keys operation failed.$response"
  return 1
fi

echo "$response"

return 0
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListAccessKeys](#)中的。

## ListUsers

以下代码示例演示了如何使用 ListUsers。

AWS CLI 使用 Bash 脚本

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
  printf "%s\n" "$*" 1>&2
}

```

```
#####
# function iam_list_users
#
# List the IAM users in the account.
#
# Returns:
#     The list of users names
# And:
#     0 - If the user already exists.
#     1 - If the user doesn't exist.
#####
function iam_list_users() {
    local option OPTARG # Required to use getopt command in a function.
    local error_code
    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_list_users"
        echo "Lists the AWS Identity and Access Management (IAM) user in the account."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "h" option; do
        case "${option}" in
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    local response

    response=$(aws iam list-users \
        --output text \
        --query "Users[].UserName")
    error_code=${?}
}
```

```
if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-users operation failed.$response"
    return 1
fi

echo "$response"

return 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListUsers](#)中的。

## 场景

### 创建用户并代入角色

以下代码示例展示了如何创建用户并代入角色。

#### Warning

为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证，而是使用与身份提供商的联合身份验证，例如 [AWS IAM Identity Center](#)。

- 创建没有权限的用户。
- 创建授予列出账户的 Amazon S3 存储桶的权限的角色
- 添加策略以允许用户代入该角色。
- 代入角色并使用临时凭证列出 S3 存储桶，然后清除资源。

## AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function iam_create_user_assume_role
#
# Scenario to create an IAM user, create an IAM role, and apply the role to the
# user.
#
# "IAM access" permissions are needed to run this code.
# "STS assume role" permissions are needed to run this code. (Note: It might be
# necessary to
# create a custom policy).
#
# Returns:
# 0 - If successful.
# 1 - If an error occurred.
#####
function iam_create_user_assume_role() {
    {
        if [ "$IAM_OPERATIONS_SOURCED" != "True" ]; then

            source ./iam_operations.sh
        fi
    }

    echo_repeat "*" 88
    echo "Welcome to the IAM create user and assume role demo."
    echo
    echo "This demo will create an IAM user, create an IAM role, and apply the role to
the user."
    echo_repeat "*" 88
    echo

    echo -n "Enter a name for a new IAM user: "
    get_input
    user_name=$get_input_result

    local user_arn
    user_arn=$(iam_create_user -u "$user_name")

    # shellcheck disable=SC2181
    if [[ ${?} == 0 ]]; then
        echo "Created demo IAM user named $user_name"
    else
        errecho "$user_arn"
    fi
}
#####
```

```
    errecho "The user failed to create. This demo will exit."
    return 1
fi

local access_key_response
access_key_response=$(iam_create_user_access_key -u "$user_name")
# shellcheck disable=SC2181
if [[ ${?} != 0 ]]; then
    errecho "The access key failed to create. This demo will exit."
    clean_up "$user_name"
    return 1
fi

IFS=$'\t ' read -r -a access_key_values <<<"$access_key_response"
local key_name=${access_key_values[0]}
local key_secret=${access_key_values[1]}

echo "Created access key named $key_name"

echo "Wait 10 seconds for the user to be ready."
sleep 10
echo_repeat "*" 88
echo

local iam_role_name
iam_role_name=$(generate_random_name "test-role")
echo "Creating a role named $iam_role_name with user $user_name as the principal."

local assume_role_policy_document="{
  \"Version\": \"2012-10-17\",
  \"Statement\": [{
    \"Effect\": \"Allow\",
    \"Principal\": {\"AWS\": \"$user_arn\"},
    \"Action\": \"sts:AssumeRole\"
  }]
}"

local role_arn
role_arn=$(iam_create_role -n "$iam_role_name" -p "$assume_role_policy_document")

# shellcheck disable=SC2181
if [ ${?} == 0 ]; then
    echo "Created IAM role named $iam_role_name"
else
```

```
errecho "The role failed to create. This demo will exit."
clean_up "$user_name" "$key_name"
return 1
fi

local policy_name
policy_name=$(generate_random_name "test-policy")
local policy_document="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Action\": \"s3:ListAllMyBuckets\",
        \"Resource\": \"arn:aws:s3:::*\"}]}"

local policy_arn
policy_arn=$(iam_create_policy -n "$policy_name" -p "$policy_document")
# shellcheck disable=SC2181
if [[ ${?} == 0 ]]; then
    echo "Created IAM policy named $policy_name"
else
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name"
    return 1
fi

if (iam_attach_role_policy -n "$iam_role_name" -p "$policy_arn"); then
    echo "Attached policy $policy_arn to role $iam_role_name"
else
    errecho "The policy failed to attach."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
    return 1
fi

local assume_role_policy_document="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Action\": \"sts:AssumeRole\",
        \"Resource\": \"$role_arn\"}]}"

local assume_role_policy_name
assume_role_policy_name=$(generate_random_name "test-assume-role-")

# shellcheck disable=SC2181
```

```
local assume_role_policy_arn
assume_role_policy_arn=$(iam_create_policy -n "$assume_role_policy_name" -p
"$assume_role_policy_document")
# shellcheck disable=SC2181
if [ ${?} == 0 ]; then
    echo "Created IAM policy named $assume_role_policy_name for sts assume role"
else
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
    return 1
fi

echo "Wait 10 seconds to give AWS time to propagate these new resources and
connections."
sleep 10
echo_repeat "*" 88
echo

echo "Try to list buckets without the new user assuming the role."
echo_repeat "*" 88
echo

# Set the environment variables for the created user.
# bashsupport disable=BP2001
export AWS_ACCESS_KEY_ID=$key_name
# bashsupport disable=BP2001
export AWS_SECRET_ACCESS_KEY=$key_secret

local buckets
buckets=$(s3_list_buckets)

# shellcheck disable=SC2181
if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. This should not have
happened."
else
    errecho "Because the role with permissions has not been assumed, listing buckets
failed."
fi

echo
echo_repeat "*" 88
```

```
echo "Now assume the role $iam_role_name and list the buckets."
echo_repeat "*" 88
echo

local credentials

credentials=$(sts_assume_role -r "$role_arn" -n "AssumeRoleDemoSession")
# shellcheck disable=SC2181
if [ ${?} == 0 ]; then
    echo "Assumed role $iam_role_name"
else
    errecho "Failed to assume role."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
"$assume_role_policy_arn"
    return 1
fi

IFS=$'\t ' read -r -a credentials <<<"$credentials"

export AWS_ACCESS_KEY_ID=${credentials[0]}
export AWS_SECRET_ACCESS_KEY=${credentials[1]}
# bashsupport disable=BP2001
export AWS_SESSION_TOKEN=${credentials[2]}

buckets=$(s3_list_buckets)

# shellcheck disable=SC2181
if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. Listing buckets succeeded
because of "
    echo "the assumed role."
else
    errecho "Failed to list buckets. This should not happen."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    export AWS_SESSION_TOKEN=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
"$assume_role_policy_arn"
    return 1
fi
```



```

local result=0
export AWS_ACCESS_KEY_ID=""
export AWS_SECRET_ACCESS_KEY=""

echo
echo_repeat "*" 88
echo "The created resources will now be deleted."
echo_repeat "*" 88
echo

clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
"$assume_role_policy_arn"

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    result=1
fi

return $result
}

```

此场景中使用的 IAM 函数。

```

#####
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
# (IAM) user already exists.
#
# Parameters:
#     $1 - The name of the IAM user to check.
#
# Returns:
#     0 - If the user already exists.
#     1 - If the user doesn't exist.
#####
function iam_user_exists() {
    local user_name
    user_name=$1

    # Check whether the IAM user already exists.

```

```

# We suppress all output - we're interested only in the return code.

local errors
errors=$(aws iam get-user \
  --user-name "$user_name" 2>&1 >/dev/null)

local error_code=${?}

if [[ $error_code -eq 0 ]]; then
  return 0 # 0 in Bash script means true.
else
  if [[ $errors != *"error"*(NoSuchEntity)* ]]; then
    aws_cli_error_log $error_code
    errecho "Error calling iam get-user $errors"
  fi

  return 1 # 1 in Bash script means false.
fi
}

#####
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#   -u user_name -- The name of the user to create.
#
# Returns:
#   The ARN of the user.
#   And:
#   0 - If successful.
#   1 - If it fails.
#####
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopt command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an WS Identity and Access Management (IAM) user. You must supply a
username:"

```

```
    echo "  -u user_name    The name of the user. It must be unique within the
account."
    echo ""
}

# Retrieve the calling parameters.
while getopts "u:h" option; do
    case "${option}" in
        u) user_name="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "  User name:  $user_name"
iecho ""

# If the user already exists, we don't want to try to create it.
if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
fi

response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

local error_code=${?}
```

```

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
fi

echo "$response"

return 0
}

#####
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#     -u user_name -- The name of the IAM user.
#     [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#     [access_key_id access_key_secret]
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_user_access_key() {
    local user_name file_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_user_access_key"
        echo "Creates an AWS Identity and Access Management (IAM) key pair."
        echo "  -u user_name    The name of the IAM user."
        echo "  [-f file_name]  Optional file name for the access key output."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "u:f:h" option; do
        case "${option}" in
            u) user_name="${OPTARG}" ;;
            f) file_name="${OPTARG}" ;;
        esac
    done

```

```
h)
  usage
  return 0
  ;;
\?)
  echo "Invalid parameter"
  usage
  return 1
  ;;
esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

response=$(aws iam create-access-key \
  --user-name "$user_name" \
  --output text)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-access-key operation failed.$response"
  return 1
fi

if [[ -n "$file_name" ]]; then
  echo "$response" >"$file_name"
fi

local key_id key_secret
# shellcheck disable=SC2086
key_id=$(echo $response | cut -f 2 -d ' ')
# shellcheck disable=SC2086
key_secret=$(echo $response | cut -f 4 -d ' ')

echo "$key_id $key_secret"

return 0
```

```

}

#####
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#     -p policy_json -- The assume role policy document.
#
# Returns:
#     The ARN of the role.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_role() {
    local role_name policy_document response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_user_access_key"
        echo "Creates an AWS Identity and Access Management (IAM) role."
        echo "  -n role_name  The name of the IAM role."
        echo "  -p policy_json -- The assume role policy document."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            p) policy_document="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
}

```

```
    esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
fi

response=$(aws iam create-role \
    --role-name "$role_name" \
    --assume-role-policy-document "$policy_document" \
    --output text \
    --query Role.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
fi

echo "$response"

return 0
}

#####
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#     -n policy_name -- The name of the IAM policy.
#     -p policy_json -- The policy document.
#
```

```
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_create_policy() {
    local policy_name policy_document response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_create_policy"
        echo "Creates an AWS Identity and Access Management (IAM) policy."
        echo "  -n policy_name  The name of the IAM policy."
        echo "  -p policy_json -- The policy document."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) policy_name="${OPTARG}" ;;
            p) policy_document="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$policy_name" ]]; then
        errecho "ERROR: You must provide a policy name with the -n parameter."
        usage
        return 1
    fi

    if [[ -z "$policy_document" ]]; then
        errecho "ERROR: You must provide a policy document with the -p parameter."
        usage
    fi
}
```



```

    return 1
fi

response=$(aws iam create-policy \
  --policy-name "$policy_name" \
  --policy-document "$policy_document" \
  --output text \
  --query Policy.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-policy operation failed.\n$response"
  return 1
fi

echo "$response"
}

#####
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a role.
#
# Parameters:
#   -n role_name -- The name of the IAM role.
#   -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#   0 - If successful.
#   1 - If it fails.
#####
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopt command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
    echo "  -n role_name  The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
  }
}

```

```
    echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
    case "${option}" in
        n) role_name="${OPTARG}" ;;
        p) policy_arn="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
fi

response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
fi
```

```

    echo "$response"

    return 0
}

#####
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#     -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_detach_role_policy() {
    local role_name policy_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_detach_role_policy"
        echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
        echo "  -n role_name  The name of the IAM role."
        echo "  -p policy_ARN -- The IAM policy document ARN."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:p:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;
            p) policy_arn="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"

```

```

        usage
        return 1
    ;;
    esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
fi

if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
fi

response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
fi

echo "$response"

return 0
}

#####
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#     -n policy_arn -- The name of the IAM policy arn.
#

```

```

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_policy() {
    local policy_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_policy"
        echo "Deletes an WS Identity and Access Management (IAM) policy"
        echo "  -n policy_arn -- The name of the IAM policy arn."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:h" option; do
        case "${option}" in
            n) policy_arn="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$policy_arn" ]]; then
        errecho "ERROR: You must provide a policy arn with the -n parameter."
        usage
        return 1
    fi

    iecho "Parameters:\n"
    iecho "  Policy arn: $policy_arn"
    iecho ""

    response=$(aws iam delete-policy \

```

```

    --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
fi

iecho "delete-policy response:$response"
iecho

return 0
}

#####
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#     -n role_name -- The name of the IAM role.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_role() {
    local role_name response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_role"
        echo "Deletes an WS Identity and Access Management (IAM) role"
        echo "  -n role_name -- The name of the IAM role."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:h" option; do
        case "${option}" in
            n) role_name="${OPTARG}" ;;

```

```

    h)
        usage
        return 0
        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done
export OPTIND=1

echo "role_name:$role_name"
if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "    Role name: $role_name"
iecho ""

response=$(aws iam delete-role \
    --role-name "$role_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
fi

iecho "delete-role response:$response"
iecho

return 0
}

#####
# function iam_delete_access_key
#

```

```

# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#     -u user_name  -- The name of the user.
#     -k access_key -- The access key to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function iam_delete_access_key() {
    local user_name access_key response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function iam_delete_access_key"
        echo "Deletes an WS Identity and Access Management (IAM) access key for the
specified IAM user"
        echo "  -u user_name    The name of the user."
        echo "  -k access_key    The access key to delete."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "u:k:h" option; do
        case "${option}" in
            u) user_name="${OPTARG}" ;;
            k) access_key="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$user_name" ]]; then
        errecho "ERROR: You must provide a username with the -u parameter."
    fi
}

```



```

    usage
    return 1
fi

if [[ -z "$access_key" ]]; then
    errecho "ERROR: You must provide an access key with the -k parameter."
    usage
    return 1
fi

iecho "Parameters:\n"
iecho "    Username:  $user_name"
iecho "    Access key:  $access_key"
iecho ""

response=$(aws iam delete-access-key \
    --user-name "$user_name" \
    --access-key-id "$access_key")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
fi

iecho "delete-access-key response:$response"
iecho

return 0
}

#####
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#     -u user_name  -- The name of the user to create.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.

```

```
#####  
function iam_delete_user() {  
    local user_name response  
    local option OPTARG # Required to use getopt command in a function.  
  
    # bashsupport disable=BP5008  
    function usage() {  
        echo "function iam_delete_user"  
        echo "Deletes an WS Identity and Access Management (IAM) user. You must supply a  
username:"  
        echo "  -u user_name    The name of the user."  
        echo ""  
    }  
  
    # Retrieve the calling parameters.  
    while getopt "u:h" option; do  
        case "${option}" in  
            u) user_name="${OPTARG}" ;;  
            h)  
                usage  
                return 0  
                ;;  
            \?)  
                echo "Invalid parameter"  
                usage  
                return 1  
                ;;  
        esac  
    done  
    export OPTIND=1  
  
    if [[ -z "$user_name" ]]; then  
        errecho "ERROR: You must provide a username with the -u parameter."  
        usage  
        return 1  
    fi  
  
    iecho "Parameters:\n"  
    iecho "  User name:  $user_name"  
    iecho ""  
  
    # If the user does not exist, we don't want to try to delete it.  
    if (! iam_user_exists "$user_name"); then  
        errecho "ERROR: A user with that name does not exist in the account."  
    fi  
}
```

```
    return 1
fi

response=$(aws iam delete-user \
  --user-name "$user_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-user operation failed.$response"
  return 1
fi

iecho "delete-user response:$response"
iecho

return 0
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## 使用 AWS CLI Bash 脚本的 Amazon S3 示例

以下代码示例向您展示了如何在 Amazon S3 中使用 with Bash 脚本来执行操作和实现常见场景。  
AWS Command Line Interface

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)
- [场景](#)

### 操作

#### CopyObject

以下代码示例演示了如何使用 CopyObject。

#### AWS CLI 使用 Bash 脚本

##### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####  
# function errecho  
#  
# This function outputs everything sent to it to STDERR (standard error output).  
#####  
function errecho() {  
    printf "%s\n" "$*" 1>&2  
}
```

```
#####
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file from and to.
#     $2 - The key of the source file to copy.
#     $3 - The key of the destination file.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_item_in_bucket() {
    local bucket_name=$1
    local source_key=$2
    local destination_key=$3
    local response

    response=$(aws s3api copy-object \
        --bucket "$bucket_name" \
        --copy-source "$bucket_name/$source_key" \
        --key "$destination_key")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api copy-object operation failed.\n$response"
        return 1
    fi
}


```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CopyObject](#)中的。

## CreateBucket

以下代码示例演示了如何使用 CreateBucket。

## AWS CLI 使用 Bash 脚本

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#     -b bucket_name -- The name of the bucket to create.
#     -r region_code -- The code for an AWS Region in which to
#                       create the bucket.
#
# Returns:
#     The URL of the bucket that was created.
```

```
# And:
# 0 - If successful.
# 1 - If it fails.
#####
function create_bucket() {
    local bucket_name region_code response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function create_bucket"
        echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
        echo "  -b bucket_name    The name of the bucket. It must be globally unique."
        echo "  [-r region_code]   The code for an AWS Region in which the bucket is
created."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "b:r:h" option; do
        case "${option}" in
            b) bucket_name="${OPTARG}" ;;
            r) region_code="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done

    if [[ -z "$bucket_name" ]]; then
        errecho "ERROR: You must provide a bucket name with the -b parameter."
        usage
        return 1
    fi

    local bucket_config_arg
    # A location constraint for "us-east-1" returns an error.
    if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
```

```
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
fi

iecho "Parameters:\n"
iecho "    Bucket name:  $bucket_name"
iecho "    Region code:  $region_code"
iecho ""

# If the bucket already exists, we don't want to try to create it.
if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
fi

# shellcheck disable=SC2086
response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
fi
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateBucket](#)中的。

## DeleteBucket

以下代码示例演示了如何使用 DeleteBucket。

AWS CLI 使用 Bash 脚本

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。



```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api delete-bucket \
        --bucket "$bucket_name")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
        return 1
    fi
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteBucket](#)中的。

## DeleteObject

以下代码示例演示了如何使用 DeleteObject。

## AWS CLI 使用 Bash 脚本

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_item_in_bucket
#
# This function deletes the specified file from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - The key (file name) in the bucket to delete.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_item_in_bucket() {
    local bucket_name=$1
    local key=$2
    local response

    response=$(aws s3api delete-object \
        --bucket "$bucket_name" \
        --key "$key")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-object operation failed.\n$response"
    fi
}
```

```

    return 1
  fi
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteObject](#)中的。

## DeleteObjects

以下代码示例演示了如何使用 DeleteObjects。

AWS CLI 使用 Bash 脚本

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#####
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#   $1 - The name of the bucket.
#   $2 - A list of keys in the bucket to delete.

# Returns:
#   0 - If successful.
#   1 - If it fails.
#####

```

```
function delete_items_in_bucket() {
    local bucket_name=$1
    local keys=$2
    local response

    # Create the JSON for the items to delete.
    local delete_items
    delete_items="{\"Objects\":["
    for key in $keys; do
        delete_items="$delete_items{\"Key\": \"$key\"},"
    done
    delete_items=${delete_items%?} # Remove the final comma.
    delete_items="$delete_items]"

    response=$(aws s3api delete-objects \
        --bucket "$bucket_name" \
        --delete "$delete_items")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-object operation failed.\n$response"
        return 1
    fi
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteObjects](#)中的。

## GetObject

以下代码示例演示了如何使用 GetObject。

AWS CLI 使用 Bash 脚本

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
```

```

# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#     $1 - The name of the bucket to download the object from.
#     $2 - The path and file name to store the downloaded bucket.
#     $3 - The key (name) of the object in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function download_object_from_bucket() {
    local bucket_name=$1
    local destination_file_name=$2
    local object_name=$3
    local response

    response=$(aws s3api get-object \
        --bucket "$bucket_name" \
        --key "$object_name" \
        "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetObject](#)中的。

## HeadBucket

以下代码示例演示了如何使用 HeadBucket。

### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function bucket_exists
#
# This function checks to see if the specified bucket already exists.
#
# Parameters:
#     $1 - The name of the bucket to check.
#
# Returns:
#     0 - If the bucket already exists.
#     1 - If the bucket doesn't exist.
#####
function bucket_exists() {
    local bucket_name
    bucket_name=$1

    # Check whether the bucket already exists.
    # We suppress all output - we're interested only in the return code.

    if aws s3api head-bucket \
        --bucket "$bucket_name" \
        >/dev/null 2>&1; then
        return 0 # 0 in Bash script means true.
    else
        return 1 # 1 in Bash script means false.
    fi
}
```

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考 [HeadBucket](#) 中的。

## ListObjectsV2

以下代码示例演示了如何使用 ListObjectsV2。

### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#     $1 - The name of the bucket.
#
# Returns:
#     The list of files in text format.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function list_items_in_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api list-objects \
        --bucket "$bucket_name" \
```

```

--output text \
--query 'Contents[].{Key: Key, Size: Size}')
```

```

# shellcheck disable=SC2181
if [[ ${?} -eq 0 ]]; then
    echo "$response"
else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
fi
}
```

- 有关 API 的详细信息，请参阅《AWS CLI 命令参考》中的 [ListObjectsV2](#)。

## PutObject

以下代码示例演示了如何使用 PutObject。

AWS CLI 使用 Bash 脚本

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
```



```

# Parameters:
#     $1 - The name of the bucket to copy the file to.
#     $2 - The path and file name of the local file to copy to the bucket.
#     $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_file_to_bucket() {
    local response bucket_name source_file destination_file_name
    bucket_name=$1
    source_file=$2
    destination_file_name=$3

    response=$(aws s3api put-object \
        --bucket "$bucket_name" \
        --body "$source_file" \
        --key "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}

```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[PutObject](#)中的。

## 场景

### 桶和对象入门

以下代码示例展示了如何：

- 创建桶并将文件上传到其中。
- 从桶中下载对象。
- 将对象复制到存储桶中的子文件夹。
- 列出存储桶中的对象。
- 删除存储桶及其对象。

## AWS CLI 使用 Bash 脚本

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
#
# Returns:
#     0 - If successful.
#     1 - If an error occurred.
#####
function s3_getting_started() {
    {
        if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
            cd bucket-lifecycle-operations || exit

            source ./bucket_operations.sh
            cd ..
        fi
    }

    echo_repeat "*" 88
    echo "Welcome to the Amazon S3 getting started demo."
    echo_repeat "*" 88

    local bucket_name
    bucket_name=$(generate_random_name "doc-example-bucket")

    local region_code
    region_code=$(aws configure get region)

    if create_bucket -b "$bucket_name" -r "$region_code"; then
        echo "Created demo bucket named $bucket_name"
    else
        errecho "The bucket failed to create. This demo will exit."
        return 1
    fi
}
```

```
fi

local file_name
while [ -z "$file_name" ]; do
    echo -n "Enter a file you want to upload to your bucket: "
    get_input
    file_name=$get_input_result

    if [ ! -f "$file_name" ]; then
        echo "Could not find file $file_name. Are you sure it exists?"
        file_name=""
    fi
done

local key
key="$(basename "$file_name")"

local result=0
if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
    echo "Uploaded file $file_name into bucket $bucket_name with key $key."
else
    result=1
fi

local destination_file
destination_file="$file_name.download"
if yes_no_input "Would you like to download $key to the file $destination_file?
(y/n) "; then
    if download_object_from_bucket "$bucket_name" "$destination_file" "$key"; then
        echo "Downloaded $key in the bucket $bucket_name to the file
$destination_file."
    else
        result=1
    fi
fi

if yes_no_input "Would you like to copy $key a new object key in your bucket? (y/
n) "; then
    local to_key
    to_key="demo/$key"
    if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
        echo "Copied $key in the bucket $bucket_name to the $to_key."
    else
        result=1
    fi
fi
```

```

    fi
fi

local bucket_items
bucket_items=$(list_items_in_bucket "$bucket_name")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
    result=1
fi

echo "Your bucket contains the following items."
echo -e "Name\t\tSize"
echo "$bucket_items"

if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
(y/n) "; then
    bucket_items=$(echo "$bucket_items" | cut -f 1)

    if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
        echo "The following items were deleted from the bucket $bucket_name"
        echo "$bucket_items"
    else
        result=1
    fi

    if delete_bucket "$bucket_name"; then
        echo "Deleted the bucket $bucket_name"
    else
        result=1
    fi
fi

return $result
}

```

此场景中使用的 Amazon S3 函数。

```

#####
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless

```

```
# it already exists.
#
# Parameters:
#     -b bucket_name  -- The name of the bucket to create.
#     -r region_code  -- The code for an AWS Region in which to
#                       create the bucket.
#
# Returns:
#     The URL of the bucket that was created.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function create_bucket() {
    local bucket_name region_code response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function create_bucket"
        echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
        echo "  -b bucket_name    The name of the bucket. It must be globally unique."
        echo "  [-r region_code]   The code for an AWS Region in which the bucket is
created."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "b:r:h" option; do
        case "${option}" in
            b) bucket_name="${OPTARG}" ;;
            r) region_code="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
```

```

if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
fi

local bucket_config_arg
# A location constraint for "us-east-1" returns an error.
if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
fi

iecho "Parameters:\n"
iecho "    Bucket name:    $bucket_name"
iecho "    Region code:    $region_code"
iecho ""

# If the bucket already exists, we don't want to try to create it.
if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
fi

# shellcheck disable=SC2086
response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
fi
}

#####
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file to.
#     $2 - The path and file name of the local file to copy to the bucket.

```

```

#     $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_file_to_bucket() {
    local response bucket_name source_file destination_file_name
    bucket_name=$1
    source_file=$2
    destination_file_name=$3

    response=$(aws s3api put-object \
        --bucket "$bucket_name" \
        --body "$source_file" \
        --key "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}

#####
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#     $1 - The name of the bucket to download the object from.
#     $2 - The path and file name to store the downloaded bucket.
#     $3 - The key (name) of the object in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function download_object_from_bucket() {
    local bucket_name=$1
    local destination_file_name=$2
    local object_name=$3
    local response

```

```

response=$(aws s3api get-object \
  --bucket "$bucket_name" \
  --key "$object_name" \
  "$destination_file_name")

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
  errecho "ERROR: AWS reports put-object operation failed.\n$response"
  return 1
fi
}

#####
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#   $1 - The name of the bucket to copy the file from and to.
#   $2 - The key of the source file to copy.
#   $3 - The key of the destination file.
#
# Returns:
#   0 - If successful.
#   1 - If it fails.
#####
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}

```



```
#####
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#     $1 - The name of the bucket.
#
# Returns:
#     The list of files in text format.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function list_items_in_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api list-objects \
        --bucket "$bucket_name" \
        --output text \
        --query 'Contents[].{Key: Key, Size: Size}')

    # shellcheck disable=SC2181
    if [[ ${?} -eq 0 ]]; then
        echo "$response"
    else
        errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
        return 1
    fi
}

#####
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - A list of keys in the bucket to delete.
#
# Returns:
```

```

#      0 - If successful.
#      1 - If it fails.
#####
function delete_items_in_bucket() {
    local bucket_name=$1
    local keys=$2
    local response

    # Create the JSON for the items to delete.
    local delete_items
    delete_items="{\"Objects\":[\"
    for key in $keys; do
        delete_items=\"$delete_items{\\\"Key\\\": \\\"$key\\\"},\"
    done
    delete_items=${delete_items%?} # Remove the final comma.
    delete_items=\"$delete_items\"]\"

    response=$(aws s3api delete-objects \
        --bucket \"$bucket_name\" \
        --delete \"$delete_items\")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-object operation failed.\n$response"
        return 1
    fi
}

#####
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#      $1 - The name of the bucket.
#
# Returns:
#      0 - If successful.
#      1 - If it fails.
#####
function delete_bucket() {
    local bucket_name=$1
    local response

```

```
response=$(aws s3api delete-bucket \  
  --bucket "$bucket_name")  
  
# shellcheck disable=SC2181  
if [[ $? -ne 0 ]]; then  
  errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"  
  return 1  
fi  
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjectsV2](#)
  - [PutObject](#)

## AWS STSAWS CLI 与 Bash 脚本一起使用的示例

以下代码示例向您展示了如何使用 with Bash 脚本来执行操作和实现常见场景。AWS Command Line Interface AWS STS

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

### 主题

- [操作](#)

## 操作

### AssumeRole

以下代码示例演示了如何使用 AssumeRole。

#### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function sts_assume_role
#
# This function assumes a role in the AWS account and returns the temporary
# credentials.
#
# Parameters:
#     -n role_session_name -- The name of the session.
```

```

#     -r role_arn -- The ARN of the role to assume.
#
# Returns:
#     [access_key_id, secret_access_key, session_token]
#     And:
#     0 - If successful.
#     1 - If an error occurred.
#####
function sts_assume_role() {
    local role_session_name role_arn response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function sts_assume_role"
        echo "Assumes a role in the AWS account and returns the temporary credentials:"
        echo "  -n role_session_name -- The name of the session."
        echo "  -r role_arn -- The ARN of the role to assume."
        echo ""
    }

    while getopt n:r:h option; do
        case "${option}" in
            n) role_session_name=${OPTARG} ;;
            r) role_arn=${OPTARG} ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done

    response=$(aws sts assume-role \
        --role-session-name "$role_session_name" \
        --role-arn "$role_arn" \
        --output text \
        --query "Credentials.[AccessKeyId, SecretAccessKey, SessionToken]")

    local error_code=${?}

```

```
if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
fi

echo "$response"

return 0
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AssumeRole](#)中的。

# AWS Command Line Interface 中的安全性

AWS 十分重视云安全性。作为 AWS 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是 AWS 和您的共同责任。[责任共担模式](#) 将其描述为云的安全性和云中的安全性：

- 云的安全性 - AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。作为[AWS 合规性计划](#)的一部分，第三方审计人员将定期测试和验证安全性的有效性。要了解适用于 AWS Command Line Interface 的合规性计划，请参阅[合规性计划范围内的 AWS 服务](#)。
- 云中的安全性——您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您的公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 AWS Command Line Interface (AWS CLI) 时应用责任共担模式。以下主题说明如何配置 AWS CLI 以实现您的安全性和合规性目标。您还将了解如何使用 AWS CLI 来帮助您监控和保护 AWS 资源。

## 主题

- [AWS CLI 中的数据保护](#)
- [Identity and Access Management](#)
- [此 AWS 产品或服务的合规性验证](#)
- [本 AWS 产品或服务的弹性](#)
- [本 AWS 产品或服务的基础设施安全](#)
- [强制使用最低版本的 TLS](#)

## AWS CLI 中的数据保护

AWS [责任共担模式](#) 适用于 AWS Command Line Interface 中的数据保护。如该模式中所述，AWS 负责保护运行所有 AWS Cloud 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。您还负责您所使用的 AWS 服务的安全配置和管理任务。有关数据隐私的更多信息，请参阅[数据隐私常见问题](#)。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的博客文章 [AWS Shared Responsibility Model and GDPR](#)。

出于数据保护目的，我们建议您保护 AWS 账户凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置单个用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 ( MFA )。
- 使用 SSL/TLS 与 AWS 资源进行通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用 AWS CloudTrail 设置 API 和用户活动日志记录。
- 使用 AWS 加密解决方案以及 AWS 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在通过命令行界面或 API 访问 AWS 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅 [《美国联邦信息处理标准 \(FIPS\) 第 140-2 版》](#)。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括使用控制台、API、AWS CLI 或 AWS SDK 处理 AWS CLI 或其他 AWS 服务时。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，我们强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

## 数据加密

所有安全服务均具有一项重要功能，即信息在未处于活动使用状态时都会加密。

### 静态加密

除了代表用户与 AWS 服务交互所需的凭证以外，AWS CLI 本身不存储任何客户数据。

如果您使用 AWS CLI 来调用 AWS 服务，此服务将客户数据传输到本地电脑进行存储，则请参阅该服务的《用户指南》中的“安全与合规”章节，了解如何存储、保护和加密数据的相关信息。

### 传输中加密

预设情况下，从运行 AWS CLI 和 AWS 服务终端节点的客户端电脑上传的所有数据，均通过使用 HTTPS/TLS 连接发送所有内容来加密。

您无需执行任何操作即可使用 HTTPS/TLS。除非您通过使用 `--no-verify-ssl` 命令行选项为单个命令显式禁用它，否则它始终处于启用状态。



# Identity and Access Management

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证 ( 登录 ) 和授权 ( 拥有权限 ) 使用 AWS 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

- [受众](#)
- [使用身份进行身份验证](#)
- [使用策略管理访问](#)
- [如何 AWS 服务 使用 IAM](#)
- [对 AWS 身份和访问进行故障排除](#)

## 受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您所做的工作 AWS。

服务用户-如果您 AWS 服务 曾经完成工作，则您的管理员会为您提供所需的凭证和权限。当您使用更多 AWS 功能来完成工作时，您可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问中的功能 AWS，请参阅[对 AWS 身份和访问进行故障排除](#)或 AWS 服务 您正在使用的用户指南。

服务管理员-如果您负责公司的 AWS 资源，则可能拥有完全访问权限 AWS。您的工作是确定您的服务用户应访问哪些 AWS 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要详细了解您的公司如何使用 IAM AWS，请参阅 AWS 服务 您正在使用的用户指南。

IAM 管理员：如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 AWS 的访问权限的详细信息。要查看您可以在 IAM 中使用的 AWS 基于身份的策略示例，请参阅 AWS 服务 您正在使用的用户指南。

## 使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证 ( 登录 AWS )。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。AWS IAM Identity Center (IAM Identity Center) 用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当您使用联合访问 AWS 时，您就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的[如何登录到您 AWS 账户](#)的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅 IAM 用户指南中的[签署 AWS API 请求](#)。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如，AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的[多重身份验证](#)和《IAM 用户指南》中的[在 AWS 中使用多重身份验证 \(MFA\)](#)。

## AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的[需要根用户凭证的任务](#)。

## 联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity Center 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。AWS Directory Service 当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户 和应用程序中使用。有关 IAM Identity Center 的信息，请参阅《AWS IAM Identity Center 用户指南》中的[什么是 IAM Identity Center ?](#)。

## IAM 用户和群组

[IAM 用户](#)是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的[对于需要长期凭证的使用场景定期轮换访问密钥](#)。

[IAM 组](#)是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的[何时创建 IAM 用户（而不是角色）](#)。

## IAM 角色

[IAM 角色](#)是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过[切换角色在中临时担任 IAM 角色](#)。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的[使用 IAM 角色](#)。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的[为第三方身份提供商创建角色](#)。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的[权限集](#)。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的[IAM 角色与基于资源的策略有何不同](#)。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务 服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。

- 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅[转发访问会话](#)。
- 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 [IAM 角色](#)。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的[创建向 AWS 服务委派权限的角色](#)。
- 服务相关角色-服务相关角色是一种与服务相关联的服务角色。AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的[使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限](#)。

要了解是使用 IAM 角色还是 IAM 用户，请参阅《IAM 用户指南》中的[何时创建 IAM 角色 \(而不是用户\)](#)。

## 使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS，当与身份或资源关联时，它会定义其权限。AWS 在委托人 (用户、root 用户或角色会话) 发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅《IAM 用户指南》中的[JSON 策略概览](#)。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或 AWS API 获取角色信息。

## 基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的[创建 IAM 策略](#)。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的[在托管式策略与内联策略之间进行选择](#)。

## 基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service ( Amazon S3 ) 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中[指定主体](#)。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

## 访问控制列表 (ACL)

访问控制列表 ( ACL ) 控制哪些主体 ( 账户成员、用户或角色 ) 有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。AWS WAF 要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的[访问控制列表 \( ACL \) 概览](#)。

## 其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 - 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体 ( IAM 用户或角色 ) 授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 Principal 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的[IAM 实体的权限边界](#)。
- 服务控制策略 (SCP)-SCP 是 JSON 策略，用于指定组织或组织单位 (OU) 的最大权限。AWS Organizations AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管



理的服务。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 [SCP 的工作原理](#)。

- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM 用户指南》中的 [会话策略](#)。

## 多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的 [策略评估逻辑](#)。

## 如何 AWS 服务 使用 IAM

要全面了解如何 AWS 服务 使用大多数 IAM 功能，请参阅 IAM 用户指南中的与 IAM [配合使用的AWS 服务](#)。

要了解如何在 IAM 中 AWS 服务 使用特定的，请参阅相关服务的《用户指南》的安全部分。

## 对 AWS 身份和访问进行故障排除

使用以下信息来帮助您诊断和修复在使用 AWS 和 IAM 时可能遇到的常见问题。

### 主题

- [我无权在以下位置执行操作 AWS](#)
- [我无权执行 iam : PassRole](#)
- [我想允许我以外的人 AWS 账户 访问我的 AWS 资源](#)

## 我无权在以下位置执行操作 AWS

如果您收到错误提示，表明您无权执行某个操作，则您必须更新策略以允许执行该操作。

当 mateojackson IAM 用户尝试使用控制台查看有关虚构 *my-example-widget* 资源的详细信息，但不拥有虚构 *awes:GetWidget* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
awes:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 mateojackson 用户的策略，以允许使用 `aws:GetWidget` 操作访问 `my-example-widget` 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

## 我无权执行 iam : PassRole

如果您收到一个错误，表明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给 AWS。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 marymajor 的 IAM 用户尝试使用控制台在 AWS 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

## 我想允许我以外的人 AWS 账户 访问我的 AWS 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表 ( ACL ) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解是否 AWS 支持这些功能，请参阅[如何 AWS 服务 使用 IAM](#)。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户，请参阅[IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限](#)。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅[IAM 用户指南中的向第三方提供访问权限。AWS 账户](#)。
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的[为经过外部身份验证的用户 \( 身份联合验证 \) 提供访问权限](#)。

- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 [IAM 角色与基于资源的策略有何不同](#)。

## 此 AWS 产品或服务的合规性验证

要了解是否属于特定合规计划的范围，请参阅AWS 服务“[按合规计划划分的范围](#)”，然后选择您感兴趣的合规计划。AWS 服务 有关一般信息，请参阅[AWS 合规计划AWS](#)。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的“[下载报告](#)”中的“[AWS Artifact](#)”。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。AWS 提供了以下资源来帮助实现合规性：

- [安全与合规性快速入门指南](#) — 这些部署指南讨论了架构注意事项，并提供了在这些基础上 AWS 部署以安全性和合规性为重点的基准环境的步骤。
- 在 [Amazon Web Services 上构建 HIPAA 安全与合规架构](#) — 本白皮书描述了各公司如何使用 AWS 来创建符合 HIPAA 资格的应用程序。

### Note

并非所有 AWS 服务 人都符合 HIPAA 资格。有关更多信息，请参阅[符合 HIPAA 要求的服务参考](#)。

- [AWS 合规资源AWS](#) — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- [AWS 客户合规指南](#) — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践，AWS 服务 并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO) ) 的安全控制。
- [使用AWS Config 开发人员指南中的规则评估资源](#) — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业指导方针和法规。
- [AWS Security Hub](#) — 这 AWS 服务 提供了您内部安全状态的全面视图 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 [Security Hub 控件参考](#)。
- [Amazon GuardDuty](#) — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务 检测您的工作负载、容器和数据面临的潜在威胁。GuardDuty 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。



- [AWS Audit Manager](#)— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

本 AWS 产品或服务通过其支持的特定 Amazon Web Services (AWS) 服务遵循[分担责任模式](#)。有关 AWS 服务安全信息，请参阅[AWS 服务安全文档页面](#)和合规[计划符合 AWS 规工作范围内的AWS 服务](#)。

## 本 AWS 产品或服务的弹性

AWS 全球基础设施是围绕 AWS 区域 可用区构建的。

AWS 区域 提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络连接。

利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅[AWS 全球基础设施](#)。

本 AWS 产品或服务通过其支持的特定 Amazon Web Services (AWS) 服务遵循[分担责任模式](#)。有关 AWS 服务安全信息，请参阅[AWS 服务安全文档页面](#)和合规[计划符合 AWS 规工作范围内的AWS 服务](#)。

## 本 AWS 产品或服务的基础设施安全

本 AWS 产品或服务使用托管服务，因此受到 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅[AWS 云安全](#)。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS security Pillar Well-Architected Framework 中的[基础设施保护](#)。

您可以使用 AWS 已发布的 API 调用通过网络访问此 AWS 产品或服务。客户端必须支持以下内容：

- 传输层安全性协议 (TLS)。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密 (PFS) 的密码套件，例如 DHE (临时 Diffie-Hellman) 或 ECDHE (临时椭圆曲线 Diffie-Hellman)。大多数现代系统 (如 Java 7 及更高版本) 都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 [AWS Security Token Service](#) (AWS STS) 生成临时安全凭证来对请求进行签名。

本 AWS 产品或服务通过其支持的特定 Amazon Web Services (AWS) 服务遵循[分担责任模式](#)。有关 AWS 服务安全信息，请参阅[AWS 服务安全文档页面](#)和合规[计划符合 AWS 规工作范围内的AWS 服务](#)。

## 强制使用最低版本的 TLS

要提高与 AWS 服务通信时的安全性，您应使用 TLS 1.2 或更高版本。使用 AWS CLI 时，Python 用于设置 TLS 版本。

AWS CLI 版本 2 使用内部 Python 脚本，该脚本编译为在与之通信的服务支持最低版本 TLS 1.2 时使用此版本。只要您使用 AWS CLI 版本 2，无需执行其他步骤即可实施此最低版本。

# 解决 AWS CLI 错误

本节介绍常见错误和解决您的问题的故障排除步骤。我们建议首先进行[一般故障排除](#)。

## 目录

- [首先尝试的一般故障排除](#)
  - [检查您的 AWS CLI 命令格式](#)
  - [检查 AWS 区域 你的 AWS CLI 命令正在使用什么](#)
  - [确认您运行的是 AWS CLI 的最新版本](#)
  - [使用 --debug 选项](#)
  - [启用并查看 AWS CLI 命令历史记录日志](#)
  - [确认您的配置 AWS CLI 已完成](#)
- [找不到命令错误](#)
- [“aws --version”命令返回的版本与您安装的版本不同](#)
- [卸载后，aws --version“” 命令会返回一个版本 AWS CLI](#)
- [AWS CLI 处理了一个参数名称不完整的命令](#)
- [访问被拒绝错误](#)
- [凭证无效和密钥错误](#)
- [签名与错误不匹配](#)
- [SSL 证书错误](#)
- [JSON 无效错误](#)
- [其他 资源](#)

## 首先尝试的一般故障排除

如果您收到错误或遇到问题 AWS CLI，我们建议您使用以下一般提示来帮助您进行故障排除。

### [回到顶部](#)

## 检查您的 AWS CLI 命令格式

如果您收到一个错误，表明某个命令不存在，或者它无法识别文档指明可用的参数 (Parameter validation failed)，则您的命令可能格式不正确。我们建议您检查以下内容：

- 检查命令是否存在拼写和格式错误。
- 确认您的命令中[适用于您的终端的所有引号和转义](#)都是正确的。
- 生成 [AWS CLI 骨架](#)以确认命令结构。
- 对于 JSON，请参阅 [JSON 值的其他问题排查](#)。如果您在终端处理 JSON 格式时遇到问题，我们建议通过使用 [Blob 将 JSON 数据直接传递给 AWS CLI](#)，以跳过终端的引号规则。

有关如何构造特定命令的更多信息，请参阅[第 2 AWS CLI 版参考指南](#)。

[回到顶部](#)

## 检查 AWS 区域 你的 AWS CLI 命令正在使用什么

### Note

使用 AWS 区域 时，必须明确指定或通过设置默认区域来指定。AWS CLI有关您可以指定的所有内容的列表 AWS 区域，请参阅中的[AWS 区域和终端节点Amazon Web Services 一般参考](#)。使用的 AWS 区域 标号与您在 AWS Management Console 网址和服务端点中看到的名称相同。AWS CLI

如果您指定的资源 AWS 服务 不可用，或者您的资源位于其他位置，则可能会出现错误 AWS 区域 或意外结果 AWS 区域。按优先顺序排列，按以下方式设置：AWS 区域

- `--region` 命令行选项。
- 与 SDK 兼容的[AWS\\_REGION](#)环境变量。
- 环境变量 [AWS\\_DEFAULT\\_REGION](#)
- [region](#)配置文件设置。

确认您的资源使用的是正确 AWS 区域 的。

[回到顶部](#)

## 确认您运行的是 AWS CLI 的最新版本

如果您收到错误消息，表明某个命令不存在，或者它无法识别参考指南[第 2 AWS CLI 版AWS CLI](#)所说的可用参数，请首先确认您的命令格式是否正确。如果格式正确，我们建议您升级到 AWS CLI 的最新

版本。的更新版本几乎每个工作日都会发布。AWS CLI 这些新版本中引入了新的 AWS 服务、功能和参数 AWS CLI。获取这些新服务、功能或参数的唯一方式是升级到首次引入该元素后发布的版本。

如何更新版本 AWS CLI 取决于您最初的安装方式，如中所述[the section called “安装/更新”](#)。

如果您使用了某个捆绑安装程序，则可能需要先删除现有安装，然后为您的操作系统下载并安装最新版本。

[回到顶部](#)

## 使用 `--debug` 选项

当 AWS CLI 报告一个你无法立即理解的错误，或者产生了你意想不到的结果时，你可以通过使用 `--debug` 选项再次运行该命令来获得有关错误的更多详细信息。使用此选项，AWS CLI 会输出有关它处理命令所执行的每一步的详细信息。输出中的详细信息可以帮助您确定错误发生的时间，并提供错误从何处开始的线索。

您可以将输出发送到文本文件以供日后查看，或者按要求发送给 AWS Support。

当您包含 `--debug` 选项时，一些详细信息包括：

- 查找凭证
- 解析提供的参数
- 构造发送到 AWS 服务器的请求
- 发送到的请求的内容 AWS
- 原始响应的内容
- 带格式的输出

以下是使用和不使用 `--debug` 选项运行命令的示例。

```
$ aws iam list-groups --profile MyTestProfile
{
  "Groups": [
    {
      "Path": "/",
      "GroupName": "MyTestGroup",
      "GroupId": "AGPA0123456789EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:group/MyTestGroup",
      "CreateDate": "2019-08-12T19:34:04Z"
    }
  ]
}
```

```
]
}
```

```
$ aws iam list-groups --profile MyTestProfile --debug
2019-08-12 12:36:18,305 - MainThread - awscli.clidriver - DEBUG - CLI version: aws-
cli/1.16.215 Python/3.7.3 Linux/4.14.133-113.105.amzn2.x86_64 boto3/1.12.205
2019-08-12 12:36:18,305 - MainThread - awscli.clidriver - DEBUG - Arguments entered to
CLI: ['iam', 'list-groups', '--debug']
2019-08-12 12:36:18,305 - MainThread - boto3.hooks - DEBUG - Event session-
initialized: calling handler <function add_scalar_parsers at 0x7fdf173161e0>
2019-08-12 12:36:18,305 - MainThread - boto3.hooks - DEBUG - Event session-
initialized: calling handler <function register_uri_param_handler at 0x7fdf17dec400>
2019-08-12 12:36:18,305 - MainThread - boto3.hooks - DEBUG - Event session-
initialized: calling handler <function inject_assume_role_provider_cache at
0x7fdf17da9378>
2019-08-12 12:36:18,307 - MainThread - boto3.credentials - DEBUG - Skipping
environment variable credential check because profile name was explicitly set.
2019-08-12 12:36:18,307 - MainThread - boto3.hooks - DEBUG - Event session-
initialized: calling handler <function attach_history_handler at 0x7fdf173ed9d8>
2019-08-12 12:36:18,308 - MainThread - boto3.loaders - DEBUG - Loading JSON
file: /home/ec2-user/venv/lib/python3.7/site-packages/boto3/data/iam/2010-05-08/
service-2.json
2019-08-12 12:36:18,317 - MainThread - boto3.hooks - DEBUG - Event building-command-
table.iam: calling handler <function add_waiters at 0x7fdf1731a840>
2019-08-12 12:36:18,320 - MainThread - boto3.loaders - DEBUG - Loading JSON
file: /home/ec2-user/venv/lib/python3.7/site-packages/boto3/data/iam/2010-05-08/
waiters-2.json
2019-08-12 12:36:18,321 - MainThread - awscli.clidriver - DEBUG - OrderedDict([('path-
prefix', <awscli.arguments.CLIArument object at 0x7fdf171ac780>), ('marker',
<awscli.arguments.CLIArument object at 0x7fdf171b09e8>), ('max-items',
<awscli.arguments.CLIArument object at 0x7fdf171b09b0>)])
2019-08-12 12:36:18,322 - MainThread - boto3.hooks - DEBUG - Event building-
argument-table.iam.list-groups: calling handler <function add_streaming_output_arg at
0x7fdf17316510>
2019-08-12 12:36:18,322 - MainThread - boto3.hooks - DEBUG - Event building-
argument-table.iam.list-groups: calling handler <function add_cli_input_json at
0x7fdf17da9d90>
2019-08-12 12:36:18,322 - MainThread - boto3.hooks - DEBUG - Event building-
argument-table.iam.list-groups: calling handler <function unify_paging_params at
0x7fdf17328048>
2019-08-12 12:36:18,326 - MainThread - boto3.loaders - DEBUG - Loading JSON
file: /home/ec2-user/venv/lib/python3.7/site-packages/boto3/data/iam/2010-05-08/
paginator-1.json
```

```
2019-08-12 12:36:18,326 - MainThread - awscli.customizations.paginate - DEBUG -
  Modifying paging parameters for operation: ListGroups
2019-08-12 12:36:18,326 - MainThread - botocore.hooks - DEBUG - Event building-
argument-table.iam.list-groups: calling handler <function add_generate_skeleton at
  0x7fdf1737eae8>
2019-08-12 12:36:18,326 - MainThread - botocore.hooks - DEBUG - Event
before-building-argument-table-parser.iam.list-groups: calling handler
<bound method OverrideRequiredArgsArgument.override_required_args of
  <awscli.customizations.cliinputjson.CliInputJSONArgument object at 0x7fdf171b0a58>>
2019-08-12 12:36:18,327 - MainThread - botocore.hooks - DEBUG - Event
before-building-argument-table-parser.iam.list-groups: calling handler
<bound method GenerateCliSkeletonArgument.override_required_args of
  <awscli.customizations.generatecliskeleton.GenerateCliSkeletonArgument object at
  0x7fdf171c5978>>
2019-08-12 12:36:18,327 - MainThread - botocore.hooks - DEBUG - Event operation-
args-parsed.iam.list-groups: calling handler functools.partial(<function
  check_should_enable_pagination at 0x7fdf17328158>, ['marker', 'max-items'], {'max-
items': <awscli.arguments.CLIArgument object at 0x7fdf171b09b0>}, OrderedDict([('path-
prefix', <awscli.arguments.CLIArgument object at 0x7fdf171ac780>), ('marker',
  <awscli.arguments.CLIArgument object at 0x7fdf171b09e8>), ('max-items',
  <awscli.customizations.paginate.PageArgument object at 0x7fdf171c58d0>), ('cli-
input-json', <awscli.customizations.cliinputjson.CliInputJSONArgument object at
  0x7fdf171b0a58>), ('starting-token', <awscli.customizations.paginate.PageArgument
object at 0x7fdf171b0a20>), ('page-size', <awscli.customizations.paginate.PageArgument
object at 0x7fdf171c5828>), ('generate-cli-skeleton',
  <awscli.customizations.generatecliskeleton.GenerateCliSkeletonArgument object at
  0x7fdf171c5978>)]))
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG - Event load-cli-
arg.iam.list-groups.path-prefix: calling handler <awscli.paramfile.URIArgumentHandler
object at 0x7fdf1725c978>
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG - Event load-cli-
arg.iam.list-groups.marker: calling handler <awscli.paramfile.URIArgumentHandler object
at 0x7fdf1725c978>
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG - Event load-cli-
arg.iam.list-groups.max-items: calling handler <awscli.paramfile.URIArgumentHandler
object at 0x7fdf1725c978>
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG -
Event load-cli-arg.iam.list-groups.cli-input-json: calling handler
<awscli.paramfile.URIArgumentHandler object at 0x7fdf1725c978>
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG -
Event load-cli-arg.iam.list-groups.starting-token: calling handler
<awscli.paramfile.URIArgumentHandler object at 0x7fdf1725c978>
```

```
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG - Event load-cli-arg.iam.list-groups.page-size: calling handler <awscli.paramfile.URIArgumentHandler object at 0x7fdf1725c978>
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG - Event load-cli-arg.iam.list-groups.generate-cli-skeleton: calling handler <awscli.paramfile.URIArgumentHandler object at 0x7fdf1725c978>
2019-08-12 12:36:18,329 - MainThread - botocore.hooks - DEBUG - Event calling-command.iam.list-groups: calling handler <bound method CliInputJSONArgument.add_to_call_parameters of <awscli.customizations.cliinputjson.CliInputJSONArgument object at 0x7fdf171b0a58>>
2019-08-12 12:36:18,329 - MainThread - botocore.hooks - DEBUG - Event calling-command.iam.list-groups: calling handler <bound method GenerateCliSkeletonArgument.generate_json_skeleton of <awscli.customizations.generatecliskeleton.GenerateCliSkeletonArgument object at 0x7fdf171c5978>>
2019-08-12 12:36:18,329 - MainThread - botocore.credentials - DEBUG - Looking for credentials via: assume-role
2019-08-12 12:36:18,329 - MainThread - botocore.credentials - DEBUG - Looking for credentials via: assume-role-with-web-identity
2019-08-12 12:36:18,329 - MainThread - botocore.credentials - DEBUG - Looking for credentials via: shared-credentials-file
2019-08-12 12:36:18,329 - MainThread - botocore.credentials - INFO - Found credentials in shared credentials file: ~/.aws/credentials
2019-08-12 12:36:18,330 - MainThread - botocore.loaders - DEBUG - Loading JSON file: /home/ec2-user/venv/lib/python3.7/site-packages/botocore/data/endpoints.json
2019-08-12 12:36:18,334 - MainThread - botocore.hooks - DEBUG - Event choose-service-name: calling handler <function handle_service_name_alias at 0x7fdf1898eb70>
2019-08-12 12:36:18,337 - MainThread - botocore.hooks - DEBUG - Event creating-client-class.iam: calling handler <function add_generate_presigned_url at 0x7fdf18a028c8>
2019-08-12 12:36:18,337 - MainThread - botocore.regions - DEBUG - Using partition endpoint for iam, us-west-2: aws-global
2019-08-12 12:36:18,337 - MainThread - botocore.args - DEBUG - The s3 config key is not a dictionary type, ignoring its value of: None
2019-08-12 12:36:18,340 - MainThread - botocore.endpoint - DEBUG - Setting iam timeout as (60, 60)
2019-08-12 12:36:18,341 - MainThread - botocore.loaders - DEBUG - Loading JSON file: /home/ec2-user/venv/lib/python3.7/site-packages/botocore/data/_retry.json
2019-08-12 12:36:18,341 - MainThread - botocore.client - DEBUG - Registering retry handlers for service: iam
2019-08-12 12:36:18,342 - MainThread - botocore.hooks - DEBUG - Event before-parameter-build.iam.ListGroups: calling handler <function generate_idempotent_uuid at 0x7fdf189b10d0>
```



```

2019-08-12 12:36:18,342 - MainThread - boto3.core.hooks - DEBUG - Event before-
call.iam.ListGroups: calling handler <function inject_api_version_header_if_needed at
0x7fdf189b2a60>
2019-08-12 12:36:18,343 - MainThread - boto3.core.endpoint - DEBUG - Making
request for OperationModel(name=ListGroups) with params: {'url_path': '/',
'query_string': '', 'method': 'POST', 'headers': {'Content-Type': 'application/x-
www-form-urlencoded; charset=utf-8', 'User-Agent': 'aws-cli/1.16.215 Python/3.7.3
Linux/4.14.133-113.105.amzn2.x86_64 boto3/core/1.12.205'}, 'body': {'Action':
'ListGroups', 'Version': '2010-05-08'}, 'url': 'https://iam.amazonaws.com/',
'context': {'client_region': 'aws-global', 'client_config': <boto3.core.config.Config
object at 0x7fdf16e9a4a8>, 'has_streaming_input': False, 'auth_type': None}}
2019-08-12 12:36:18,343 - MainThread - boto3.core.hooks - DEBUG - Event request-
created.iam.ListGroups: calling handler <bound method RequestSigner.handler of
<boto3.core.signers.RequestSigner object at 0x7fdf16e9a470>>
2019-08-12 12:36:18,343 - MainThread - boto3.core.hooks - DEBUG - Event choose-
signer.iam.ListGroups: calling handler <function set_operation_specific_signer at
0x7fdf18996f28>
2019-08-12 12:36:18,343 - MainThread - boto3.core.auth - DEBUG - Calculating signature
using v4 auth.
2019-08-12 12:36:18,343 - MainThread - boto3.core.auth - DEBUG - CanonicalRequest:
POST
/

content-type:application/x-www-form-urlencoded; charset=utf-8
host:iam.amazonaws.com
x-amz-date:20190812T193618Z

content-type;host;x-amz-date
5f776d91EXAMPLE9b8cb5eb5d6d4a787a33ae41c8cd6eEXAMPLEEca69080e1e1f
2019-08-12 12:36:18,344 - MainThread - boto3.core.auth - DEBUG - StringToSign:
AWS4-HMAC-SHA256
20190812T193618Z
20190812/us-east-1/iam/aws4_request
ab7e367eEXAMPLE2769f178ea509978cf8bfa054874b3EXAMPLE8d043fab6cc9
2019-08-12 12:36:18,344 - MainThread - boto3.core.auth - DEBUG - Signature:
d85a0EXAMPLEb40164f2f539cdc76d4f294fe822EXAMPLE18ad1ddf58a1a3ce7
2019-08-12 12:36:18,344 - MainThread - boto3.core.endpoint - DEBUG - Sending
http request: <AWSPreparedRequest stream_output=False, method=POST,
url=https://iam.amazonaws.com/, headers={'Content-Type': b'application/
x-www-form-urlencoded; charset=utf-8', 'User-Agent': b'aws-cli/1.16.215
Python/3.7.3 Linux/4.14.133-113.105.amzn2.x86_64 boto3/core/1.12.205',
'X-Amz-Date': b'20190812T193618Z', 'Authorization': b'AWS4-HMAC-SHA256
Credential=AKIA01234567890EXAMPLE-east-1/iam/aws4_request, SignedHeaders=content-

```

```

type;host;x-amz-date, Signature=d85a07692aceb401EXAMPLEa1b18ad1ddf58a1a3ce7EXAMPLE',
  'Content-Length': '36'}>
2019-08-12 12:36:18,344 - MainThread - urllib3.util.retry - DEBUG - Converted retries
value: False -> Retry(total=False, connect=None, read=None, redirect=0, status=None)
2019-08-12 12:36:18,344 - MainThread - urllib3.connectionpool - DEBUG - Starting new
HTTPS connection (1): iam.amazonaws.com:443
2019-08-12 12:36:18,664 - MainThread - urllib3.connectionpool - DEBUG - https://
iam.amazonaws.com:443 "POST / HTTP/1.1" 200 570
2019-08-12 12:36:18,664 - MainThread - botocore.parsers - DEBUG - Response headers:
{'x-amzn-RequestId': '74c11606-bd38-11e9-9c82-559da0adb349', 'Content-Type': 'text/
xml', 'Content-Length': '570', 'Date': 'Mon, 12 Aug 2019 19:36:18 GMT'}
2019-08-12 12:36:18,664 - MainThread - botocore.parsers - DEBUG - Response body:
b'<ListGroupResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">\n
  <ListGroupResult>\n    <IsTruncated>>false</IsTruncated>\n    <Groups>\n
    <member>\n      <Path>/</Path>\n      <GroupName>MyTestGroup</GroupName>
\n      <Arn>arn:aws:iam::123456789012:group/MyTestGroup</Arn>\n
    <GroupId>AGPA1234567890EXAMPLE</GroupId>\n      <CreateDate>2019-08-12T19:34:04Z</
CreateDate>\n    </member>\n    </Groups>\n  </ListGroupResult>\n
  <ResponseMetadata>\n    <RequestId>74c11606-bd38-11e9-9c82-559da0adb349</RequestId>\n
  </ResponseMetadata>\n</ListGroupResponse>\n'
2019-08-12 12:36:18,665 - MainThread - botocore.hooks - DEBUG - Event needs-
retry.iam.ListGroups: calling handler <botocore.retryhandler.RetryHandler object at
0x7fdf16e9a780>
2019-08-12 12:36:18,665 - MainThread - botocore.retryhandler - DEBUG - No retry needed.
2019-08-12 12:36:18,665 - MainThread - botocore.hooks - DEBUG - Event after-
call.iam.ListGroups: calling handler <function json_decode_policies at 0x7fdf189b1d90>
{
  "Groups": [
    {
      "Path": "/",
      "GroupName": "MyTestGroup",
      "GroupId": "AGPA123456789012EXAMPLE",
      "Arn": "arn:aws:iam::123456789012:group/MyTestGroup",
      "CreateDate": "2019-08-12T19:34:04Z"
    }
  ]
}

```

[回到顶部](#)

## 启用并查看 AWS CLI 命令历史记录日志

您可以使用 [cli\\_history](#) 文件设置启用 AWS CLI 命令历史记录日志。启用此设置后，会 AWS CLI 记录 `aws` 命令的历史记录。

您可以使用 `aws history list` 命令列出您的历史记录，然后使用 `aws history show` 命令中生成的 `command_id` 获取详细信息。有关更多信息，请参阅《AWS CLI 参考指南》中的 [aws history](#)。

当您包含 `--debug` 选项时，一些详细信息包括：

- 针对 `botocore` 的 API 调用
- 状态代码
- HTTP 响应
- 标头
- 返回代码

您可以使用此信息来确认参数数据和 API 调用是否按预期方式运行，然后可以推断出您的命令在哪个步骤失败。

[回到顶部](#)

## 确认您的配置 AWS CLI 已完成

如果您的 `config` 和 `credentials` 文件或您的 IAM 用户或角色未得以正确配置，可能发生各种错误。有关解决 `config` 和 `credentials` 文件或您的 IAM 用户或角色的错误的更多信息，请参阅 [the section called “访问被拒绝错误”](#) 和 [the section called “凭证无效和密钥错误”](#)。

[回到顶部](#)

## 找不到命令错误

此错误意味着操作系统找不到该 AWS CLI 命令。安装可能不完整或需要更新。

可能的原因：您正在尝试使用比已安装版本更新的 AWS CLI 功能，或者格式不正确

错误示例文本：

```
$ aws s3 copy
usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

aws help
aws <command> help
aws <command> <subcommand> help
aws: error: argument subcommand: Invalid choice, valid choices are:

ls                | website
cp                | mv
.....
```

如果命令格式不正确，或者您使用的是发布此功能之前的早期版本，可能会发生各种错误。有关解决围绕这两个问题的错误的更多信息，请参阅[the section called “检查您的 AWS CLI 命令格式”](#)和[the section called “确认您运行的是 AWS CLI 的最新版本”](#)。

### [回到顶部](#)

可能的原因：安装后需要重新启动终端

错误示例文本：

```
$ aws --version
command not found: aws
```

如果在首次安装或更新后找不到该aws命令 AWS CLI，则可能需要重新启动终端才能识别任何PATH更新。

### [回到顶部](#)

可能的原因：AWS CLI 未完全安装

错误示例文本：

```
$ aws --version
command not found: aws
```

如果在首次安装或更新后找不到该aws命令 AWS CLI，则该命令可能尚未完全安装。按照[the section called “安装/更新”](#)中适用于您的平台的步骤尝试重新安装。

### [回到顶部](#)

可能的原因：AWS CLI 没有权限 (Linux)

如果在 Linux AWS CLI 上首次安装或更新后找不到该aws命令，则该命令可能没有execute权限访问其安装的文件夹。在 AWS CLI 安装时运行以下命令，为[chmod](#)用户提供权限 AWS CLI : PATH

```
$ sudo chmod -R 755 /usr/local/aws-cli/
```

[回到顶部](#)

可能的原因：安装期间未更新操作系统 PATH。

错误示例文本：

```
$ aws --version  
command not found: aws
```

您可能需要将 aws 可执行文件添加到操作系统的 PATH 环境变量中。要将 AWS CLI 添加到您的 PATH，请按照以下适用于您的操作系统的说明进行操作。

Linux and macOS

1. 在您的用户目录中查找 Shell 的配置文件脚本。如果您不能确定所使用的 Shell，请运行 `echo $SHELL`。

```
$ ls -a ~  
.  ..  .bash_logout  .bash_profile  .bashrc  Desktop  Documents  Downloads
```

- Bash – `.bash_profile`、`.profile` 或 `.bash_login`
- Zsh – `.zshrc`
- Tcsh – `.tcshrc`、`.cshrc` 或 `.login`

2. 向配置文件脚本中添加导出命令。以下命令将您的本地 bin 添加到当前 PATH 变量。

```
export PATH=/usr/local/bin:$PATH
```

3. 将更新的配置文件重新加载到当前会话中。

```
$ source ~/.bash_profile
```

## Windows

1. 在 Windows 命令提示符下，使用带 `/R path` 参数的 `where` 命令来查找 `aws` 文件位置。结果将返回所有包含 `aws` 的文件夹。

```
C:\> where /R c:\ aws
c:\Program Files\Amazon\AWSCLIV2\aws.exe
...
```

默认情况下，AWS CLI 版本 2 位于：

```
c:\Program Files\Amazon\AWSCLIV2\aws.exe
```

2. 按 Windows 键并输入 **environment variables**。
3. 从建议列表中，选择 Edit environment variables for your account ( 编辑您账户的环境变量 )。
4. 选择 PATH，然后选择 Edit ( 编辑 )。
5. 将在第一步中找到的路径添加到 Variable value ( 变量值 ) 字段中，例如 **C:\Program Files\Amazon\AWSCLIV2\aws.exe**。
6. 选择 OK ( 确定 ) 两次以应用新设置。
7. 关闭任何运行的命令提示符并重新打开命令提示符窗口。

### [回到顶部](#)

## “aws --version”命令返回的版本与您安装的版本不同

您的终端返回的结果可能与您预期PATH的 AWS CLI 不同。

可能的原因：安装后需要重新启动终端

如果 `aws` 命令显示错误的版本，您可能需要重新启动终端以便它识别任何 PATH 更新。需要关闭所有打开的终端，而不仅仅是活动的终端。

### [回到顶部](#)

可能的原因：安装后需要重新启动系统

如果 `aws` 命令显示了错误的版本，并且重新启动终端不起作用，您可能需要重新启动系统，系统才能识别您的 PATH 更新。


[回到顶部](#)

可能的原因：您有多个版本的 AWS CLI

如果您更新了 AWS CLI 并使用了与先前安装不同的安装方法，则可能会导致安装多个版本。例如，如果您在 Linux 或 macOS 上使用 pip 进行了当前安装，但试图使用 .pkg 安装文件进行更新，这可能会导致一些冲突，尤其是当 PATH 指向旧版本时。

要解决这个问题，[请卸载 AWS CLI 的所有版本](#)，然后执行净安装。

卸载所有版本后，请按照适用于您的操作系统的说明安装所需版本：[AWS CLI 版本 1](#) 或 [AWS CLI 版本 2](#)。

 Note

如果在安装 AWS CLI 版本 2 并预先安装了版本 1 之后发生这种情况，则在说明进行操作。

[回到顶部](#)

## 卸载后，`aws --version` 命令会返回一个版本 AWS CLI

这种情况通常发生在您的系统上还有 AWS CLI 安装的软件时。

可能的原因：卸载后需要重新启动终端

如果 `aws --version` 命令仍然有效，您可能需要重新启动终端以便它识别任何终端更新。

[回到顶部](#)

可能的原因：您的系统 AWS CLI 上有多个版本的，或者使用的卸载方法与最初安装时使用的卸载方法不同 AWS CLI

如果您 AWS CLI 使用与安装方法不同的方法卸载，或者安装了多个版本，则 AWS CLI 可能无法正确卸载。例如，如果您使用 pip 执行了当前安装，则必须使用 pip 卸载它。要解决此问题，请 AWS CLI 使用与安装方法相同的方法进行卸载。

1. 按照适用于您的操作系统的说明和原始安装方法卸载 [AWS CLI 版本 1](#) 和 [AWS CLI 版本 2](#)。
2. 关闭所有打开的终端。
3. 打开首选终端，输入以下命令并确认没有返回任何版本。

```
$ aws --version
command not found: aws
```

如果输出中仍列出了一个版本，则 AWS CLI 很可能是使用其他方法安装的，或者有多个版本。如果您不知道安装的是哪种方法 AWS CLI，请按照适用于您的操作系统的[AWS CLI 版本 1](#)和[AWS CLI 版本 2](#)的每种卸载方法的说明进行操作，直到没有收到任何版本输出。

#### Note

如果您是使用软件包管理器 ( pip、apt、brew 等 ) 安装 AWS CLI 的，则必须使用同一个软件包管理器来卸载它。请务必按照软件包管理器提供的有关如何卸载软件包的所有版本的说明进行操作。

[回到顶部](#)

## AWS CLI 处理了一个参数名称不完整的命令

可能的原因：您使用了普遍接受的 AWS CLI 参数缩写

由于 AWS CLI 是使用 Python 构建的，因此 AWS CLI 使用 Python argparse 库，包括[allow\\_abbrev](#)参数。参数的缩写由识别 AWS CLI 并进行处理。

以下命令示例更改了 CloudFormation 堆栈名称。该参数 `--change-set-n` 被识别为的缩写 `--change-set-name`，然后 AWS CLI 处理该命令。

```
$ aws cloudformation create-change-set --stack-name my-stack --change-set-n my-change-set
```

当您的缩写可能代表多个命令时，参数将不会被识别为缩写。

以下命令示例更改了 CloudFormation 堆栈名称。参数 `--change-set-` 无法识别为缩写，因为它可以是多个参数 ( 例如 `--change-set-name` 和 `--change-set-type` ) 的缩写。因此，AWS CLI 不处理该命令。

```
$ aws cloudformation create-change-set --stack-name my-stack --change-set- my-change-set
```



### Warning

不要刻意使用参数缩写。缩写不可靠且不向后兼容。如果在命令中添加了任何会使缩写产生混淆的新参数，则会中断您的命令。

此外，如果参数是单值参数，则会导致命令出现意外行为。如果传递了单值参数的多个实例，则只有最后一个实例会运行。在以下示例中，参数 `--filters` 是单值参数。指定了参数 `--filters` 和 `--filter`。`--filter` 参数是 `--filters` 的缩写。这会导致应用 `--filters` 的两个实例，只有最后一个 `--filter` 参数会应用。

```
$ aws ec2 describe-vpc-peering-connections \  
  --filters Name=tag:TagName,Values=VpcPeeringConnection \  
  --filter Name=status-code,Values=active
```

在运行命令之前，请确认您使用的是有效参数，以防止出现意外行为。

[回到顶部](#)

## 访问被拒绝错误

可能的原因：AWS CLI 程序文件没有“运行”权限

在 Linux 或 macOS 上，确保 `aws` 程序具有发出调用的用户的运行权限。通常，权限设置为 755。

要添加用户的运行权限，请运行以下命令，并将 `~/.local/bin/aws` 替换为您计算机上指向此程序的路径。

```
$ chmod +x ~/.local/bin/aws
```

[回到顶部](#)

可能的原因：您的 IAM 身份没有执行此操作的权限

错误示例文本：

```
$ aws s3 ls  
An error occurred (AccessDenied) when calling the ListBuckets operation: Access  
denied.
```

当您运行 AWS CLI 命令时，将使用与您与 IAM 账户或角色关联的凭证代表您执行 AWS 操作。附加的策略必须向您授予权限，才能调用与通过 AWS CLI 运行的命令相对应的 API 操作。

大多数命令会通过一个与命令名称匹配的名称来调用单个操作。但是，像 `aws s3 sync` 这样的自定义命令会调用多个 API。您可以查看命令通过使用 `--debug` 选项调用哪些 API。

如果您确定用户或角色具有策略分配的适当权限，请确保您的 AWS CLI 命令使用的是您期望的证书。请参阅[下一节中有关凭据](#)的内容，以验证所使用的凭证 AWS CLI 是否符合您的期望。

有关分配 IAM 权限的信息，请参阅《IAM 用户指南》中的[访问管理概述：权限和策略](#)。

[回到顶部](#)

## 凭证无效和密钥错误

错误示例文本：

```
$ aws s3 ls
An error occurred (InvalidAccessKeyId) when calling the ListBuckets operation: The AWS
Access Key Id
you provided does not exist in our records.
```

```
$ aws s3 ls
An error occurred (InvalidClientTokenId) when calling the ListBuckets operation: The
security token
included in the request is invalid.
```

可能的原因：AWS CLI 正在读取错误的凭据或来自意外的位置

AWS CLI 可能正在从与您预期不同的位置读取证书，或者您的密钥对（key pair）信息不正确。您可以运行 `aws configure list` 以确认使用哪些凭证。

以下示例说明如何检查用于默认配置文件的凭证。

```
$ aws configure list
      Name                               Value                               Type    Location
      ----                               -
      profile                             <not set>                          None    None
      access_key                          *****XYVA shared-credentials-file
      secret_key                           *****ZAGY shared-credentials-file
```

```
region                us-west-2    config-file    ~/.aws/config
```

以下示例说明如何检查命名配置文件的凭证。

```
$ aws configure list --profile saanvi
  Name                Value                Type    Location
  ----                -
  profile             saanvi              manual  --profile
  access_key          ***** shared-credentials-file
  secret_key          ***** shared-credentials-file
  region              us-west-2          config-file    ~/.aws/config
```

要确认证据对详细信息，请检查 `config` 和 `credentials` 文件。有关 `config` 和 `credentials` 文件的更多信息，请参阅[the section called “配置和凭证文件设置”](#)。有关凭证和身份验证（包括凭证优先顺序）的更多信息，请参阅[身份验证和访问凭证](#)。

### [回到顶部](#)

可能的原因：您计算机的时钟不同步

如果您使用的凭证是有效的，则可能是您的时钟不同步。在 Linux 或 macOS 上，运行 `date` 以检查时间。

```
$ date
```

如果您的系统时钟在几分钟内不正确，则使用 `ntpd` 进行同步。

```
$ sudo service ntpd stop
$ sudo ntpdate time.nist.gov
$ sudo service ntpd start
$ ntpstat
```

在 Windows 上，使用控制面板中的日期和时间选项来配置系统时钟。

### [回到顶部](#)

## 签名与错误不匹配

错误示例文本：

```
$ aws s3 ls
```

```
An error occurred (SignatureDoesNotMatch) when calling the ListBuckets operation: The request signature we calculated does not match the signature you provided. Check your key and signing method.
```

AWS CLI 运行命令时，它会向 AWS 服务器发送加密请求以执行相应的 AWS 服务操作。您的证书（访问密钥和私有密钥）参与加密，并 AWS 允许对提出请求的人进行身份验证。有多种因素可能会干扰此过程的正常执行，如下所示。

可能的原因：您的时钟与 AWS 服务器不同步

为了帮助防范[重播攻击](#)，在加密/解密过程中可能会使用当前时间。如果客户端和服务器的时间不一致超出允许的时间量，该过程可能会失败，并且请求会被拒绝。当您在时钟与主机时钟不同步的虚拟机中运行命令时，也可能发生此错误。一个可能的原因是，当虚拟机休眠时，唤醒后需要一些时间才能将时钟与主机同步。

在 Linux 或 macOS 上，运行 `date` 以检查时间。

```
$ date
```

如果您的系统时钟在几分钟内不正确，则使用 `ntpd` 进行同步。

```
$ sudo service ntpd stop
$ sudo ntpdate time.nist.gov
$ sudo service ntpd start
$ ntpstat
```

在 Windows 上，使用控制面板中的日期和时间选项来配置系统时钟。

## [回到顶部](#)

可能的原因：您的操作系统对包含某些特殊字符的 AWS 密钥处理不当

如果您的 AWS 密钥包含某些特殊字符，例如、`-+/%`、或，则某些操作系统变体会不正确地处理字符串，从而导致密钥字符串解释不正确。

如果您使用其他工具或脚本处理密钥，例如在新实例上构建证书文件作为其创建过程的一部分的工具，则这些工具和脚本可能对特殊字符有自己的处理方式，从而导致它们被转换为 AWS 无法识别的东西。

我们建议重新生成私有密钥，以使获得的私有密钥不包含会导致问题的特殊字符。

[回到顶部](#)

## SSL 证书错误

可能的原因：AWS CLI 不信任您的代理服务器的证书

错误示例文本：

```
$ aws s3 ls
[SSL: CERTIFICATE_ VERIFY_FAILED] certificate verify failed
```

使用 AWS CLI 命令时，您会收到一条 [SSL: CERTIFICATE\_ VERIFY\_FAILED] certificate verify failed 错误消息。这是由于代理人的证书是自签名的，您的公司被设置为证书颁发机构 (CA)，导致您 AWS CLI 不信任代理人的证书。这可以防止在本地 CA 注册表中找到您公司的 CA 根证书。AWS CLI

要解决此问题，请使用 [ca\\_bundle](#) 配置 .pem 文件设置、[--ca-bundle](#) 命令行选项或 [AWS\\_CA\\_BUNDLE](#) 环境变量指示在 AWS CLI 哪里可以找到您的公司文件。

[回到顶部](#)

可能的原因：您的配置未指向正确的 CA 根证书位置

错误示例文本：

```
$ aws s3 ls
SSL validation failed for regionname [Errno 2] No such file or directory
```

这是由于您的证书颁发机构 (CA) 捆绑包文件位置在 AWS CLI 中配置不正确所致。要解决此问题，请确认您的公司 .pem 文件所在的位置，并使用 [ca\\_bundle](#) 配置文件设置、[--ca-bundle](#) 命令行选项或 [AWS\\_CA\\_BUNDLE](#) 环境变量更新 AWS CLI 配置。

[回到顶部](#)

可能的原因：您的配置使用不正确 AWS 区域

错误示例文本：

```
$ aws s3 ls
[SSL: CERTIFICATE_ VERIFY_FAILED] certificate verify failed
```

如果您指定的资源 AWS 服务 不可用，或者您的资源位于其他位置，则可能会出现错误 AWS 区域 或意外结果 AWS 区域。有关问题排查步骤，请参阅 [the section called “检查 AWS 区域 你的 AWS CLI 命令正在使用什么”](#)。

### [回到顶部](#)

可能的原因：你的 TLS 版本需要更新

错误示例文本：

```
$ aws s3 ls
[SSL: UNSAFE_LEGACY_RENEGOTIATION_DISABLED] unsafe legacy renegotiation disabled
```

使用 AWS 服务的 TLS 版本与您设备的 TLS 版本不兼容。要解决此问题，请更新到支持的 TLS 版本。有关更多信息，请参阅 [the section called “强制实施最低 TLS 版本”](#)。

### [回到顶部](#)

## JSON 无效错误

错误示例文本：

```
$ aws dynamodb update-table \
  --provisioned-throughput '{"ReadCapacityUnits":15,WriteCapacityUnits":10}' \
  --table-name MyDDBTable
Error parsing parameter '--provisioned-throughput': Invalid JSON: Expecting property
name enclosed in
double quotes: line 1 column 25 (char 24)
JSON received: {"ReadCapacityUnits":15,WriteCapacityUnits":10}
```

使用 AWS CLI 命令时，您会收到“Invalid JSON”错误消息。当您输入具有预期的 JSON 格式的命令并且 AWS CLI 无法正确读取您的 JSON 时，通常会出现错误。

可能的原因：您没有输入有效的 JSON AWS CLI 供使用

确认您为命令输入了有效的 JSON。我们建议您对格式设置遇到问题的 JSON 使用 JSON 验证器。

要在命令行中使用更高级的 JSON，请考虑使用 jq 等命令行 JSON 处理器来创建 JSON 字符串。有关的更多信息jq，请参阅上的 [jq 存储库](#)。GitHub

### [回到顶部](#)

可能的原因：您的终端的报价规则导致无法将有效的 JSON 发送到 AWS CLI

在从命令 AWS CLI 接收任何内容之前，您的终端会使用自己的引用和转义规则处理该命令。由于终端的格式设置规则，您的某些 JSON 内容可能会在命令传递给 AWS CLI 之前剥离掉。在构建命令时，请务必使用[终端的引号规则](#)。

要进行故障排除，请使用 echo 命令来查看 Shell 如何处理您的参数：

```
$ echo {"ReadCapacityUnits":15,"WriteCapacityUnits":10}
ReadCapacityUnits:15 WriteCapacityUnits:10
```

```
$ echo '{"ReadCapacityUnits":15,"WriteCapacityUnits":10}'
{"ReadCapacityUnits":15,"WriteCapacityUnits":10}
```

修改您的命令，直到返回有效的 JSON。

要进行更深入的故障排除，请使用 --debug 参数来查看调试日志，因为它们将确切地显示传递给 AWS CLI 的内容：

```
$ aws dynamodb update-table \
  --provisioned-throughput '{"ReadCapacityUnits":15,WriteCapacityUnits":10}' \
  --table-name MyDDBTable \
  --debug
2022-07-19 22:25:07,741 - MainThread - awscli.clidriver - DEBUG - CLI version: aws-
cli/1.18.147
Python/2.7.18 Linux/5.4.196-119.356.amzn2int.x86_64 botocore/1.18.6
2022-07-19 22:25:07,741 - MainThread - awscli.clidriver - DEBUG - Arguments entered
to CLI:
['dynamodb', 'update-table', '--provisioned-throughput',
 '{"ReadCapacityUnits":15,WriteCapacityUnits":10}',
 '--table-name', 'MyDDBTable', '--debug']
```

使用终端的引号规则来修复 JSON 输入在发送到 AWS CLI 时出现的任何问题。有关引号规则的更多信息，请参阅[the section called “含字符串的引号”](#)。

#### Note

如果您在获取有效 JSON 时遇到问题 AWS CLI，我们建议您使用 Blobs 将您的 JSON 数据直接传递给，从而绕过终端对 JSON 数据输入的报价规则。AWS CLI 有关 Blob 的更多信息，请参阅[Blob](#)。

[回到顶部](#)

## 其他资源

要获得 AWS CLI 有关问题的更多帮助，请访问[AWS CLI 社区GitHub](#)或[AWS re:Post 社区](#)。

[回到顶部](#)



# 从 AWS CLI 版本 1 迁移至版本 2

本节包含将 AWS CLI 版本 1 更新为 AWS CLI 版本 2 的说明。AWS CLI 版本 2 建立在 AWS CLI 版本 1 的基础上，包括基于社区反馈的功能和增强功能。

在迁移至版本 2 之前，[了解各版本之间的差异](#)。AWS CLI 版本 2 包括的新功能和更改可能要求您更新脚本或命令以实现向后兼容性。

AWS CLI 版本 1 和 2 使用相同的 `aws` 命令名称。如果您安装了两个版本，电脑将使用在搜索路径中找到的第一个版本。

如果您之前安装了 AWS CLI 版本 1，请按照我们的[迁移说明开始使用版本 2](#)。

如果您之前未安装 AWS CLI 版本 1，请按照[开始使用](#)中的说明进行操作。

## 主题

- [AWS CLI 版本 2 中的新功能和变化](#)
- [AWS CLI 版本 2 迁移说明](#)

## AWS CLI 版本 2 中的新功能和变化

本主题介绍 AWS CLI 版本 1 和 AWS CLI 版本 2 之间的新功能和行为变化。这些变化可能要求您更新脚本或命令，以便在版本 2 中获得与版本 1 中相同的行为。

## 主题

- [AWS CLI 版本 2 新功能](#)
- [AWS CLI 版本 1 和 AWS CLI 版本 2 之间的突破性更改](#)

## AWS CLI 版本 2 新功能

AWS CLI 版本 2 是 AWS CLI 的最新主版本，支持所有最新功能。版本 2 中引入的某些功能无法向后兼容版本 1，您必须升级才能访问这些功能。这些特征如下所示：

### 不需要 Python 解释器

AWS CLI 版本 2 不需要单独安装 Python。它包括一个嵌入式版本。

## [向导](#)

您可以将向导与 AWS CLI 版本 2 结合使用。此向导将引导您构建某些命令。

## [IAM Identity Center 身份验证](#)

如果您的企业使用 AWS IAM Identity Center(IAM Identity Center)，则用户可以登录到 Active Directory、内置 IAM Identity Center 目录或[连接到 IAM Identity Center 的其他 IdP](#)。然后，它们映射到允许您运行 AWS CLI 命令的 AWS Identity and Access Management (IAM) 角色。

## [自动提示](#)

如果启用，当您运行 `aws` 命令时，AWS CLI 版本 2 可以提示您输入命令、参数和资源。

## [使用亚马逊 ECR 官方公共镜像或 Docker 镜像运行 AWS CLI](#)

AWS CLI 的官方 Docker 映像提供 AWS 直接支持和维护的隔离、可移植性和安全性。这样，您可以在基于容器的环境中使用 AWS CLI 版本 2，而无需自行管理安装。

## [客户端分页程序](#)

AWS CLI 版本 2 支持使用客户端分页程序处理输出。原定设置情况下，此功能处于启用状态，并通过操作系统的原定设置分页程序返回所有输出。

## [aws configure import](#)

导入从 AWS Management Console 生成的 `.csv` 凭证。将导入一个配置文件名称与 IAM 用户名匹配的 `.csv` 文件。

## [aws configure list-profiles](#)

列出您已经配置的所有配置文件的名称。

## [the section called “YAML 流输出格式”](#)

`yaml` 和 `yaml-stream` 格式利用 [YAML](#) 格式，同时通过向您流式传输数据，使您在查看大型数据集时的响应能力更高。您可以在整个查询下载之前开始查看和使用 YAML 数据。

## [适用于 DynamoDB 的新的高级 `ddb` 命令](#)

AWS CLI 版本 2 具有高级 Amazon DynamoDB 命令 [ddb put](#) 和 [ddb select](#)。这些命令提供了一个简化的界面，用于将项目放入 DynamoDB 表中以及在 DynamoDB 表或索引中进行搜索。

## [aws logs tail](#)

AWS CLI 版本 2 具有自定义 `aws logs tail` 命令，用于跟踪 Amazon CloudWatch 日志组的日志。原定设置情况下，该命令从过去十分钟内所有关联的 CloudWatch Logs 流返回日志。

## [添加了对高级 s3 命令的元数据支持](#)

AWS CLI 版本 2 将 `--copy-props` 参数添加到高级 s3 命令中。使用此参数，您可以为 Amazon Simple Storage Service (Amazon S3) 配置其他元数据和标签。

## [AWS\\_REGION](#)

AWS CLI 版本 2 有一个与 AWS SDK 兼容的环境变量，名为 `AWS_REGION`。此变量指定要向其发送请求的 AWS 区域。它覆盖 `AWS_DEFAULT_REGION` 环境变量，该变量仅适用于 AWS CLI。

## AWS CLI 版本 1 和 AWS CLI 版本 2 之间的突破性更改

本节介绍 AWS CLI 版本 1 和 AWS CLI 版本 2 之间行为方面的所有变化。这些变化可能要求您更新脚本或命令，以便在版本 2 中获得与版本 1 中相同的行为。

### 主题

- [添加了用于设置文本文件编码的环境变量](#)
- [原定设置情况下，二进制参数作为 base64 编码字符串进行传递](#)
- [改进了执行分段复制时 Amazon S3 处理文件属性和标签的方式](#)
- [不自动检索 http:// 或 https:// URL 以获取参数](#)
- [原定设置情况下，使用分页程序处理所有输出](#)
- [时间戳输出值标准化为 ISO 8601 格式](#)
- [改进了 CloudFormation 部署的处理，而这不会导致任何更改](#)
- [更改了区域 Amazon S3 端点对于 us-east-1 区域的原定设置行为](#)
- [已更改区域 AWS STS 端点的原定设置行为](#)
- [`ecr get-login` 已删除并替换为 `ecr get-login-password`](#)
- [AWS CLI 版本 2 对插件的支持会不断变化](#)
- [已删除隐藏别名支持](#)
- [不支持 `api\_versions` 配置文件设置](#)
- [AWS CLI 版本 2 仅使用签名 v4 对 Amazon S3 请求进行身份验证](#)
- [AWS CLI 版本 2 与分页参数更一致](#)
- [AWS CLI 版本 2 在所有命令间提供了更一致的返回代码](#)

## 添加了用于设置文本文件编码的环境变量

原定设置情况下，[the section called “Blob”](#)的文本文件使用与已安装的区域设置相同的编码。由于 AWS CLI 版本 2 使用的是 Python 的嵌入式版本，因此，不支持 PYTHONUTF8 和 PYTHONIOENCODING 环境变量。要将文本文件的编码设置为不同于区域设置，请使用 AWS\_CLI\_FILE\_ENCODING 环境变量。下面的示例将 AWS CLI 设置为在 Windows 上使用 UTF-8 打开文本文件。

```
AWS_CLI_FILE_ENCODING=UTF-8
```

有关更多信息，请参阅[用于配置的环境变量 AWS CLI](#)。

## 原定设置情况下，二进制参数作为 base64 编码字符串进行传递

在 AWS CLI 中，有些命令需要使用 [base64](#) 编码的字符串，其他命令则需要使用 UTF-8 编码的字节字符串。在 AWS CLI 版本 1 中，在两种编码字符串类型之间传递数据通常需要一些中间处理。AWS CLI 版本 2 使处理二进制参数更加一致，从而有助于更可靠地将值从一个命令传递到另一个命令。

原定设置情况下，AWS CLI 版本 2 将所有二进制输入和二进制输出参数作为 base64 编码字符串 blobs（二进制大型对象）进行传递。有关更多信息，请参阅[the section called “Blob”](#)。

要恢复为 AWS CLI 版本 1 行为，请使用 [cli\\_binary\\_format](#) 文件配置或 [--cli-binary-format](#) 参数。

## 改进了执行分段复制时 Amazon S3 处理文件属性和标签的方式

当您使用 `aws s3` 命名空间中的 AWS CLI 版本 1 命令将文件从一个 S3 存储桶位置复制到另一个存储桶位置，并且该操作使用[分段复制](#)时，源对象中的任何文件属性都不会复制到目标对象。

原定设置情况下，AWS CLI 版本 2 中的相应命令将所有标签和某些属性从源副本传输到目标副本。与 AWS CLI 版本 1 相比较，这可能会导致对 Amazon S3 端点进行更多 AWS API 调用。要更改 AWS CLI 版本 2 中 `s3` 命令的原定设置行为，请使用 `--copy-props` 参数。

有关更多信息，请参阅[the section called “分段复制中的文件属性和标签”](#)。

## 不自动检索 `http://` 或 `https://` URL 以获取参数

当参数值以 `http://` 或 `https://` 开头，但不使用返回的内容作为参数值时，AWS CLI 版本 2 不执行 GET 操作。因此，将从 AWS CLI 版本 2 中删除关联的命令行选项 `cli_follow_urlparam`。

如果您需要检索 URL 并将 URL 内容传递给参数值，我们建议您使用 `curl` 或类似的工具将 URL 的内容下载到本地文件。然后，使用 `file://` 语法读取该文件的内容，并将其用作参数值。

例如，以下命令不再尝试检索在 `http://www.example.com` 中找到的页面的内容，并将这些内容作为参数传递。相反，它将文字文本字符串 `https://example.com` 作为参数传递。

```
$ aws ssm put-parameter \  
  --value http://www.example.com \  
  --name prod.microservice1.db.secret \  
  --type String 2
```

如果您需要检索 Web URL 的内容并将其用作参数，则可以在版本 2 中执行以下操作。

```
$ curl https://my.example.com/mypolicyfile.json -o mypolicyfile.json  
$ aws iam put-role-policy \  
  --policy-document file:///./mypolicyfile.json \  
  --role-name MyRole \  
  --policy-name MyReadOnlyPolicy
```

在前面的示例中，`-o` 参数指示 `curl` 将文件保存在与源文件同名的当前文件夹中。第二个命令检索该下载文件的内容，并将内容作为 `--policy-document` 的值传递。

## 原定设置情况下，使用分页程序处理所有输出

原定设置情况下，AWS CLI 版本 2 会通过操作系统的原定设置分页程序返回所有输出。此程序是 Linux 或 macOS 上的 [less](#) 程序，以及 Windows 上的 [more](#) 程序。这样，通过一次一页显示服务的输出，从而帮助您浏览大量输出。

您可以将 AWS CLI 版本 2 配置为使用其他分页程序或根本不使用分页程序。有关更多信息，请参阅[the section called “客户端分页程序”](#)。

## 时间戳输出值标准化为 ISO 8601 格式

原定设置情况下，AWS CLI 版本 2 以 [ISO 8601 格式](#) 返回所有时间戳响应值。在 AWS CLI 版本 1 中，命令以 HTTP API 响应返回的任何格式返回时间戳值，而格式可能因服务而异。

要查看以由 HTTP API 响应返回的格式表示的时间戳，请在 `config` 文件中使用 `wire` 值。有关更多信息，请参阅[cli\\_timestamp\\_format](#)。

## 改进了 CloudFormation 部署的处理，而这不会导致任何更改

原定设置情况下，在 AWS CLI 版本 1 中，如果您部署的 AWS CloudFormation 模板未导致任何更改，则 AWS CLI 返回失败错误代码。如果您不认为这是一个错误，并希望您的脚本继续运行，这

可能导致出现问题。在 AWS CLI 版本 1 中，可以通过添加返回 0 的标志 `--no-fail-on-empty-changeset` 来解决此问题。

因为这是常见使用案例，所以，当部署没有导致更改并且操作返回空更改集时，AWS CLI 版本 2 原定设置为返回成功的退出代码 0。

要恢复到原始行为，请添加标志 `--fail-on-empty-changeset`。

## 更改了区域 Amazon S3 端点对于 **us-east-1** 区域的原定设置行为

当您将 AWS CLI 版本 1 配置为使用 `us-east-1` 区域时，AWS CLI 使用物理托管在 `us-east-1` 区域中的全局 `s3.amazonaws.com` 端点。AWS CLI 版本 2 会在指定区域时使用真正的区域端点 `s3.us-east-1.amazonaws.com`。要强制 AWS CLI 版本 2 使用全局终端节点，您可以将命令的区域设置为 `aws-global`。

## 已更改区域 AWS STS 端点的原定设置行为

原定设置情况下，AWS CLI 版本 2 会将所有 AWS Security Token Service (AWS STS) API 请求发送到当前配置的 AWS 区域的区域端点。

原定设置情况下，AWS CLI 版本 1 将 AWS STS 请求发送到全局 AWS STS 端点。您可以使用 [sts\\_regional\\_endpoints](#) 设置来控制版本 1 中的此原定设置行为。

## **ecr get-login** 已删除并替换为 **ecr get-login-password**

AWS CLI 版本 2 将 `aws ecr get-login` 命令替换为 `aws ecr get-login-password` 命令，以改进与容器身份验证的自动集成。

`aws ecr get-login-password` 命令可降低在进程列表、shell 历史记录或其他日志文件中公开您的凭证的风险。它还提高了与 `docker login` 命令的兼容性，从而改善自动化处理的效果。

该 `aws ecr get-login-password` 命令在 AWS CLI 版本 1.17.10 及更高版本以及 AWS CLI 版本 2 中提供。为保持向后兼容性，较早的 `aws ecr get-login` 命令仍在 AWS CLI 版本 1 中可用。

使用 `aws ecr get-login-password` 命令，您可以替换以下用于检索密码的代码。

```
$ (aws ecr get-login --no-include-email)
```

要降低在 shell 历史记录或日志中公开密码的风险，请改用以下示例命令。在此示例中，密码会被直接传送给 `docker login` 命令，通过 `--password-stdin` 选项分配给密码参数。

```
$ aws ecr get-login-password | docker login --username AWS --password-stdin MY-REGISTRY-URL
```

有关更多信息，请参阅《AWS CLI 版本 2 参考指南》中的 [aws ecr get-login-password](#)。

## AWS CLI 版本 2 对插件的支持会不断变化

AWS CLI 版本 2 中的插件支持完全是暂时的，目的是在发布更新的稳定版插件接口之前帮助用户从 AWS CLI 版本 1 进行迁移。不保证某个特定的插件甚或是 AWS CLI 插件接口在未来版本的 AWS CLI 版本 2 中受支持。如果您依赖于插件，请务必锁定到 AWS CLI 的特定版本，并在升级时测试插件的功能。

要启用插件支持，请在 `~/.aws/config` 中创建 `[plugins]` 区段。

```
[plugins]
cli_legacy_plugin_path = <path-to-plugins>/python3.7/site-packages
<plugin-name> = <plugin-module>
```

在 `[plugins]` 区段中，定义 `cli_legacy_plugin_path` 变量并将其值设置为插件模块所在的 Python 站点包路径。然后，您可以通过提供插件名称 (`plugin-name`) 以及包含插件源代码的 Python 模块 (`plugin-module`) 的文件名，对插件进行配置。AWS CLI 会通过导入插件的 `plugin-module` 并调用插件的 `awscli_initialize` 函数来加载每个插件。

## 已删除隐藏别名支持

AWS CLI 版本 2 不再支持版本 1 中支持的以下隐藏别名。

在下表中，第一列显示适用于所有版本（包括 AWS CLI 版本 2）的服务、命令和参数。第二列显示不再适用于 AWS CLI 版本 2 的别名。

有效的服务、命令和参数	过时的别名
cognito-identity create-identity-pool open-id-connect-provider-arns	open-id-connect-provider-arns
storagegateway describe-tapes tape-arns	tape-ar-ns
storagegateway.describe-tape-archives.tape-arns	tape-ar-ns
storagegateway.describe-vtl-devices.vtl-device-arns	vtl-device-ar-ns



有效的服务、命令和参数	过时的别名
storagegateway.describe-cached-iscsi-volumes.volume-arns	volume-ar-ns
storagegateway.describe-stored-iscsi-volumes.volume-arns	volume-ar-ns
route53domains.view-billing.start-time	start
deploy.create-deployment-group.ec2-tag-set	ec-2-tag-set
deploy.list-application-revisions.s3-bucket	s-3-bucket
deploy.list-application-revisions.s3-key-prefix	s-3-key-prefix
deploy.update-deployment-group.ec2-tag-set	ec-2-tag-set
iam.enable-mfa-device.authentication-code1	authentication-code-1
iam.enable-mfa-device.authentication-code2	authentication-code-2
iam.resync-mfa-device.authentication-code1	authentication-code-1
iam.resync-mfa-device.authentication-code2	authentication-code-2
importexport.get-shipping-label.street1	street-1
importexport.get-shipping-label.street2	street-2
importexport.get-shipping-label.street3	street-3
lambda.publish-version.code-sha256	code-sha-256
lightsail.import-key-pair.public-key-base64	public-key-base-64
opsworks.register-volume.ec2-volume-id	ec-2-volume-id

## 不支持 **api\_versions** 配置文件设置

AWS CLI 版本 2 不支持通过使用 `api_versions` 配置文件设置调用较早版本的 AWS 服务 API。所有 AWS CLI 命令现在可调用终端节点当前支持的服务 API 的最新版本。



## AWS CLI 版本 2 仅使用签名 v4 对 Amazon S3 请求进行身份验证

AWS CLI 版本 2 不支持使用较早的签名算法对发送到 Amazon S3 端点的服务请求进行加密身份验证。这种签名会在每个 Amazon S3 请求中自动发生，但仅支持[签名版本 4 签名流程](#)。您无法配置签名版本。现在，所有 Amazon S3 存储桶预签名 URL 都只使用 Sigv4，最长有效期为一周。

## AWS CLI 版本 2 与分页参数更一致

在 AWS CLI 版本 1 中，如果您在命令行上指定分页参数，则会按预期关闭自动分页。但是，当您使用带有 `--cli-input-json` 参数的文件指定分页参数时，并不会关闭自动分页，这可能会导致意外输出。无论您如何提供参数，AWS CLI 版本 2 都会关闭自动分页。

## AWS CLI 版本 2 在所有命令间提供了更一致的返回代码

与 AWS CLI 版本 1 相比，AWS CLI 版本 2 在所有命令间更加一致，并正确返回适当的退出代码。我们还添加了退出代码 252、253 和 254。有关退出代码的更多信息，请参阅[the section called “返回代码”](#)。

如果您依赖于 AWS CLI 版本 1 使用返回代码值的方式，建议检查退出代码以确保您获得了预期的值。

## AWS CLI 版本 2 迁移说明

本主题提供用于从 AWS CLI 版本 1 迁移至 AWS CLI 版本 2 的说明。

AWS CLI 版本 1 和 2 使用相同的 `aws` 命令名称。如果您安装了两个版本，电脑将使用在搜索路径中找到的第一个版本。如果您之前安装过 AWS CLI 版本 1，建议您执行以下操作之一来使用 AWS CLI 版本 2：

- 推荐 – [卸载 AWS CLI 版本 1 并仅使用 AWS CLI 版本 2](#)。
- [要同时安装这两个版本](#)，请使用操作系统的功能为两个 `aws` 命令之一创建具有不同名称的符号链接 (symlink) 或别名。

有关版本 1 和版本 2 之间的突破性更改的信息，请参阅 [the section called “新功能和变化”](#)。

## 将版本 1 替换为版本 2

执行以下步骤以将 AWS CLI 版本 1 替换为 AWS CLI 版本 2。

## 将 AWS CLI 版本 1 替换为 AWS CLI 版本 2

1. 通过在 [the section called “新功能和变化”](#) 中确认版本 1 和版本 2 之间的任何突破性更改，为迁移准备任何现有的脚本。
2. 按照 [安装、更新和卸载 AWS CLI 版本 1](#) 中适用于您的操作系统的卸载说明操作，卸载 AWS CLI 版本 1。
3. 使用以下命令确认 AWS CLI 已完全卸载。

```
$ aws --version
```

根据输出完成下列操作之一：

- 没有返回任何版本：您已成功卸载 AWS CLI 版本 1，可以继续执行下一步。
  - 返回了版本：您仍有 AWS CLI 版本 1 的安装。有关问题排查步骤，请参阅 [the section called “卸载后，aws --version” 命令会返回一个版本 AWS CLI](#)。执行故障排除步骤，直到不会收到版本输出。
4. 按照 [安装或更新到最新版本的 AWS CLI](#) 中适用于您的操作系统的安装说明安装 AWS CLI 版本 2。

## 并排安装

要同时安装这两个版本，请使用操作系统的功能为两个 aws 命令之一创建具有不同名称的符号链接 (symlink) 或别名。

1. 按照 [安装或更新到最新版本的 AWS CLI](#) 中适用于您的操作系统的安装说明安装 AWS CLI 版本 2。
2. 使用操作系统的功能为两个 aws 命令中的一个命令创建具有不同名称的符号链接或别名，例如，将 `aws2` 用于 AWS CLI 版本 2。以下是用于 AWS CLI 版本 2 的符号链接示例。将 `PATH` 替换为您的安装位置。

### Linux and macOS

您可以在 Linux 和 macOS 上使用 [符号链接或别名](#)。

```
$ alias aws2='PATH'
```

### Windows command prompt

Windows 上的 [DOSKEY](#)。

```
C:\> doskey aws2=PATH
```

# 卸载 AWS CLI 版本 2

本主题介绍如何卸载 AWS Command Line Interface 版本 2 ( AWS CLI 版本 2 )。

AWS CLI 版本 2 卸载说明：

## Linux

要卸载 AWS CLI 版本 2，请运行以下命令。

### 1. 找到符号链接和安装路径。

- 使用 `which` 命令查找符号链接。这显示与 `--bin-dir` 参数一起使用的路径。

```
$ which aws
/usr/local/bin/aws
```

- 使用 `ls` 命令查找符号链接指向的目录。这为您提供与 `--install-dir` 参数一起使用的路径。

```
$ ls -l /usr/local/bin/aws
lrwxrwxrwx 1 ec2-user ec2-user 49 Oct 22 09:49 /usr/local/bin/aws -> /usr/local/
aws-cli/v2/current/bin/aws
```

### 2. 删除 `--bin-dir` 目录中的两个符号链接。如果您的用户对这些目录具有写入权限，则无需使用 `sudo`。

```
$ sudo rm /usr/local/bin/aws
$ sudo rm /usr/local/bin/aws_completer
```

### 3. 删除 `--install-dir` 目录。如果您的用户对此目录具有写入权限，则无需使用 `sudo`。

```
$ sudo rm -rf /usr/local/aws-cli
```

### 4. ( 可选 ) 删除 `.aws` 文件夹中的共享 AWS SDK 和 AWS CLI 设置信息。

#### Warning

这些配置和凭证设置跨所有 AWS SDK 和 AWS CLI 进行共享。如果删除此文件夹，则您系统上的任何 AWS SDK 都无法访问它们。

.aws 文件夹的默认位置因平台而异，默认情况下，该文件夹位于 `~/.aws/`。如果您的用户对此目录具有写入权限，则无需使用 `sudo`。

```
$ sudo rm -rf ~/.aws/
```

## macOS

要卸载 AWS CLI 版本 2，请运行以下命令，同时替换用于安装的路径。示例命令使用原定设置安装路径。

1. 找到包含指向主程序和补全程序的符号链接的文件夹。

```
$ which aws
/usr/local/bin/aws
```

2. 使用该信息，运行以下命令来查找符号链接指向的安装文件夹。

```
$ ls -l /usr/local/bin/aws
lrwxrwxrwx 1 ec2-user ec2-user 49 Oct 22 09:49 /usr/local/bin/aws -> /usr/local/
aws-cli/aws
```

3. 删除第一个文件夹中的两个符号链接。如果您的用户已具有这些文件夹的写入权限，则无需使用 `sudo`。

```
$ sudo rm /usr/local/bin/aws
$ sudo rm /usr/local/bin/aws_completer
```

4. 删除主安装文件夹。使用 `sudo` 获得对 `/usr/local` 文件夹的写入访问权限。

```
$ sudo rm -rf /usr/local/aws-cli
```

5. (可选) 删除 `.aws` 文件夹中的共享 AWS SDK 和 AWS CLI 设置信息。

### Warning

这些配置和凭证设置跨所有 AWS SDK 和 AWS CLI 进行共享。如果删除此文件夹，则您系统上的任何 AWS SDK 都无法访问它们。

.aws 文件夹的默认位置因平台而异，默认情况下，该文件夹位于 `~/.aws/`。如果您的用户对此目录具有写入权限，则无需使用 `sudo`。

```
$ sudo rm -rf ~/.aws/
```

## Windows

1. 通过执行以下操作之一打开程序和功能：

- 打开控制面板，然后选择程序和功能。
- 打开命令提示符，然后输入以下命令。

```
C:\> appwiz.cpl
```

2. 选择名为 AWS Command Line Interface 的条目，然后选择 Uninstall ( 卸载 ) 启动卸载程序。
3. 确认您要卸载 AWS CLI。
4. ( 可选 ) 删除 .aws 文件夹中的共享 AWS SDK 和 AWS CLI 设置信息。

### Warning

这些配置和凭证设置跨所有 AWS SDK 和 AWS CLI 进行共享。如果删除此文件夹，则您系统上的任何 AWS SDK 都无法访问它们。

.aws 文件夹的默认位置因平台而异，默认情况下，该文件夹位于 `%UserProfile%\aws`。

```
$ rmdir %UserProfile%\aws
```

## AWS CLI 安装和卸载错误故障排除

如果您在安装或卸载 AWS CLI 后遇到问题，请参阅[排查错误](#)以了解故障排除步骤。有关相关性最高的故障排除步骤，请参阅[the section called “找不到命令错误”](#)、[the section called ““aws --version”命令返回的版本与您安装的版本不同”](#)和[the section called “卸载后，aws --version”命令会返回一个版本 AWS CLI”](#)。

# AWS CLI 用户指南文档历史记录

下表介绍了自 2019 年 1 月以来对 AWS Command Line Interface 用户指南的重要补充。如需对此文档更新的通知，您可以订阅 RSS 源。

变更	说明	日期
<a href="#">更新了凭证和身份验证信息。</a>	更新了凭证和身份验证方法的说明和示例。这包括更新相关的入门页面和配置页面。为了适应此文档内容增加，相关凭证主题已移至新的 <a href="#">身份验证和访问凭证</a> 部分。	2023 年 3 月 31 日
<a href="#">添加了 AWS IAM Identity Center 的自动身份验证刷新功能的令牌提供程序配置</a>	配置 AWS CLI 以使用 SSO 令牌提供程序配置向 AWS IAM Identity Center ( IAM Identity Center ) 验证用户身份的新流程，该配置可以自动检索刷新的身份验证令牌。	2022 年 12 月 7 日
<a href="#">发布了 AWS CLI 版本 2 的官方 Amazon ECR Public 映像</a>	针对 Linux、macOS 和 Windows 发布了 AWS CLI 版本 2 的官方支持 Amazon ECR Public 映像。	2022 年 11 月 18 日
<a href="#">更新了从 AWS CLI V1 迁移到 V2 的指南</a>	扩展了重大更改指南，以包括从 AWS CLI 版本 1 迁移到 AWS CLI 版本 2 的迁移说明。包括“问题排查”页面的更新，以帮助解决安装问题。	2022 年 5 月 13 日
<a href="#">从源代码构建 AWS CLI 安装程序的新流程。</a>	在受支持的操作系统上从源代码安装或更新到 AWS CLI 最新版本的新流程。	2022 年 2 月 17 日
<a href="#">AWS CLI V1 和 V2 的内容现已归入各自的指南中</a>	为了清晰表达和易于使用，AWS CLI 版本 1 和 AWS	2021 年 11 月 2 日

	CLI 版本 2 的内容现已归入各自的指南中。对于 AWS CLI 版本 1，请参阅 <a href="#">AWS CLI 版本 1 用户指南</a> 。	
<a href="#">已添加 AWS CLI 别名信息</a>	已添加 AWS CLI 别名信息。别名是您可以在 AWS Command Line Interface (AWS CLI) 中创建的快捷方式，以缩短您经常使用的命令或脚本。	2021 年 3 月 11 日
<a href="#">更新筛选输出信息</a>	更新了筛选条件的信息并移至自己的页面。	2021 年 2 月 1 日
<a href="#">添加了向导的信息</a>	添加了 AWS CLI 版本 2 向导信息。	2020 年 11 月 20 日
<a href="#">更新了自动提示</a>	使用当前功能更新了 AWS CLI 版本 2 自动提示信息。	2020 年 11 月 10 日
<a href="#">添加了 Amazon S3 脚本示例</a>	添加了 Amazon S3 生命周期脚本示例。	2020 年 10 月 15 日
<a href="#">添加了 Amazon EC2 脚本示例</a>	添加了 Amazon EC2 实例类型脚本示例。	2020 年 10 月 15 日
<a href="#">添加了重试信息</a>	在 AWS CLI 中添加了重试功能和行为的重试页面。	2020 年 9 月 17 日
<a href="#">服务器端和客户端分页</a>	更新了分页信息并集中在单个页面上。	2020 年 8 月 17 日
<a href="#">更新了 s3 命令页面</a>	使用新示例和资源更新了高级别 s3 命令页面。	2020 年 7 月 30 日
<a href="#">更新的安装信息</a>	将更新 Linux、macOS 和 Windows 的安装、更新和卸载信息。	2020 年 5 月 19 日



<a href="#">添加了 AWS CLI 版本 2 上的文本文件编码的信息</a>	预设情况下，AWS CLI 版本 2 使用与本地文件编码相同的文本文件编码。您现在可以使用环境变量来设置文本文件编码。	2020 年 5 月 14 日
<a href="#">发布了 AWS CLI 版本 2 的官方 Docker 映像</a>	针对所有 Linux、macOS 和 Windows 发布了 AWS CLI 版本 2 的官方支持 Docker 镜像。	2020 年 3 月 31 日
<a href="#">添加了有关 AWS CLI 版本 2 的客户端分页程序的信息</a>	预设情况下，AWS CLI 版本 2 对所有客户端输出使用分页程序 less。	2020 年 2 月 19 日
<a href="#">AWS Command Line Interface (AWS CLI) 版本 2 正式发布</a>	AWS CLI 版本 2 已正式发布，建议客户安装此版本。	2020 年 2 月 10 日
<a href="#">现在，AWS CLI 版本 2 的 macOS 安装程序是一个 Apple 程序包安装程序 .pkg 文件。</a>	AWS CLI 版本 2 的 macOS 安装程序已从带有 Shell 脚本的 .zip 文件更新为完整的 macOS 安装程序包。这可以简化安装，并使其与最新的 macOS 版本兼容。	2020 年 2 月 3 日
<a href="#">添加了关于 AWS CLI 版本 2 改进了对 S3 和 STS 区域端点原定设置处理方式的内容</a>	原定设置情况下，AWS CLI 版本 2 现在将针对 Amazon S3 和 AWS STS 服务的请求定向到当前配置的区域端点，而不是全局端点。	2020 年 1 月 13 日
<a href="#">AWS CLI 版本 2 的开发人员预览版</a>	宣布推出 AWS CLI 版本 2 的预览版。添加了有关安装版本 2 的说明。添加迁移主题以讨论版本 1 和版本 2 之间的差异。	2019 年 11 月 7 日

<a href="#">添加了 AWS IAM Identity Center 对 AWS CLI 命名配置文件的支持</a>	AWS CLI 版本 2 添加了对创建命名配置文件的支持，该配置文件可以直接登录到 IAM Identity Center 并获取 AWS 临时凭证以在后续 AWS CLI 命令中使用。	2019 年 11 月 7 日
<a href="#">新的 MFA 部分</a>	增加了一个新部分，其中介绍如何使用多重验证和角色访问 CLI。	2019 年 5 月 3 日
<a href="#">更新了“使用 CLI”部分</a>	对使用说明和过程进行了重大改进和补充。	2019 年 3 月 7 日
<a href="#">更新了“安装 CLI”部分</a>	AWS CLI 安装说明和过程进行了重大改进和补充。	2019 年 3 月 7 日
<a href="#">更新了“配置 CLI”部分</a>	AWS CLI 配置说明和过程进行了重大改进和补充。	2019 年 3 月 7 日

# AWS 词汇表

有关最新 AWS 术语，请参阅《AWS 词汇表 参考资料》中的[AWS 词汇表](#)。